

**Hacking Practices and Software  
Development:  
A Social Worlds Analysis of ICT  
Innovation and the Role of Free/Libre  
Open Source Software**

Yuwei Lin

Thesis in partial fulfilment submitted for the degree  
of Doctor of Philosophy in Sociology

University of York  
Science and Technology Studies Unit, Department of Sociology

September 2004

## **ABSTRACT**

Through use of social worlds theory and qualitative research methods, this thesis explores hackers' practices and their relationships with the computing world and the wider society from a socio-technical perspective. The hacker social world comprises actors from diverse social-technical backgrounds who share a constellation of im/material practices, namely open source practices (OSPs). Through engaging with these collective practices, actors and actants communicate, negotiate, and shape each other's identities, practices and understandings of the innovation structure and system in various aspects. In examining the diverse articulations and performances in which hacker culture and hacker identity are both reflected and constructed, the thesis tries to contextualise and deconstruct the ICT architecture we take for granted, as well as the innovations made possible by this architecture.

The major findings of my research are: 1) As a community of open source practices, the FLOSS social world allows diverse actors to engage in the innovation process and therefore contains more innovation resources than other relatively conventional software engineering models. 2) The strategic collaboration between the public (i.e. the free software community) and the private (i.e. information technologies corporations) sectors symbolises a pattern of hybrid innovation that entails complex communications and networks. 3) Tacit knowledge anchored in everyday experiences is peculiarly valued in a community-based innovation system where social networking and information sharing are undergoing vigorously. 4) The development of FLOSS democratises software innovation process and allows lay people to develop their understanding and knowledge of a shared problem/issue, especially through the web, to challenge established views on the issue.

## List of Contents

Chapter 1 Introduction: Culture, Community & Creativity.....	10
1.0 The focus of the thesis.....	10
1.1 The structure of the thesis.....	13
Chapter 2 Innovation and Culture.....	19
2.0 Introduction.....	19
2.1 The Epistemology of Innovation.....	20
2.2 Cultural Dynamics in Innovation Systems: Social and Organisational Change.....	25
2.3 Dynamics in Software Innovation .....	28
2.3.1 Technical Issues in Software Innovation: Modelling Software Process.....	29
2.3.2 Socio-economic Issues in Software Innovation.....	30
2.3.3 Prescriptions for Studying the Dynamics in Software Innovation . .....	31
2.4 A Brief History of FLOSS.....	35
2.5 Open Source Definition: Is there ‘the’ FLOSS or just FLOSS? .....	41
2.6 Literature Review on FLOSS.....	42
2.7 Hackers.....	44
2.8 Hackers and FLOSS Innovation.....	50
2.9 Possibility, Uncertainty and Deviant Innovation.....	53
2.10 Conclusion: Innovation and Culture.....	55
Chapter 3 The Hacker Social World and FLOSS.....	58
3.0 Introduction.....	58
3.1 Previous Research Methodologies for Studying Computer Hackers .58	
3.2 Conceptualising Hackers.....	60
3.2.1 Pilot Fieldwork.....	60
3.2.2 Grounded Theory.....	63
3.3 Methodology.....	64
3.3.1 Social Worlds Theory.....	64
3.3.2 Methodological Rationale.....	69
3.3.3 Validation of Research Scenario and Core Research Questions.... .....	72
3.4 Methods and Meanings.....	75
3.4.1 Ethnography.....	76
3.4.2 Interviews.....	78
3.4.3 Documentary analysis.....	80
3.4.4 Case Studies.....	81
3.5 Research Ethics and Reflection.....	82

3.6 Conclusion.....	84
Chapter 4 Hacking, Heterogeneity and Cross-boundary Practices.....	86
4.0 Introduction.....	86
4.1 The world of hackers: A community of practices.....	87
4.2 Hackfest.....	93
4.2.1 HAL2001.....	97
4.2.2 Lifestyle: social diversity.....	98
4.2.3 Intellectual diversity.....	101
4.2.4 Law and order: Hacking vs. Cracking.....	105
4.2.5 Multiple identities.....	111
4.3 The pan-hacking phenomenon.....	117
4.3.1 The hacker social world.....	117
4.3.2 Shared practices.....	119
4.3.2.1 Hacking for protection.....	123
4.3.2.2 Institutionalised hacking.....	126
4.3.3 Cross-Boundary practices.....	129
4.3.3.1 Two independent social worlds.....	129
4.3.3.2 The opposition of the social worlds.....	131
4.3.3.3 The encounter of the social worlds.....	134
4.3.3.4 The translation of hacking practices.....	138
4.4 Conclusion.....	140
Chapter 5 A Case Study of the Problem-Solving Process in Developing FLOSS.....	142
5.0 Introduction.....	142
5.1 Problems and solutions: the sine qua non of the software innovation process.....	147
5.2 EMACS.....	149
5.3 Affordance and EMACS: Extensibility and Customisation.....	151
5.4 Constructing problems and crafting solutions: framing questions (classification) and building a social network of expertise.....	156
5.5 Innovation elements.....	160
5.5.1 Teamwork and communication.....	160
5.5.2 Shared interests and trust.....	167
5.5.3 Material culture: Interactions between Human and Technical Artefacts.....	171
5.5.4 TODO list.....	173
5.5.5 Machines and Compatibility.....	175
5.5.6 Programming languages.....	177
5.6 EMACS as a boundary object.....	178
5.7 Conclusion: The heterogeneous FLOSS social world.....	181
Chapter 6 Hybrid Innovation: The Dynamics of Collaboration Between the Public and the Private in the FLOSS Innovation System.....	184
6.0 Introduction.....	184

6.1 Beyond Game Theory.....	185
6.2 A Community of Open Source Practices (OSPs).....	187
6.3 Working practices in OSS firms.....	190
6.4 Linux Conferences: A Bridge between the Corporation and the Community.....	195
6.5 Pragmatic Collaborations in Practice.....	197
6.6 Shared Goals and Divided Goals.....	202
6.7 The Hybrid Identity of FLOSS Developers.....	212
6.8 The Specificity of FLOSS Innovation.....	216
6.9 Conclusion.....	218
 Chapter 7 Deciphering Expert Knowledge in Software Engineering: Glocalising FLOSS Innovation.....	 221
7.0 Introduction.....	221
7.1 Localisation within Globalisation.....	226
7.2 A Community-based Innovation: An Alternative Software Process..... .....	 230
7.2.1 Demography of a hybrid Organisation: the Virtual and the Real... .....	 231
7.2.2 Hacking Experience and Tacit Knowledge.....	233
7.2.3 An Open Hierarchy of Expertise.....	240
7.2.4 Local Expertise and Global Innovation .....	243
7.2.5 Social Facilitation: Mobilising the Community.....	252
7.2.6 Glocalised innovation: Local Peer Culture and Global Knowledge System.....	258
7.3 Empowering the Minority: Usability, Accessibility, and the Digital Divide.....	262
7.3.1 The Blind’s Self-Prescription – the Innovation Story of Brlty..... .....	 263
7.3.2 Bridging the Digital Divide: Implementing Software in Local Contexts.....	264
7.3.3 Democratising Software Innovation Process—Whose Democracy? Utopia or Dystopia?.....	267
7.4 Conclusion.....	272
 Chapter 8 Conclusion: Towards a (Open) Knowledge-based Information Society.....	 276
8.0 The Contextualisation of Hacker Culture and FLOSS Innovation. .	276
8.1 Summary of the Chapters.....	278
8.2 Research Implications.....	284
8.3 Future Research.....	287
 Appendix A: Interview schedule.....	 290

Appendix B: Interview schedule .....	292
Appendix C: List of interviewees.....	294
Appendix D: Customised Linux.....	296
Bibliography.....	298

## List of Figures

Figure 2.1: Boundaries and exchange: innovation, social worlds and the information society.....	57
Figure 4.1: Two Independent Social Worlds: the hacker social world and the mainstream social world.....	131
Figure 4.2: The connection of the two social worlds provided by the boundary practice.....	132
Figure 4.3: Cross-boundary: overlapping social worlds.....	135
Figure 5.1: A screenshot of Emacs.....	142

## Figures in Appendix D

Figure 1: The initial booting.....	296
Figure 2: Starting-up progress.....	297
Figure 3: the final desktop environment.....	297

## List of Tables

Box 4 1: Example of Cross-site scripting (Source: Amit Klein, ©Sanctum, Inc. 2002).....	121
Table 4 2: Focal periods of the sequence of interactions between the two social worlds.....	136

## ACKNOWLEDGEMENTS

Finishing the dissertation allows me to thank those who have helped me get on with writing as well as with living during some difficult times. Above all, I would like to thank my supervisor, Professor Andrew Webster, who provided enormous intellectual and emotional support during my PhD studies. Without him, I do not think I could have gone this far. I am also grateful to all SATSUMAs, who lighten up my stay in York. And Ragna Zeiss is the one who has shared the most laughs and tears together for the last 4 years. I would also like to thank all departmental colleagues who contributed in inestimable ways and in various aspects.

I would also like to acknowledge the explicit and implicit assistance of members from the free software community, who have supported me when I was slighted in York. Their energy and enthusiasm are infectious and invigorating. Because of them, my data collection and research process was not entirely full of bitterness but with sweet moments as well. Many of them have become good friends of mine and cooperation still continues. I am particularly grateful to Enrico Zini who not only offers intellectually spirited comments but also warms me up with his generous love, helping me go through the end of the writing-up stage.

Lastly, I would like to dedicate this thesis to my late grandmother, my beloved parents and two lovely brothers. Without their love, it is impossible for me to go through so many things during the PhD studying years. This thesis is the honour I would like to share with them. My late grandmother, Mrs. Gjo-Kang Lin, was always a central prop of my life. I would have loved to give her this thesis as a gift and I am certain she would have been delighted to receive it. Sadly, the dedication is all that I can now offer.

York, September 2004

## **Chapter 1 Introduction: Culture, Community & Creativity**

### **1.0 The focus of the thesis**

Software is at the heart of the development of information and communication technologies (ICTs). It engages with a system of communication in which it is embedded and gives meaning to this system through its implementation. This feature denotes that ICTs are particularly mobile and mutable compared with other technologies, expressed through the malleability of software languages and source code, and exposed to diverse and changing implementation environments. Such mutability also leads to a situation where there is considerable debate over the imperfection of software. As a result, risk and uncertainty about software production and its implementation emerge. The characteristic of software as always being in some way unfinished, or carrying an 'imperfection', is rarely found in other technology fields. There is no universal operating system for use even though the mainstream operating system remains Microsoft's *Windows*. There is no entirely bug-free software. Software is vulnerable: once a symbol in a programme is misplaced, it is unlikely to work properly. There is no universal design: one could never find software products empowering all citizens and fulfilling all human needs in a fragmented post/modern society. Risk and uncertainty also relate to human errors in the software process, long acknowledged and emphasised in literatures of software engineering. Whereas the requirements of more reliable and comprehensible products have always been there, it is suggested that the production of software is an ongoing socio-technical process rather than a closed or finished product. However, the communication and negotiation over the control and management of these factors are often concealed in and black-boxed by the mainstream innovation culture established via the creations of proprietary software. Although a large number of literatures in science and technology studies (STS) have pointed out that artefacts (tangible or intangible) and actors (individual or institutional) interact and mutually shape each other in the innovation process, a previous analytic inheritance with a linear view remains influential. As Kelly *et al.* (1986) note, this is

... because there seems to be something inherently attractive about simple explanations for complex phenomena. Even scholars engaged in innovation research are not entirely immune to this attraction. To serve as a corrective for the half-truths that might emerge from such one-sided approaches, we must endeavour to investigate the innovation process in all its ramifications.

(Kelly *et al.* 1986: 19)

When we to ignore the interactions between actors and actants in innovation processes, technology and society are placed in two distinct domains. As a result, many ground-breaking ICT products have failed to meet users' real needs (particularly their social needs) and neglect the context of their use. Whilst terms such as 'an Internet society', 'a networked society' and 'a digital society' reflect the vision and aspiration of policy makers and technological architects towards a modern world, their definitions of "efficiency" and "empowerment" are somehow dominant and pre-given. This characterisation and positioning overestimates the potential of ICTs and fails to acknowledge the socio-cultural contexts of application and utilisation in practice. A rationalised image of the ICT world is created and reproduced in every ICT product design.

In contrast to mainstream systems such as *Windows*, free/libre open source software (FLOSS) as an innovation system a) reveals the contingency and mutability of all software development but b) capitalises on this as a positive means to enable innovation through collaborative practice and community building. Recent developments in the social world of ICT point towards this. Part of this process involves a mobilisation of hacking and this will form an important focus of this thesis. In order to understand the socio-technical identity of *hackers* as neither a fixed essence nor utterly contingent and fragmented, I will argue that the conventional view of hackers is unacceptable. I suggest that the term should be examined in an empirical context as a range of collective practices influencing software technology, as the practices deployed that lead the FLOSS community into a more institutionalised form of organisation, such as the Linux-related ones. I propose that different descriptions, inscriptions and prescriptions of hackers are amplified or sacrificed through ongoing negotiation and compromise

between actors and actants. A detailed investigation of FLOSS phenomenon with attention to its context, using multiple sources of evidence and various methods of data collection, will also enable me to examine the innovation process by which new FLOSS technologies are created, arguing that this is ongoing and involves diverse groups who give the technology different meanings. In examining how the artefacts (e.g. software or source code) and key notions are described and inscribed in multiple FLOSS-based contexts, one can understand how the FLOSS development enacts and embodies the 70s hacker culture. Based on an empirical enquiry of real-life events in the hacker social world, the contextual thickness of my research makes qualitative methodology appropriate for "how" and "why" research questions, because answering these questions deals with processes needing to be tracked over time. Given these empirical studies, one can see how hacker culture has emerged, developed, followed and is embedded in the FLOSS innovation process.

Rooted in a hacker ethic, FLOSS-related technologies are not only a technological revolution, but also a social movement that operates largely in terms of symbols and meanings, both at the level of everyday life and at that of institutional operation. Under the framework of a FLOSS social world, FLOSS development with the participation of diverse actors and actants has made a 'co-fabrication of knowledge and identities' possible. It illustrates how experts can learn from users "in the wild" (Callon & Rabeharisoa 2003). Users get involved in and contribute to the technology of innovation. Since users and experts are often brought together through both virtual and physical sites of interaction, the results are often unforeseen and unpredictable, full of possibilities. This process breaks down the dichotomy of expert and lay and emphasises the value of soft skills, tacit knowledge and 'tinkering' practices in everyday ICT contexts. It also blurs the boundary of formal and informal, public and private in terms of practices and knowledge. Moreover, therefore, I hope to offer both new empirical material about hacking activities, their relation to the now extensive FLOSS system(s) and thereby contribute towards the theoretical work on ICT innovation within the field of STS.

## **1.1 The structure of the thesis**

Following the argument above, there are three research issues, which are linked together, explored in this thesis:

1. To examine how actors interact mutually and collectively in a heterogeneous technology field and how they assign meanings to the artefacts they create through their daily practices.

2. To assess the potential contribution of FLOSS to mainstream software innovation and to explore the relationships and interactions between its diverse actors and actants in the innovation process.

3. To consider the potential contribution of hacking practices and FLOSS to ICT innovation more generally, at both the local and global levels.

To investigate these issues, the argument is organised in the following way.

After this introductory first chapter, chapter 2 summarises the existing literature on technological innovation from various perspectives. In doing so, the chapter challenges mainstream thinking on technological innovation as linear and proposes to view innovation as a social process from a sociological perspective, particularly through the account of STS. Furthermore, the chapter examines various social issues related to the development of ICTs. Given this discussion, software innovation is then considered in a heterogeneous and contingent socio-technical context. Taking the nature of software into account, software engineering illustrates a different innovative process in comparison with other technologies that contain tangible items. Subsequently, FLOSS, what we can regard as a cluster of software created through an unconventional innovative approach is examined. Previous research on FLOSS is discussed and the socio-technical dynamics overlooked in this existing literature are summarised in part through a brief review of the historical origins of FLOSS. Subsequently, several main research fields that have sought to account for the development of FLOSS are discussed. In reviewing

these literatures, the role of hacker culture is regarded by most authors as a main factor shaping FLOSS innovation. However, the concept of a hacker culture is either championed in a somewhat over-romanticised way or defined as criminal or part of a deviant subcultural group. Both discourses presume to categorise the hacker in an unproblematic way, though clearly with contrasting meanings. In this chapter, I argue the inadequacy of both explanations inasmuch as the former presupposes a fixed hacker culture that does not completely comply with reality or oversimplifies the diversity of actors in the FLOSS innovation system, while the latter simply stigmatises hackers as digital deviants. Unlike these rather limited perspectives, I propose that the notions of hackers and hacker culture should not be taken as pre-given but explored through an STS analysis of the actual practices and discourses of FLOSS innovation wherein their socio-cultural meanings are embedded. In investigating the identities and cross-boundary activities of diverse actors in the FLOSS innovation system, I argue that the notions of hackers and hacker culture are given different meanings by different actors from different social worlds. I define a hacker social world as a heterogeneous and contingent milieu. The hacker social world is a terrain within which values are debated, decisions are made and particular forms of action are undertaken to express individual readings of the identity of hackers. The socio-technical dynamics of FLOSS innovation thus is reflected in the communication and negotiation over this identity. In this process, not only are hacker identity and hacking practice being defined and redefined but so, thereby, is innovation shaped and reshaped. This perspective helps, I suggest, to deliver a critical but comprehensive analysis about FLOSS innovation.

Chapter 3 introduces the research design of the thesis. It presents the principal research questions that are informed by the methodological framework of social worlds theory and the reasons for selecting a qualitative research combining several methods including interviewing and observation techniques, both on- and off- line. This ethnographically oriented approach is employed to address the multiplicity, dynamism, and conflictual nature of culture as innovation resources and repertoires. It also describes a methodological shift made during the thesis from a grounded theory to a social worlds theoretical framework. This reflexive presentation of the building of my research design is made to acknowledge that research is never a simple linear process; rather, the research process is continually

changing/evolving with its surprises, design changes, and reformulation of concepts and hypotheses.

Chapter 4 employs the notion of ‘a social world’ through which to understand the diverse hacking activities observed during the fieldwork. This concept avoids having to subscribe to a notion of a strong hacker subculture, so is more useful at the empirical level to investigate the socio-technical relationships and structures built on the collective practices found within and outside the hacker social world. In light of the fieldwork, it is found that hackers and hacking have many different meanings across different social actors whose activities can be generally seen as comprising a constellation of hacking practices. In a hacker social world, one can see that the meaning of ‘hacking’ is interpreted, inscribed in, confronted/challenged and negotiated along with participants’ activities, which are often cross-boundaried. The hacker subculture, if there is one, is not predetermined or prescribed. Instead, it is constructed by diverse actors coming across different social worlds and embedded in various practices, some of which are collective. Moreover, participants’ identity as hackers is not simply a self-description or ascription. The identity is inscribed in their material practices, which are adopted, translated across, and found in diverse social worlds. The constellation of hacking practices that are found across different social worlds serves as the boundary practice that brings these social worlds into connection and enables their amalgamation at some level. In investigating the cross-boundary activities, one can see how the technologies related to hacking are socially and technically constructed, and how marginalised hacking technologies deployed in the development of ICTs are assigned different meanings and actually incorporated into mainstream ICT practice.

Chapter 5 explores how a historic FLOSS editor software, EMACS (short for Editing MACroS), is created, developed and employed/deployed in mundane programming within an actor-centred network. Actors from different backgrounds contribute multiple ways of knowing, understanding and resolving problems that arise in the innovation process. From a socio-technical perspective, I analyse how EMACSen<sup>1</sup> are shaped by diverse actors, and at the same time also shape these actors and their practices. This case study aims to track the historical importance

of a prolonged and prominent FLOSS project as well as analyse its innovation process from a socio-technical perspective.

Chapter 6 continues to analyse the socio-technical construction of GNU/Linux in open source software (OSS) through the co-production of the public and the private sectors, that is, the community and the corporation. 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 do have competent skills to understand programming and working in a public domain; 2) corporations. The commercialisation of OSS denotes a hybrid innovation model, which takes advantage of acquiring resources both from the community and the corporate world. The community offers space for experimental projects and informal communications, while the corporation stabilises and standardises the development of these community projects by integrating them and putting them into markets. Unlike working in an informal innovation setting 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. Rather, previous (informal) cooperation on parallel community projects remains of significance in these firm-based developers' daily practices.

In chapter 7, I focus on the local and tacit knowledge whereby different interests and definitions of problems are articulated to form and to shape the process of software innovation. Whereas FLOSS innovation has been emerging as a global spread, the knowledge network is built on a variety of local events and tacit intelligences anchored in the widely adopted open source practices (OSPs). In the deployment of these practices, the FLOSS social world and its knowledge network expand and enrol more actors. In this chapter, locality in the FLOSS social world is explored in terms of local performances of accumulating and producing knowledge in response to a global software problem concerning usability. The

local performance, found in local amateur groups, Linux User Groups (LUGs), serves as an ideal social niche to observe how users translate their interests and perceptions in a form of asking and answering questions, and create collective learning environments on- and off-line. I take LUGs as an example to illustrate how mutual help among local Linux users forms an alternative knowledge network, which connects with the global knowledge network, and facilitates a wider community-based innovation. York LUG (YLUG), where my fieldwork for this part of the thesis was done, will be drawn on to examine how the body of expert knowledge is translated into a local system and how the local expertise is codified as connoisseur knowledge. In analysing the members' everyday languages and interactions, one can also understand how expertise is presented and represented in a *glocalised* context, and subsequently shapes and reshapes the identity of the knowledge holders. Mutual-help and community-based socio-technical (-instrumental) support challenge conventional professional and the industry-led expertise. Expert knowledge is contested by lay knowledge. Hence, the boundary of the professional is both reproduced yet redrawn.

In the concluding chapter 8, I argue the importance of contextualising studies on the relationships and interactions between users and ICTs in everyday practice. I also argue that to develop sustainable software in the future, the model of community-based innovation appears to be a more appropriate and effective approach. I suggest that a community of practice(s), notably seen in the FLOSS social world, acts as a strategic innovation space engaging with diverse perspectives and experiences, providing a platform to generate high quality, innovative ideas. The diverse experiences and information shared freely among members symbolise tacit knowledge anchored in daily experience at a lay level that could be preserved as a source of innovation. To engender such a creative space for designing sustainable products and services for future ICTs, the concept of community participation could be strengthened to leverage the deployment of more heterogeneous and contingent elements in cosmopolitan innovation systems. The development of FLOSS appears to be a compelling case demonstrating the fact that the innovation processes found therein act as catalysts to stimulate new thinking and viewpoints, both at the local and the global levels. Whilst everyday FLOSS activities are worth exploring to learn about the interactions between human actors (e.g. users, developers) and non-human actants (e.g. software, source

## Chapter 1

---

code, hardware), I suggest that the ways in which production and consumption (in a broad sense, not merely in terms of economic purchase) in ICTs should be examined as well in future research, particularly how ICTs are shaped by identities such as gender, sexuality, race and ethnicity, and embodiment. Additionally, the roles, rights and responsibilities of software users and developers established in providing ICTs, in both public and private domains, are also worth exploring.

## Chapter 2 Innovation and Culture

### 2.0 Introduction

This chapter provides a general overview of what is known as ‘innovation’ from a series of perspectives. In doing so, it challenges the mainstream thinking on technological innovation and proposes to view innovation as a socio-technical process from a sociological perspective, particularly via the account of STS. Furthermore, the chapter examines various social issues concerned in the development of ICTs. Given these discussions, software innovation is considered as a heterogeneous and contingent process within a socio-technical context, engaging diverse actors including users, developers and vendors. Taking the im/material nature of software into account, software engineering illustrates a different innovative approach in comparison with other technologies that contain tangible items. Followed by a brief history of the development of FLOSS, this chapter then highlights some current areas of investigation in the development of FLOSS and pulls together some key themes concerning FLOSS innovation. To explore the origin of the socio-technical dynamics of the FLOSS innovation process, the research focuses on a central concept that of the ‘hacker’ and its related practices, which are articulated, interpreted, and rendered differently across spatial and temporal boundaries. Unlike previous research that usually presupposes a fixed hacker subculture or stigmatises hackers as what we might call digital deviants, this research conceptualises hackers and hacker culture as a much more heterogeneous set of practices found both within and at the boundaries of FLOSS itself. My framework analyses the socio-technical dynamics in the FLOSS innovation system through observing how ‘hacking’ exhibits diverse articulations, interpretations and renderings among the social actors involved. At the end of this chapter, therefore, FLOSS, a cluster of software generated through an unconventional innovative approach, is considered as a possible route through which to rethink current ICT development. More discussion of the FLOSS development is provided in subsequent chapters.

## 2.1 The Epistemology of Innovation

An innovation is an idea, practice, or object perceived as new by an individual.

(Rogers & Shoemaker 1971: 19)

Innovation is about something new, but it's hard to find a universal argument about what innovation actually is. Indeed, it has been suggested that there is no need to define what exactly innovation is nor to consolidate the uses of different terms such as 'invention', 'innovation', 'creation', 'discovery' and 'design'. It makes little sense to have any semantic argument because an innovation is "not 'objectively' new as measured by the lapse of time since its first use or discovery. It is the perceived or subjective newness of the idea for the individual that determines his/her reaction to it. If the idea seems new to the individual, it is an innovation." (ibid.)

On the other hand, one finds a contrasting position that appears to see invention as easily characterised. An old saying 'Necessity is the mother of invention' illustrates a common view that human needs are the main driving forces to innovation. This view is given a more formal expression in a variety of engineering textbooks. For instance, the Accreditation Board for Engineering and Technology, Inc. (ABET<sup>ii</sup>) provides the following definition of engineering as a profession:

Engineering is the profession in which a knowledge of the mathematical and natural sciences, gained by study, experience, and practice, is applied with judgment to develop ways to utilize, economically, the materials and forces of nature for the benefit of mankind.

(Volland 1999: 2)

Consequently, engineers are taught that design is to achieve the goal based on the assessment of users' requirements. Indeed, this explanation based on necessity

drives one ‘to consider the social needs and human wants that help formulate the problems toward which inventors direct their attention.’ (Kelly *et al.* 1986: 18). The design process, therefore, is often depicted as a causal loop starting from needs assessment, problem formulation, abstraction and synthesis, analysis, and ending with implementation (Volland 1999). Innovation is also defined according to the degree of change, some referring to ‘radical innovation’ which is different from normal design work. In contrast, another view of innovation focuses on incremental innovation (Rothwell 1992). Elsewhere, Clark and Staunton (1989) note the linear scheme of innovation (that is, invention -->commercial innovation-->diffusion), mostly favoured by economists of technical change, often omits the design state, gives limited attention to implementation, and decontextualises the consequences.’ (ibid.: 13). This restrictive and simplistic thinking of innovation, found before the late 1980s fails to address the social dynamics and complexities of the innovation process, such that a few scholars have proposed to rethink orthodox mainstream innovation theories (Fleck 1988; Kelly *et al.* 1986). Such writers also suggest that innovation is not an infrequent and dramatic event; innovation is often mundane and always socially constructed and embodied/reified in products, production processes, market places, corporate expertise and other areas in the search for simple solutions to everyday problems (Clark & Staunton 1989).

Apart from arguments that adopt a broadly linear account of historical change in the context of a technological determinism, some innovation studies that seem to be more sophisticated, still have their limitations. For example, the notion of “technological paradigms” developed by Dosi (1982), while highlighting the social and economic shaping of technological development, is ‘limited to its philosophical features, makes little reference to, and even less use of, those features immanent in the structure and nature of knowledge on account of the associated social characteristics of paradigms.’ (Fleck 1988: 8). In addition, the dynamics of innovation need to be explored at both micro and macro levels. Individual ingenuity, such as in the example of Edison and his electric bulb, and institutional contributions of organisational know-how, such as capital, production and marketing capabilities, etc., all occur in technological innovation systems. To tackle the common viewpoint amongst management that innovation is ‘a leap ahead of rivals which is then followed by stability’, Clark and Staunton call for

attention on 'exnovation' (Clark and Stuarton 1989: 12). Their view resonates with Schumpeter's (1939) notion of 'creative destruction'. Arguing that innovation and growth leads to the replacement of obsolete products, processes, and firms by more up-to-date and superior successors; the Schumpeterian explanation suggests that innovation does not always lead to closure and stability.

The socio-cultural dimensions of technologies have been central to the analyses of innovation found within the field of STS. Problems about a static hypothesis, objectification, ignorance of the involvement of diverse actors in the innovation process, de-contextualised innovation, neglect of the tacit knowledge in innovation systems, all have been addressed in various STS discourses. A non-determined, multidirectional process that involves constant negotiation and re-negotiation among and between groups shaping a technology has been put forward. A more appropriate innovation model of STS understands innovation as a non-linear process, and it shows how the social environment shapes technical systems. The social groups that constitute the social environment play a critical role in defining and solving the problems that arise during the development of new technologies (see eg. Latour & Woolgar 1979; MacKenzie & Wajcman 1985; Pinch & Bijker 1989). Since different social groups give different meanings to technology and its problems, and also define how the problems of technological development are resolved differently, there is flexibility in the way things are designed; things could have always been different. Building on this conception, everything invented contains various social designations and interests. But because the ideas and consensus turn to closure and stabilization through compromise between interested groups, the artefact that is taken for granted now hides these earlier divergences. It has been suggested that the hidden agenda in innovation might be explored through moving away from the verified practice in engineer's routine work to focus instead on what engineers actually do (Star 1991; Pickering 1992). It is only by opening up our analysis to hidden contingencies that we can open up discussion of innovation as a socio-technical and uncertain practice.

Meanwhile, scholars in evolutionary economics also challenge the traditional linear innovation model. For example, Von Hippel (1988) proposes a user-led innovation model, which is claimed to be more efficient than manufacturer-led

innovation in certain cases. Similarly, Fleck (1988) suggests a new innovation process named ‘innofusion’ to describe the dynamic innovation process in ‘configurational technologies’. As Fleck defines it, ‘configurational technologies are particularly subject to influence by contingencies, and particularly dependent for their development upon the role played by users. The structure of knowledge associated with technological innovation is examined to identify the role of different agents in the technological innovation process. In these terms, innofusion can be characterised as an experimental learning process which crucially involves a range of agents across an industrial sector, and across several organisations. Consequently, politics aimed at encouraging industry sector learning effects may be the most appropriate for facilitating innofusion.’ (Fleck 1988: 1). Likewise, both Von Hippel and Fleck have noticed the important role that users play in the innovation process. Certain technologies, such as computers, robotics and software, make the innovation process more configurable and contingent than others. The character of the technologies enables their users to take a greater part in the innovation system, to empower the users and also to facilitate the development of the technologies themselves. The innovation process is dependent ‘upon the uncertain unfolding of contingencies’ (Fleck 1988: 10). The idea of ‘innofusion’ stresses the importance of the learning process in the innovation system, “a process of learning by experience—the learning by doing hypothesis, that technological innovations originate in accumulated experience of a practical nature” (Sahal D. 1981; Fleck 1988: 8). It also highlights the elements of user experience in the fashion of user-centred design (UCD).

Schumpeterian (1939) notions of creative destruction, and the feature of diverse actors mentioned in Von Hippel's (1988) or Fleck's (1988) approaches, both have inspired new ideas towards innovation. However, what they deal with is still monogenetic patterns. As often criticised, Schumpeter's one-sided interpretation of the source and driving force of self-transformation fails to acknowledge other important facets of economic evolution, such as the roles of individual entrepreneurs and consumers in innovation processes (Witt 2002). On the other hand, ‘users’, though emphasised in Von Hippel’s account, appears to be another variable employed to replace other existing variables in econometric equations. While the variable "users" is treated as an alternative source of innovation, the role played by users serves simply as a substitution for the position that orthodox

designers/manufacturers occupy. His focus on users follows “the tendency in management literature to equate ‘the user’ with a whole, undifferentiated organisation” and fails to acknowledge “the variety of user groups implicated in change” (McLaughlin *et al.* 1999: 22). The changing identity and practice of users are overlooked as is the embedded cultures shaped by them. He pays no attention to the shifting dimension in the innovation process. The instrumental measurement of innovation variables limits the possibility of identifying and exposing the shifting characters of the same. Though the term ‘ecosystem’ or ‘bionetwork’ is mentioned by Fleck (1991, 1992, 1995), he observes that development is a thoroughgoing evolutionary process, in which ecological contingencies are built in the process of ‘innofusion’. While the term ‘innofusion’ is used to show how hybrid and complex the innovation process is, the story that Fleck tells, nevertheless, fails to register the informal interactions between actors, which will be discussed below. Moreover, hybridity and complexity is widely observed in innovation systems and should not be restricted to certain types of technologies, as observed in the multi-dimensional development of the ‘Penny Farthing’ bicycle by Pinch and Bijker (1989). While I suggest Fleck’s argument based on his observation of ‘configurational technologies’ can be applied to studying a wider genre of technologies, such a wide-reaching argument should still allow one to consider the commonalities of as well as differences between different technologies and their innovation processes.

In the FLOSS innovation arena, one sees a broad and relatively boundless innovation system that allows multiple actors to participate and engage in its development. The innovation process cannot be clearly identified phase by phase; it is not a state but an active socio-technical process. FLOSS innovation happens in multiple dimensions including the economic, political, social and technical. Such an innovation based on the incentive of a community with collective practices or interests cannot be understood solely from one single perspective in innovation studies. Multiple contexts must be explored to ensure that a thorough picture of FLOSS innovation can be drawn. This is not to say that the technological innovation process is to be excluded when looking at the social context. Instead, the materialised practices of the technological innovation process are key elements in this framework. The collective technical practices found among actors from various social worlds thus are central to our understanding of how the boundary

objects that various actors engage with are actively ‘crafted’ in a process of ‘mutual enrolment’. Such technical practices embed various conceptions and social meanings found within the FLOSS social world.

Hence, the social-worlds analytical framework, is employed in this research to allow the examination of the interactive and temporal stabilisation of a plethora of cultural elements, while enriching our understanding of that process in its focus on the new patterns of intersection and circulation that come into sight when one recognises the social heterogeneity of practices (Fujimura 1992; Pickering 1992). Unlike some evolutionary studies explaining innovation in an oversimplified, selective social Darwinist manner, the social-worlds framework concentrates on ‘the interplay between technological systems and organisational dynamics and between internal settings and external contexts’ (McLaughlin *et al.* 1999: 17).

## **2.2 Cultural Dynamics in Innovation Systems: Social and Organisational Change**

As noted above, innovation is a social process involves on-going communication, compromise, negotiation and decision-making. An innovation system can be interpreted as ‘a social system that is a collectivity of units which are functionally differentiated and engaged in joint problem solving with respect to a common goal’ (Rogers and Shoemaker 1971: 28). Within such a system, one should ‘seek insight into the messy, long-drawn-out embedding of technology into the specifics of organisations.’ (McLaughlin *et al.* 1999: 23). In other words, it is ‘a long-running process by which technology is embedded into the local setting of the organisation’ in the diffusion of innovation (*ibid.*: 39). Innovation thus should be characterised as ‘contingent, local, unpredictable and as taking place over an extended period of time’ (*ibid.*: 20).

A range of research in the field of STS has dealt with this question over time. From Merton (1957, 1961, 1963, 1971, 1972, 1973) who argues that scientists, their achievements, and the accorded norms are shaped both institutionally and informally, to Kuhn (1970) who contends that mainstream science is produced in structural and methodical ways under stable paradigms, and to recent analysis

anchored in conversation analysis and ethnographic methodology to understand how actors and actants interact in processes of knowledge production (see e.g. Garfinkel 1967; Latour and Woolgar 1979; Knorr-Cetina 1981; Star 1983; Lynch and Woolgar 1988; Star and Griesemer 1989; Clarke 1991, 1997; Fujimura 1991, 1992; Lynch 1993), STS has primarily focused on episodes found in formal settings such as laboratories, offices or classrooms. Although situated practices, local knowledge, networks, translations and controversies are highlighted in later works such as Callon (1986, 1992) and Latour's (1987, 1992) Actor-Network Theory, the interactions found in informal settings that also shape the constitution of knowledge have not yet been paid much attention. In cultural terms, however, an informal and collaborative networking is central to the mobilisation of innovation. This point is particularly important for our understanding of innovations in contemporary society that are heavily dependent on and upheld by ICTs, especially given the prospect that computer-mediated communication (CMC), data-archiving and resource sharing is regarded as key to greater innovative efficiency (Hine 2002). An informal and collaborative peer culture seems to be of great value in innovation when seen as part of a wider cultural practice. As Brian notes, 'collective production of skills and practices enable social actors to make sense of their lives, articulate an identity, and resist with creative energy the apparent dictates of structural conditions they nonetheless reproduce' (Brian 1994: 192). In Brian's account, 'the social construction of cultural artifacts entails the production of practices which, in turn, enact their own status in broader social contexts by inscribing both the boundaries of cultural domains and the social status of the author in qualities of the artifact' (ibid.: 193). This is also what Callon (1987) terms 'society in the making'. "The duality of cultural production", in Brian's observation, "is reflected in two orienting questions: First, what is the practical logic imposed on techniques and strategies of cultural production by their implication in the reproduction of the relatively autonomous domains of culture in which they operate? Second, in what ways are both the practical constitution and the sociological significance of particular cultural objects rooted in what will be referred to as the 'art' of artifacts?" (op.cit). This issue links to the concern of organisation and management amongst individuals and diverse agents. Brian (1994) 'proposes a sociology of culture that looks at the practical construction of socially positioned agency, and at the way particular forms of agency are objectified in domains of artifacts. The modalities of objects refer to procedures

and practices that inscribe agency in artifacts, give objective status to agents, and enact the boundaries of the domains in which they are empowered to act, in which the available technologies of production operate.’ (op. cit: 217). He notes that ‘the social construction of artifacts is at the same time the materialization of a practice that enables particular kinds of agents to intervene productively in the world of things; as cultural or technical artifacts are stabilized, they stabilize the field of operations in which they are produced, the practices that produce them, and the social relations implied in both their production and use.’ (ibid.: 193). Unlike others who also underline the interpretative flexibility in the production process of knowledge and science (see e.g. Collins & Pinch 1993, 1998), Brian's focus on cultural factors does not mean that he presumes there is a tendency towards closure and stability. As McLaughlin *et al.* (1999) say, ‘Behind the appearance of stability, [organisations] are in a constant state of flux.’ (page 24). Indeed, ‘developments that appear outside the organisation are a major source of instability and uncertainty, [but] equally, inside the organisation there are everyday contingencies that can never be entirely erased.’ (ibid.). Though organisations endeavour to pursue stability or certainty, ‘attempts to manage uncertainty are in continuing struggle with indeterminacy and the stubbornness of the particular and the local.’ (ibid.). ‘In addition, different interests are at play, across an often diverse and dispersed organisational space, which contest and confound management objectives.’ (ibid.). Given the fact above, the framework based on “a new kind of ‘postmodern’ organisation that embraces ambiguity, contradiction and flexibility” (ibid.) is preferred to catch the dynamic organisational change in innovation systems whereas standardisation, stability and closure of technologies are seen in a broader context.

Given the arguments above, this thesis centres on collective and situated practices and narratives in innovative environments, both formal and informal, to strengthen the socio-technical commonalities and uncommonalities in innovation systems. To paraphrase McLaughlin *et al.* (ibid.), practices and narratives operate mutually to organise actors and actants in innovation systems. Actors in organisations conform and accommodate their collective identities through continually translating and reifying actants into their lives, meanings and settings. Cultures, as shared beliefs, assumptions and ways of life emerging in organisations, are not necessarily designable and manipulable to particular

interests, but should be considered as essentially differentiated and fragmented across the organisation. That said, investigating innovation practices from a sociological perspective helps understand activities, judgements and interpretations in different social groups and answer ‘how’ over ‘why’ questions in the construction of science and technology. This account leads the discussion further to the localisation of innovative practice and knowledge where a generalised set of innovative assumptions are enacted and exploited differently in particular local settings. As McLaughlin *et al.* argue, in innovation systems, therefore, ‘technology acquisition is, in one sense, about how the organisation is defined in relation to shifting and multiple local understandings of the wider environment.’ (ibid.: 30). In other words, ‘techno-organisational change must be situated within broader developments without reducing it to a simple expression of those developments.’ (ibid.). This process, as I will argue, characterises much of the FLOSS innovation system.

### 2.3 Dynamics in Software Innovation

Software is not just made up of computer programs but also all associated documentation and configuration data which is needed to make these programs operate correctly. A software system usually consists of a number of separate programs, configuration files which are used to set up these programs, system documentation which describes the structure of the system and user documentation which explains how to use the system and for software products, web sites for users to download recent product information. ... In software engineering, the development principally follows this process.

(Sommerville 2001: 5)

Software development, like all technological development, is a social process. It is not simply *shaped by* society: as the outcome of work, software development is part of society, *part of* its more general social development.

(Peláez 1988: 1)

Software is made of symbolic programming language. It is precisely its nature as *language* that makes the innovation process here differ from that of tangible objects, and the socio-technical complexity and hybridity of software innovation are much greater (and unstable). This feature means that software technology is particularly mobile and mutable compared with others. But such mutability also leads to a situation where there is considerable debate over the imperfection of software languages: vulnerabilities are easily found; incompatibilities always exist somewhere with different machines. As a result, uncertainty about software production and its implementation emerge. Meanwhile, under the conditions of uncertainty, there are opportunities that can be exploited in the shaping of software production. From a socio-technical point of view, a variety of cultures are embedded and exhibited in the process of software innovation, making it more dynamic. This section will introduce two main streams of the issues concerning the heterogeneity and contingency of software technologies, namely technical and social-economic ones, in the software innovation process.

### **2.3.1 Technical Issues in Software Innovation: Modelling Software Process**

Computer scientists have acknowledged that without using the software for a long period, it is difficult to assess software attributes with a range of criteria for software quality: maintainability, dependability, efficiency, and usability (Sommerville 2001). To facilitate the technological innovation process, several software process models<sup>iii</sup> have been proposed within computer science to standardise and effectuate the software production, such as the “waterfall model” (Royce, 1970) or the “spiral model” (Boehm 1986, 1988). The waterfall model, derived from other engineering processes, was the first published model of the software development process. Because of the cascade effect from one phase to another, this model is known as the ‘waterfall model’ or “software life cycle”<sup>iv</sup>. The principal stages of the model map onto fundamental development activities from requirements analysis and definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. Though “the spiral model” has tried to advance the waterfall model,

it fails to encompass the complexity and dynamics of software production. Consequently, in recent years, a series of so-called “lightweight” methodologies such as “extreme programming” (XP)<sup>v</sup> have been suggested to lessen the requirements of following many rules, practices and documents in developing software. These new methodologies actually reflect and underline the socio-technical complexities in the software process. Furthermore, changing the process does not always lead to improved products to meet users' real needs. Faced with these difficulties, software innovation has been regarded as a problem related to the skills of computer programmers. Studies in the late 1970s and 1980s have suggested that the shortage of skilled programmers and tensions between programmers and managers are the main obstacles to effective software innovation. (Peláez 1988). However, given the attributes of software, one should not ‘understand its development by relating it solely to the labour market for programmers or the conflict between a particular group of workers and their managers. Nor can we understand software development, as many computer scientists suggest, in terms of an inner logic, in terms of its own dynamics. This is unsatisfactory because software is a social process, being produced by people in society. This is a society based on conflict, but the conflicts cannot be reduced to the clash between programmers and managers.’ (Peláez 1988: 2)

### **2.3.2 Socio-economic Issues in Software Innovation**

Apart from technological uncertainties, therefore, social issues are symmetrically crucial in software innovation. In light of his case study of the electronic funds transfer (EFT) systems, Kling (1987) recognises the conflicts between designers, managers and wider social actors. The negotiation process between these actors means that a technological innovation faces both social and technical challenges, internally and externally. New technologies, in the external context, confront different social values and current everyday practices. The merits of change are always open to challenge: as Kling observes,

Many new technologies exacerbate conflicts of interests, and conflict alone should not discourage innovation. Groups that can exploit technical innovations often fare better than competitive groups which

do not. ... However, value conflicts are more subtle. It may be easy to say that groups which develop market-supported innovations best serve the public interest. But it is harder to assert that public life is improved, say, if legitimate opportunities for freedom of individual expression are sacrificed for administrative efficiency.

(Kling 1987: 51)

Moreover, the local implementation of new practice and knowledge serves specific interests. In Edwards's (1987, 2001) investigation of British banks, computers served as strategies for organizational change and restructured banking labour.

Computers facilitated, for example, progressive specialization of tasks and automation of a great deal of work once done by hand. ... Along with this specialization went a deliberate restructuring of career paths.... [M]ore specialized job mean greater expertise but less flexibility [allowing labours to shift from task to task during the banking day]. ... This centralization reinforces the segmentation of banking work and creates a class of specialist managers. The outcome of this computer-based restructuring of bank organization was mixed. While productivity in such repetitive tasks as data entry rose, numbers of clerical staff did not decline and frequently rose. A new gender division of labor also emerged, with more women working in the low-ceiling role of clerks and men clustering in what was known as the 'accelerated career program'.

(Edwards 2001: 277-278)

### **2.3.3 Prescriptions for Studying the Dynamics in Software Innovation**

Different local effects have been apparent in the computerisation of the global banking industry. In each local implementation, there are some socio-technical aspects that are not foreseen nor desired by the designers. These examples not only serves to critique notions of 'progress' based on scientists' and technicians'

dominant presumption of 'efficiency' and 'improvement' (in terms of state-of-the-art), but also strengthen the case for accepting the never-neutral and socio-technical construction of technologies. That said, “experts” views in regard to technological innovations are often uncritically accepted and unproblematically endorsed by policy-makers and opinion-shapers. Products designed for users sometimes lose their focuses in the baroque technical jargons or marketing languages. Whilst users requirements are emphasised emphatically in the literatures of engineering design (and so in software engineering), designers’ individual concerns sometimes override such basic purposes. To assess users requirements in software innovation, software engineers begin with software specification, and then follow up with software development, software validation and software evolution respectively. It is generally acknowledged that a good programmer must have problem-solving skills to be able to identify and define the problem to be solved, develop alternative design solutions and implement the solution finally selected. (This issue of problem-solving is highlighted in my empirical research and will be discussed in later chapters). A common view of the conflict of interest between the computer (or software) producer and the end user is noted by Paláez:

It is in the interest of the producer selling the product to inflate the image of what it can achieve, and with it the expectations of the user, to expand the user’s ‘requirements’ as much as possible. When these expectations are not fulfilled, the users claim that the producers have duped them, the producers claim that the users’ expectations were unrealistic. Often the software experts feel that they are caught in the crossfire between the producers who have ‘oversold’ and the user who has ‘overbought’, with both sides blaming the programmers for their inability to make the machine perform miracles.

(Peláez 1988: 3)

Another common example of a disjunction between design and use happened between data processing staff and the company’s requirements: ‘instead of defining the needs in terms of profit and growth, data processing staffs tend to want to try out the latest technology available’ (Peláez 1988: 3). To return to

McLaughlin *et al.*'s argument on the dynamics of organisational needs, “talking of ‘organisational needs’ hides the variety of different needs voiced by individuals and groups within the organisation. [Moreover,] needs are formed and reformed over time as an integral part of the process of innovation.” (McLaughlin *et al.* 1999: 22). In their terms, ‘the value of a new technology in an organisational setting is often uncertain, unstable and contested.’ (ibid.: 23). Therefore, not every user need has an equal chance of being met in conventional ‘needs-driven’ innovation.

In addition to the issue of user requirements, software engineering also faces the heterogeneity challenge of operating as distributed systems across networks that include different types of computer and with different kinds of support systems (Sommerville 2001: 13). Since software is a set of instructions detailing the operations to be performed by the computer, inevitably, the design of the hardware shapes the development of the software (Peláez 1988: 2). The heterogeneity challenge, in other words, is the challenge of developing techniques to build dependable software which is flexible enough to cope with this heterogeneity. Issues of portability and usability will be explored in the analysis of the primary material discussed later in this thesis.

Given the arguments above, software programmers are aware that their work is affected by social factors and human errors, but again, they take the conflicts as part of the problem-solving routine. In response to the user requirement issue, there are two methods generally employed in the industry: to introduce social scientists to participate in the software process is one method, as in the ESRC’s PACCIT (People @ the Centre of Communication and Information Technologies) programme, and to perceive user requirements through various surveys is the other. The former is popularly accepted in human computer interaction (HCI)<sup>vi</sup> and computer-supported cooperative work (CSCW) and the latter is more often used in commercial software production. More recently, participatory design (PD) is brought to the software design and development process (Nyce & Bader 2002).

Although the taken-for-granted engineering view on ‘problem’ and ‘solution’ has been challenged, a simplified hierarchy of the goals of software development,

namely writing program code and making sure the program actually works, still exists (Kaptelinin 2002: 57). For example, in response to their dependency on hardware, and as a way of coping with (by reducing) heterogeneity, software corporations have worked with the hardware corporations to form a stronger technological regime. Following up IBM, the first leading company that produced computer systems in the computer industry, Apple and Compaq as PC specialists, Intel and Microsoft Windows fostered a new paradigm of 'Wintelism' in the computer industry (Borus and Zysman 1997; Nohara and Verdier 2001).

"Wintelism led to the decline of "proprietary systems" and placed designers of "operating systems" and microprocessors in a key position, to the detriment of computer manufacturers, "Big Blue" notably, but Compaq and Toshiba as well." (Nohara and Verdier 2001: 201). The advent of the Internet renders the software industry even stronger. Producers of applications (SAP, Adobe), of interfaces (Netscape) and of languages (Sun) and other 'pure product definition companies' such as Cisco and 3COM, as Nohara and Verdier name, gained impetus in the computer industry. Nohara and Verdier argue that 'the result was a move away from proprietary systems towards open systems, which ensured compatibility between the standards of the various suppliers whose products and services provide the foundations on which the networks depend' (ibid.). Standardisation among various software systems is developing. 'Control of these standards and of the associated intellectual property rights are essential resources for those seeking to obtain competitive advantages in these new markets in the IT industry.' (ibid.). Given these features, it seems that the software industry is following a typical 'technological trajectory' (Peláez 1988: 6).

However, as I will critique Peláez, it is not entirely wise to consider the evolving software industry as having a natural technological trajectory. In fact, as Nohara and Verdier summarise, it has "a certain dynamic irreversibility contained within 'particular institutional infrastructures'." (Dosi et al. 1988; op. cit.: 202). At the micro level of the software innovation, the linear problem-solution thinking is dominated in the industry; at the macro level, strong mission-oriented innovation policies are favoured among corporations. Technological innovation is a social process of cultural construction in action, which is the mainstream engineering view fails to see. After a product is innovated, there are still ongoing struggles with social-cultural factors and reinvention is possible in local settings. 'Social

contexts and design interpenetrate; no element is purely essential and no others purely accidental.’ (Edwards 1987: 284). In this respect, software innovation should be perceived as ‘a social process through which computer technology and the social production of knowledge and values interact.’ (ibid.). In this regard, I suggest that looking at the FLOSS development where software products are shared among different communities of practice. In so doing, it will help contextualise and crystallise the socio-technical dynamics in software innovation.

### 2.4 A Brief History of FLOSS

Open source is not a new way of doing things—it is the original computer way of doing things.

(Rosenberg 2000: 3)

As the quote above implies, it is difficult to define the exact start date of FLOSS. Though conventionally the origin of FLOSS is linked with the early Unix source distributions (Raymond 2004), the practices of peer-review and sharing source code started before the beginning of Unix. While the computer industry was still dominated by mainframe computers that were capable of undertaking large-scale computational tasks, software was regarded as less valuable than machines. It is documented that in the early days of computing (i.e. the 1960s and early 1970s), hardware manufacturers gave away their software because their machines would not operate without it. Users paid to cover the cost of copying a programme and could then make their own copies for free. Few thought about copyrighting software, let alone business practices. Software was a give-away needed to sell the actual article of commerce—the expensive hardware. 'Because none of the software would run on a competitor's machine, no one gave "software piracy" a thought, let alone a name.' (op cit.: 4). Richard Stallman, the founder of the Free Software Foundation (FSF) and the initiator of the GNU (GNU's Not Unix) General Public License<sup>viii</sup> (GPL), describes how he and his group worked in the MIT AI Lab in the 1970s. In his view, the practice of peer-review, mutual-help and the sharing of source code represented an environment which was free of central authority. This feature facilitated the development of many software programmes,

a set of open standards and most importantly of a portable operating system, UNIX. The main feature of the UNIX system is its uniform toolkit that can work across platforms as well as be combined flexibly for further use. The development of UNIX not only corresponded with the development of many modern Web technologies, but also advanced FLOSS. FLOSS operating systems such as BSD and Linux are based on UNIX with a strong modular feature to their architectures. As has been noted, "the manner in which different individuals can take responsibility for different self-contained modules within Linux is acknowledged as being a major factor in its successful evolution" (Feller & Fitzgerald 2002: 170).

With the development of the minicomputer, software gained greater importance in computing and source codes were put under protection of a non-disclosure agreement (NDA), and only binary code, readable by machines but not by humans, was released to be available. 'Hardware manufacturers, who had originally seen software only as something that had to be supplied to sell machines, began to realize that not only could the software that came with the machine benefit the customer, it could also tie the customer to the machine.' (Rosenberg 2000: 9). Accordingly, in the early 1980s, many programmers at MIT AI lab left to work in corporations, and the MIT AI lab's open system was replaced with a closed, proprietary system that was maintained by the vendor. The source code of UNIX was also locked up and sold to large corporations for large sums that students could not afford to pay (ibid.: 10). Under strict licensing control by the majority of commercial firms, closed code is more of a secret language. Since it is shared, if at all, in binary form designed for interpretation by the computer, it will not be easily interpreted—let alone modified—by other programmers. Without source code, hacking would be impossible (ibid.).

In response to the rise of proprietary software, programmers started to clone UNIX to serve their technical interests. The University of California at Berkeley began distributing a version of UNIX, called BSD (Berkeley Software Distribution). 'BSD at times surpassed Bell's UNIX, and its freely modifiable and distributable code made its way into numerous commercial versions of UNIX', e.g. Sun Microsystem's operating system (ibid.: 10). As BSD began to establish itself,

AT&T started a lawsuit against BSD. Under legal pressure from AT&T, the BSD developers dropped every line of Bell Lab's code and substituted with their own. The BSD system counters the view that FLOSS always lags behind and imitates proprietary software, and suggests that innovations and improvements have originated from within BSD and FLOSS (ibid.: 11). Meanwhile, Andrew Tanenbaum, a professor at the Free University of Amsterdam, didn't agree with closed source code either. 'From an educational viewpoint, the ban on discussing Unix's source code was unsatisfactory.' (Moody 2002: 33). In 1979, ten years after Unix was first created, Version 7 of the system was released by AT&T. 'Version 7 represented the symbolic closing of Unix inside the black box of proprietary software—a sad end to what had long been the ultimate student hacker's system.' (ibid.) Therefore, Tanenbaum originated Minix, a clone of UNIX, and distributed it in 1987 as an illustrative operating system with his course book '*Operating System: Design and Implementation*' for students. He created a new Usenet newsgroup to go with the software and that newsgroup attracted at least 40,000 people within a month, including the originator of Linux, the Finnish student Linus Torvalds.

About the same time when Tanenbaum started his Minix project, Stallman, the veteran free software activist and developer, had founded the FSF and the GNU project. The GNU project was created to mimic the Unix system, and could be shared freely. Stallman started GNU with EMACS (an acronym for EditingMACroS), followed by GCC (GNU C Compiler). Stallman supported himself by charging \$150 for actually making a copy. After releasing the programmes, he asked that all changes and improvements be sent back to him. In this way, he evolved the model of the GPL, which allows free use, modification, and distribution of the software and any changes to it, restricted only by the stipulation that those who received the software relay it to others with the identical freedoms to obtain the source code, modify it, and redistribute it. In other words, the products of a developer's skill had to remain free for all developers to use and reuse. 'Free' did not mean that no money changes hands, but it did mean that no authority or non-disclosure agreement could prevent developers from sharing code. "Because of the freedoms conferred by the Free Software licensing method, and because the legal means of its enforcement is the copyrighting of the software itself, the use of the GNU GPL is often called 'copylefting'." (Rosenberg 2000:

11).

By 1990, Stallman's GNU project, including the 'C library'<sup>viii</sup> and 'Bash'<sup>ix</sup> along with many other elements of the Unix operating system, was almost complete. Still missing, however, was the kernel<sup>x</sup> programme. 'Although many other programs are indispensable for a fully functional operating system—for example a C library and a shell—it is the kernel that defines its essence.' (ibid.). Stallman even approached Tanenbaum to provide Minix as the missing kernel of the otherwise complete GNU system. But Tanenbaum considered Stallman to be an abrasive person, and Stallman's idea was not one he fully shared (Moody 2002). This rejection by Tanenbaum, as the later FLOSS history shows, prompted Linus Torvalds to begin to develop the 'Linux' system. As Torvalds observed, 'if the GNU kernel had been ready, [for example, through a form of Minix—when Linus was casting around for his own copy of Unix] I'd not have bothered to even start my [Linux] project.' (Moody 2002: 32).

Inspired by Minix, Linus Torvalds wrote Linux and uploaded the source code to the Internet. Torvalds started the Linux project when he was studying at Helsinki University. In contrast to Tanenbaum's attitude of moderating Minix, Torvalds was more willing to solicit ideas for improvement and welcomed other people's own efforts in this regard. He also had a more pragmatic attitude towards free source code distribution that led him to release Linux under the GNU GPL scheme. Since the GNU *Hurd* kernel program had not yet appeared, Linux was picked up as the GNU kernel program and became a hackers' toy. As Torvalds claimed in one of his early posts to the mailing list, '[Linux] is a nice learning tool, and it was/is fun working on it.' (Moody 2002: 46).

The advantage of FLOSS lies in its capacity to be divided into segments or modules so that programmers can tinker with different parts of the chain without affecting the programme as a whole. Apart from the technical development that BSD, Linux and other FLOSS operating systems reflect, these systems share important socio-legal characteristics, particularly in fostering social networking, creating social capital and challenging the traditional concept of intellectual property (IP). FLOSS licences for example have been an important mechanism

through which innovation is encouraged. Allied to expectations centred on altruism and voluntarism, selection, leadership-building, and reputational reward (Ghosh 1998; Raymond 1999; Rosenberg 2000; DiBona *et al.* 1999) within the FLOSS community, open licences serve as the legal and formal basis upon which FLOSS is built. There are many licences within FLOSS, including the GNU General Public License (GPL), the BSD License, GNU Library General Public License (LGPL), QT Free Edition License, Apache License, Mozilla Public License, X License etc.. Though they all allow source code to be distributed freely, they each provide terms and conditions for copying, distribution and modification that help retain the ‘open’ character of the systems. Take the GPL<sup>xi</sup>, the earliest licence, for example. Though one may copy, modify, and distribute verbatim copies of the program’s source code, one may not appropriate any modification one makes (i.e. modifications must still be distributed under the GPL). Thus, the author of a GPL-ed program is likely to receive improvements from others, including commercial companies who modify his software for their own purposes. (Perens 1999). GPL also does not allow the incorporation of a GPL-ed program into a proprietary program.

However, the open source position accepts there may be circumstances where the conditions may not be fully met. It is worth noting that although copyleft allows people to share works and their improvements, it does not restrict uncooperative people to convert the program into proprietary software. For example, Torvalds decreed that it is permissible to let proprietary modules of software be embedded within the kernel. This again, was based on his pragmatic attitude where, for example, he recognised that some video card companies might want to support Linux but not reveal the inner workings of the software that controls their products, a position against the GPL’s policy that modification cannot be a private asset. Torvalds’ position also can be seen in a recent case regarding DRM (Digital Rights Management) in Linux. DRM tools are technological locks or identification measures that range from ensuring a software program is genuine to protecting a movie against unauthorized copying. Given the strong commitment towards open information in the GPL, what DRM stands for conflicts with the GPL. But as reported in C|NET News.com (April 24 2003), Torvalds outlines a controversial proposal in a posting sent to a key Linux-focused

email list that nothing in the basic rules for the Linux operating system should block developers from using DRM technology. There has always been an argument in the FLOSS community about how far information should be completely open. ‘In some open-source and “free software” circles, such locks and authentication measures are seen as infringements of programmers’ freedom. In his posting, Torvalds took a more pragmatic approach—Linux is an operating system, not a political movement, and people should ultimately be able to do what they want with it, he said.’ (ibid.).

Whereas FLOSS licences thus serve as an instrument to institutionalise the community, there are de facto a range of views about the contested issue of open information. More questions have been proposed with regard to the dual licensing—GPL and proprietary commercial in parallel—phenomenon and the trade-off between FLOSS licences and intellectual property rights (IPR). The former are now in place. As observed in the case of MySQL, GPL is adopted for open source use while licensing separately for clients is used by businesses. ‘[The dual-licensing] is also a possible solution to the public funding / corporate use debate, in that publicly funded or academic software could be released under the GPL for non-commercial use, where modifications must be retained in the “commons”, and released separately under commercial licence for incorporation or modification in proprietary software.’ (Ghosh 2002: 8). As to the second question, the pros-and-cons of IPR and the copyleft-ed licenses, this is still under debate. Some proponents of IPR contend that the IPR can protect the motives of innovation while the others contend that IPR regimes, especially software patents, act as disincentives for open source, and create entry barriers to software innovation. (ibid.: 9). There are surveys available that claim to show which licensing system fosters innovation. Some preliminary surveys have shown that the IPR route may not be the first choice among software developers. While some within the FLOSS social world would support radical action to gain unauthorized access to computing systems or allow duplication of copyrighted items, others remain more conservative except for employing and contributing source code. How these different socio-technical perspectives on FLOSS shape its innovation will be dealt with in later chapters. Here I want to draw attention to the fact that the open-source programming field is actually a highly contested social arena that can create serious tensions in the FLOSS innovation system. How these are resolved might

well shape the direction of FLOSS innovation.

## **2.5 Open Source Definition: Is there ‘the’ FLOSS or just FLOSS?**

As the term FLOSS is widely applied in the software world, one fundamental question with regard to the definition of FLOSS needs to be answered: Is there ‘the’ FLOSS or just FLOSS? What are its boundaries? Can all software developed in a similarly open-source model be deemed to be FLOSS? Or is there any other criterion for defining the FLOSS? Or is there actually no single arena called ‘the’ FLOSS, but a hybrid category of software mixed up with proprietary software and open source code contributed by volunteers? As Gacek *et al.* (2002) argue, a project may be open source but this does not provide a precise description of the approach used to support the project. There are many meanings of the term ‘open source’ given a collection of common characteristics found in various FLOSS projects.

Perens tries to give a definition for the FLOSS. In *The Open Source Definition*<sup>xiii</sup>, he suggests that open source not only means free access to source code, but also the distribution terms of open-source software must comply with the following criteria. Basically speaking, the Open Source refers to a software licence that certifies users have:

- The right to make copies of the programme, and distribute those copies.
- The right to have access to the software’s source code, a necessary preliminary before you can change it.
- The right to make improvements to the programme.

In Perens’ account, this Open Source Definition captures what the great majority of the software community originally meant, and still mean, by the term ‘Open Source’. However, the term has become widely used and its meaning has

lost some precision. In a sociological sense, these notions are interpreted differently by different groups of people. The criteria that mark open source code can be seen as a repertoire and resource mobilised in different ways by different actors working with open source. As Perens notes, a licence as strict and complete as the GPL is, is still unable to prevent FLOSS going proprietary. Even *Debian*, one of the Linux distributions that follows the GPL most closely, cannot avoid its source code to be used in or to use proprietary software. It is difficult to classify clearly then which software is *the* FLOSS. Authors can dual-license their software: one is warranted by one FLOSS licence and the other is patented. But the commercialisation of FLOSS, such as the commercial Linux distributions, might make the commitment to Open Source difficult to maintain. However, this research has no wish to argue whether the commercialisation of FLOSS would lead to its demise or not, but rather explores how the conflicts, negotiations and compromises between FLOSS and proprietary software shape software innovation itself. Therefore, FLOSS innovation is constantly defined and redefined within and between diverse socio-technical groups through their practices within the FLOSS system itself. These groups negotiate innovation through exchange across various boundary objects – such as software patches – as will be explored more fully in later chapters.

### **2.6 Literature Review on FLOSS**

FLOSS has emerged as an important phenomenon in the ICT sector as well as in the wider public domain. A new research strand has attracted scholars and practitioners to analyse the development of FLOSS from many perspectives. While the FLOSS community continues to grow, diverse actors (e.g. developers, firms, end-users, organisations, governments etc. just to name a few) are brought into play. Meanwhile, a variety of apparatuses and inscriptions (e.g. technical ones such as software and hardware tools, socio-economic ones such as licences, educational ones such as certificates, and socio-cultural ones such as on/off line discussion forums) are developed and employed to maintain the practice. The complex composition of the FLOSS community entails a heterogeneous field where innovation is socio-technically constructed. Practices and norms in the FLOSS community are interpreted differently in support of individual demands

(social, economic, political, technical) of the actors. Such a heterogeneous world resembles an ecological system that contains diversity while resources (information, knowledge and tools) are commonly shared amongst actors.

Technically speaking, current research on FLOSS, across academic disciplines and industry fields, mainly focuses on measuring the efficiency of productivity in the context of open source development in terms of code reuse, density of bugs, complexity of code or frequency of release, usage, and adoption in the software engineering approach of productivity cycles. At this point, ‘contribution needs to be measured through several sources—code itself together with the version control and management data that come with the development process, but also the discussion groups, documentation and other processes that go into collaborative authorship’ (Ghosh 2002: 5). A prominent example with regard to determining the benefits of the open source development model is improving security. Given the nature of software as depicted in chapter 2, it is generally agreed that ‘given enough eyeballs, all bugs are shallow’ (Raymond 1999). Moreover, FLOSS also contributes to open standards and interoperability because the availability of source code increases the transparency of software and eases the development of compatible complementary software, even though no formal standard is defined or adhered to in this regard.

Socio-economically speaking, an increasing number of literatures are dedicated to the understanding of the motivation of a FLOSS-like innovation approach. In this area of work, hacker culture is considered as the essential cultural factor upholding the development of FLOSS (Himanen 2001; Raymond 1999). Both the collaborative and competitive characters derived from hacker culture appear in recent sociological and anthropological studies of FLOSS. Whilst the former highlights the features of gift culture, community-forming, information-sharing, and social networking in FLOSS innovation, the latter emphasises the mutual-challenging and self-exploring aspects in a reputation-reward system. However, the views on the hackers in, for example, Himanen’s (2001) and Raymond’s (1999) accounts, repeatedly overstate ‘the hackers’ as a homogeneous group and overlook the diversity in the hacker social world. As MacKenzie (2001) comments on Himanen’s work, ‘Its focus on hacker heroes and their individual ethical

values as the core of hacker culture largely ignores the complicated practices of software development for the sake of what I can only read as an uncritical individualism centred on passion: ‘hackers want to realize their passions.’ (Himanen 2001: 18)” (MacKenzie 2001: 544). In MacKenzie’s view, sociological research on the FLOSS community should go beyond the idealised and self-serving versions of the FLOSS projects towards understanding FLOSS as a practical sociological phenomenon. Though work like Moody’s *Rebel Code* tells the detailed story of the FLOSS movement, his narration comes from a historical perspective rather than a sociological one. MacKenzie thus suggests it is important to analyse material practices as well as social practices ‘to highlight the fact that mechanisms do silence the compromises they embody’ (ibid.: 549). This avenue of thinking corresponds to what has been proposed elsewhere in cultural studies that ‘the social construction of artefacts is at the same time the materialisation of a practice that enables particular kinds of agents to intervene productively in the world of things.’ (Brian 1994: 193). In other words,

If we understand open source software as a material practice of reading and writing distributed across the computer networks, we can also comprehend how these values of openness and freedom offer themselves to commodification. Openness and accessibility as values in themselves resist the dominant norms of property, but they also facilitate the formation of auto-regulating labour force focused on the production of software

(op cit.: 552)

### 2.7 Hackers

A *hacker-driven* innovation has been proposed to denote FLOSS development. It is generally recognised that FLOSS was originated from the hacker culture of the 60s and 70s, when hackers defined themselves as ‘clever software programmers who push the limits of the doable’ (Rosenberg 2000: 6).

Previous research on hackers is piecemeal and limited. The first text

systematically introducing computer hackers appears in 1984 when Levy compiled a detailed chronology of hackers in his book entitled *Hackers: Heroes of the Computer Revolution*. This book revealed an unknown world where technical innovation was developing at a high speed. Levy describes how the activities of hackers influenced and pushed the computer revolution forward. Members of this world tried to mobilise the power of computing in entirely novel ways. They communicated with each other through the Net in a binary-coded language. Because this world was so different from wider social life, its members were regarded with suspicion and often seen as deviant. Levy's book in 1984 decriminalized hackers, as readers learned that the era of hacking had commenced in the 1960s in university computer science departments where highly skilled students worked and shared information on the Net. Not surprisingly, perhaps, during the following decade hackers gained great success in computer businesses such as Apple and Hewlett Packard. It seemed that their business success was so marked that their identity as 'hackers' per se was downplayed. Then there came the software hackers of the early 1980s who created the application, education and entertainment programs for personal computers. Bill Gates' Microsoft was started at this time. With the growth of the internet, the contemporary hacker generation is engaging with new '.net' issues such as licensing, patents, security and privacy, all key to the development of the software industry. In addition to developing software technology, hackers in this generation also have to deal with more social and political issues than before.

Levy classifies hackers into three generations from 1950s to 1980s according to their various actions and beliefs 'associated with hacking's original connotation of playful ingenuity' (Taylor 2000: 36). According to Levy, the earliest hackers, the pioneering computer enthusiasts at MIT's laboratories in the 1950s and 1960s, were the first generation of hackers, who were involved in the development of the earliest computer programming techniques. Then there was the second generation of hackers who were engaged in computer hardware production and the advent of the PC; the third generation of hackers (who were also fanatic game players), devoted their time into writing scripts and programmes for game architecture. Other literature providing an account of the social role of computer hackers is *The Second Self* written by Turkle (1984), a psychologist and sociologist. Turkle sees the hacker as someone who is fascinated by the possibilities of control in computer

technology, and takes the advent of hacker subculture as a postmodern phenomenon.

In the 1990s, especially in the late 90s, studies about hackers have sprung up as part of the huge social science interest in with the emerging ICT network society. Researchers endeavour to investigate the hacker world and understand the key role that computer hackers play, however, a thorough picture has never been mapped. Researchers hardly avoid categorising hackers. Very often hackers are placed in the context of deviance, crime or the expression of an obsessed user subculture. Chantler (1996) has observed hackers since 1989 and finally brought all the materials together in a thesis in 1996 titled *Risk: The Profile of the Computer Hacker*, which mainly introduces the biographical life of hackers and their activities. Meyer (1989), the criminologist, has studied the social organization of the computer underground from a postmodernist view. Taylor's (1999) book named *Hackers: Crime in the Digital Sublime*, which tried to explore the hacker subculture from a more open perspective, nevertheless, still locates it in the context of digital crime, as the book title suggests. Thomas (2002) discusses the relationship between hackers and technology and portrays hacker culture in terms of their perception of technology, and human relationships (Thomas, 2002: xxiii). In this sense, hacker culture is seen as being formed through interaction with technology, culture and subculture. Thomas concludes his analysis of hacker culture with an account of the two controversial hacker figures, Kevin Mitnick and Chris Lamprecht. I would suggest that while Minick and Lamprecht may represent one type of hacker, the point is that there are so many different expressions of hacker identity. It is inadequate to focus the analysis on criminal hacking alone. As Skibell (2002) states, 'the hacker only exists in the social consciousness' (Skibell 2002: 337). It is this criminal label that is most likely to be attached to hacking. His own work seeks 'to demonstrate that the computer hacker that society assumes is the principal threat is nothing more than a mirage, and that a revaluation of the dangers to computer security needs to be undertaken before sensible policy can emerge.' (ibid.). And he comes to the conclusion that 'the majority of computer intruders are neither dangerous nor highly skilled, and thus nothing like the mythical hacker.' (ibid.: 336). Other monographs have either illustrated a subcultural approach or concentrated on the technical 'geek' life and political ethic of hackers. Generally speaking, none of these stories express the diversity of the

hacker in modern society.

In contrast to the sensational media coverage about the huge damage to companies from the attacks of malicious 'hackers', and their portrayal as negative factors in the development of ICT, a hacker is regarded as a creative and enthusiastic programmer for some groups of actors (see e.g. *New Hacker's Dictionary; The Jargon File*<sup>xiii</sup>). In the context of a system attack, hacking is seen as a technical activity deploying arbitrary codes to challenge the weakness of software, database or firewall. Such codes include viruses and scripts that are both programmes. The operation of these codes might raise people's vigilance towards the network security. Sometimes, codes are written to fulfil clients' requirements for security reasons rather than conducting attacks. Under these circumstances, codes are written to improve software quality or reliability in a way. Most of the time, these hacking tools are available on the Internet. Whilst this situation is said to allow 'script kiddies' to perform malicious acts on the web (e.g. to deface webpages or send viruses), their activities can be seen as an alternative form of self-expressions as well that demonstrates a trial-and-error mindset. It is possible that the existing tools can be improved or new tool can be created to conduct these actions. In light of this, Ross (1991) summarises a variety of narratives found within the hacker community that express their behaviour:

- (a) Hacking performs a benign industrial service of uncovering security deficiencies and design flaws.
- (b) Hacking, as an experimental, free-form research activity, has been responsible for many of the most progressive developments in software development.
- (c) Hacking, when not purely recreational, is [a sophisticated] educational practice that reflects the ways in which the development of high technology has outpaced orthodox forms of institutional education.
- (d) Hacking is an important form of watchdog[, countering] to the use

of surveillance technology and data-gathering by the state, and to the increasingly monolithic communications power of giant corporations.

(e) Hacking, as guerrilla knowhow, is essential to the task of maintaining fronts of cultural resistance and stocks of oppositional knowledge as a hedge against a technofascist future.

(Ross 1991: 81-2)

Though on- /off-line documents such as Raymond's *New Hacker's Dictionary* or *The Jargon File* have sought to define what is the hacker, a single and stable definition of the 'hacker' is hard to give. While the majority of the public still regards the hacker as hostile, for actors in the hacker community, being a hacker does not necessarily mean being exactly good or bad; rather, being a hacker means being creative and innovative. As Ross comments on his list above, 'it is easy to see how the social and cultural management of hacker activities has become a complex process that involves state policy and legislation at the highest level.' (ibid.) He reflects that 'the hacker cyberculture is not a dropout culture; its disaffiliation from a domestic parent culture is often manifest in activities that answer, directly or indirectly, to the legitimate needs of industrial R&D.' (ibid.: 90).

In light of this, it appears that previous literature on hackers actually is of limited value in understanding the hacking practices and their relationship with the ICT innovation system. It presents a reductionist notion, which appears to be, from my point of view, very problematic. Neither do I wish to begin with a proposition that categorises hackers as deviant or marginal actors, nor do I wish to portray hackers simply in a positive light. As Taylor notes in the preface to his book published in 1999, 'analysing the computer underground is inherently difficult... [as in the computer underground] social ties are loose, even by subculture standards' (Taylor 1999: x). It is unwise then to characterise hackers as a single community and hierarchy akin to a gang organisation, as Chantler did in his thesis in 1995. When Taylor studied the relationship between the computer underground and the computer security industry, it turned out to be difficult to pursue because

both groups are far from being coherent and well-established given the relative youth of computing and its hectic evolutionary pace. [Moreover,] the boundaries between groups are unusually fluid and there is no established notion of expert knowledge... It is thus at times problematic, in choosing interview subjects and source materials, to fall back on conventional notions of what constitutes an expert or even a member of a subculture.

(ibid.)

Ross also gives a similar explanation:

While only a small number of computer users would categorise themselves as “hackers,” there are defensible reasons for extending the restricted definition of hacking down and across the caste hierarchy of systems analysts, designers, programmers, and operators to include all high-tech workers—no matter how inexpert—who can interrupt, upset, and redirect the smooth flow of structured communications that dictates their position in the social networks of exchange and determines the pace of their work schedules. To put it in these terms, however, is not to offer any universal definition of hacker agency. There are many social agents ... All [these social agents], then, fall under a broad understanding of the politics involved in any extended description of hacker activities.

(Ross 1991: 92-3)

In short, there is no clearly bounded constituency of hackers. A methodology that on the one hand can explore the social and technical dimensions of hacking, and on the other hand, not lose sight of the various localised definitions of hackers, is required. As the next chapter will demonstrate, social worlds theory is employed to pursue this end. This thesis, instead of presuming hackers as a specific and relatively closed social group, treats the term ‘hacker’ as flexibly as possible. The notion “hacker”, as I will suggest and illustrate, is interpreted differently to demonstrate one's identity and self-expression. Since the notion is not pre-defined, it allows heterogeneous readings and performances.

## 2.8 Hackers and FLOSS Innovation

Is FLOSS innovation distinct from other technological innovations? Is it totally opposite to the conventional proprietary system that gains success in the current capitalist world? Will it undermine the notion that IPRs foster and enable the most valuable innovation? While many have tried to understand the FLOSS phenomenon from different perspectives, this thesis seeks to provide a critical but constructive analysis of the FLOSS innovation process from a socio-technical point of view. The aim of this research is to investigate the identity and relationship of actors and actants found in this social world; to see how actors and actants interact within and across boundaries of different social groups; to understand why FLOSS, if not a new way of doing things, succeeds so well in the computer industry; to observe how the malleable and informal FLOSS practices originating from hacker culture become standardised, stabilised, professionalised and institutionalised in a wider society; and finally to understand software innovation more fully.

As noted in Chapter 1, what is peculiar about software technology is that software is always in some way unfinished, or carries an 'imperfection'. There are various programming languages serving the same purposes, mainly for compilation or interpretation. There is no universal operating system for use, nor is there an entirely bug-free software. Software is vulnerable: if a letter in a code is misplaced, it is unlikely to work properly. There is never a finished 'product' though new versions of programmes are always on their way. Given this, it seems more appropriate to see the production of software as a process than a completed product. It challenges the conventional notion of 'production'. Within the software process, innovation and culture interact together. It is especially the case in terms of the FLOSS, and as MacKenzie notes, 'Open sources software cannot be separated from open source practices of socialising' (MacKenzie 2001: 548). As the Linux operating system illustrates, the source code of the kernel programme could be downloaded, read, configured, modified, and submitted back to a central repository. It is observable that there exists a cluster of practices that mirror with what 'hackers' (computer practitioners) did at the early development of computing. These practices are 'material practices' in MacKenzie's account. The

heterogeneous materialisations of values, desires and practices could mean that the FLOSS community encourages new forms of ICT innovation.

As said earlier, security is one of the main weaknesses of software technology and one of the reasons why FLOSS is favoured (Feller & Fitzgerald 2002). Software testing is rather an endless process as it is impossible to cover all kinds of inputs and outputs. When the Internet was invented, many thought it was the greatest ICT invention there has ever been, because it provided instant and geography-free communication. But later on, it was found that the Internet is not fail-safe. Actually it is very vulnerable to viruses. Viruses, penetrating scripts, or patch programmes<sup>xiv</sup>, are vehicles through which heterogeneous material practices serve as tools for some members (in certain sense of ‘hackers’) in the computing world to challenge orthodoxy. Speaking of ‘tools’, Star and Ruhleder (1996) argue that, ‘A tool is not just a thing with pre-given attributes frozen in time—but a thing becomes a tool in practice, for someone, when connected to some particular activity. The tool emerges in situ’ (Star and Ruhleder 1996: 112). In light of this passage, MacKenzie argues that the ‘hackers’ is ‘the someone’ in Star and Ruhleder’s account. He argues that ‘The suppleness and effectiveness of their software reflects the fact that they have built it for themselves and their own communities of practice.’ (MacKenzie 2001: 547). MacKenzie may be right to suggest that ‘the someone’ is the ‘hacker’, but, as I argued above, this may suggest to clear and well-defined a hacker identity, which I want to steer away from.

The character of software, not least because of the creation and response to viruses, is changeable and in flux, but since it is so easily dispersed via the web, the effects of this instability are much greater than they would have been in less mobile technological systems. In this context, one might argue that FLOSS functions particularly well. This is especially because of the dual role that its ‘members’ play as both producer and user of software solutions to problems encountered in the open source ‘community’. This is unique to the fact that users themselves seldom participate in conventional innovation. Given the nature of software, each phase in the development contains various possibilities. Corresponding with countless requirements, one might expect to find countless options as well. What makes the FLOSS process peculiar is the relation between

requirements and its participants. First, for FLOSS practitioners, the requirements of the software either reflect their personal necessities or an attempt to disrupt and challenge existing systems. They consider their work as “user-oriented”, where they are the users, rather than “customer-oriented”. Second, after finishing writing codes, FLOSS developers will distribute their work on the Internet. They welcome volunteer debuggers and correctors, and at the same time, they also share their work with other users. The FLOSS process is a ‘step-by-step’ one, as modelled by the traditional waterfall software process mentioned above. But each step can be turned into another new project because each phase contains various possibilities. Not only that different people would have different requirements and designs, but also that software testing is rather an endless process as it is impossible to try all kinds of inputs and outputs. It said, the loop is not a closed one but open in this sense that each project is ‘unfinished’ (as each software product is). This open characteristic changes the conventional software process and, I want to argue, brings more possibilities into the innovation system. I explore this more fully in Chapter 5 and 6.

Finally, one should bear in mind that FLOSS generates information goods. As much IT management research has found, ‘when the object of production is physical, the product can be readily differentiated from the process that created it’ (Fulk & Desanctis 1999: 12). But, as Heydebrand (1989) claims, ‘When the core product is service or information, the process itself becomes the product or is indistinguishable from it, leading to a restriction of evaluation to largely processual criteria’ (p. 326).

Although FLOSS is an informational good, its open characteristic challenges the conventional software innovation system especially with regard to intellectual property rights. Despite mainstream software companies' attempts to seal up their proprietary program, this seems to be especially difficult given the character of software. The appearance of the FLOSS indicates that software can be easily made transparent as long as its authors share the source codes with other users. Stallman's concept of "copyleft" has broadened the access to the software source code and makes the adaptation and revision of programmes feasible.

In terms of STS analysis, rather than being a question of whether open source software is technically better, the question from this perspective would be: what kind of activities and what kinds of connections between software and practices or particular activities occurs with open source? Unlike the proprietary software development, there appear to be few hidden social networks in the FLOSS world. Because of the openness, not only is the technology itself transparent, but it seems so too are the social networks among actors. But even if FLOSS helps open up the black box of software technology, not everybody can understand what is inside it. Getting onto [www.kernel.org](http://www.kernel.org) and downloading the source code of the Linux kernel programme hardly makes any sense for someone with little knowledge of the basic programming language. Therefore, there is an argument that the FLOSS is too expert-centred to be user-friendly, and only the software competent can use it. It is true that the core of the FLOSS projects remains very specialised for programmers, but FLOSS is seeking to create user-friendly GUI environment for lay-users (eg. KDE, GNOME). It is the openness of the FLOSS community that is, then, important to examine and to determine how developers and users exchange experiences and opinions towards software and how this acts to broaden the knowledge base and enlarge the boundary of the FLOSS community. How far does FLOSS permit newcomers to access source code and facilitate a self-learning process? Can we identify an informal *social learning* that brings new actors into the innovation system on a regular basis? These questions will be explored in more details in Chapter 7. Over time, the technological outsiders may gradually become mainstream insiders, as I shall discuss later. This feature also complicates the dynamics of the FLOSS innovation system.

### **2.9 Possibility, Uncertainty and Deviant Innovation**

In terms of Schumpeter's (1939) notion of 'creative destruction', innovation and growth lead to the replacement of obsolete products, processes, and firms by more up-to-date and superior successors. And the birth and growth of the new entity must interact with other factors. This thesis will therefore attempt to explore the ongoing but dynamic process of the interactions between actors and actants by investigating their communications and negotiations grounded in their everyday practices.

This discussion of the concept of diverse possibilities existing within the process of innovation also can be seen in terms of 'reverse salients' suggested by Hughes (1983) where problems in technological systems crop up as the systems grow. He argues that, 'Conservative inventions solved these problems, whereas radical ones brought the birth of systems' (p. 80). Hughes has argued that industrial research laboratories, which proliferated in the first quarter of this century, proved especially effective in conservative invention, which involves more unqualified groups, organisation and bureaucracy dampening the originality of inventors and innovators: Hughes claims that the laboratories routinised invention. In addition, Hughes has also observed that radical inventions in disproportionate number still come from the independents. When a reverse salient cannot be corrected within the context of an existing system, the problem becomes a major one, the solution to which may bring a new and competing system. The outsiders, Hughes believed, create the radical inventions that must stand initially without substantial organisational support. Radical inventions are often stifled by organisations that consider them a threat to the technology that they nurture (like the telephone invented by Bell). But radical inventions, created by so-called 'outsiders', are often the genesis of new systems.

Reflecting on this concept of 'reverse salient' as it applies to ICT invention, standardisation, interoperability, imperfection and vulnerability are problems routinely encountered. While industrial corporations or government agencies seek to solve these 'reverse salients' by political and economic steps to control and limit user autonomy, some individual programmers in the computing world clearly show more audacity. They realise that to solve the unstable 'reverse salient' within software needs collective effort from diverse users. The FLOSS is the outcome of this collective creation, evidence, it is suggested, of radical software innovation. From the technological wizards at MIT in 1960s to recent actively independent programmers and hackers, these 'outsiders', in Hughes's context, outside the orthodox institutions, demonstrate considerable enthusiasm to overcome obstacles. As they are not inclined to follow conventional societal norms, hackers are regarded as deviant or even pathological. However, as previous studies of 'deviant science' have suggested, they are often dismissed because they do not conform to the prevailing wisdom and its interests, rather than because of their inadequacies (Dolby 1979).

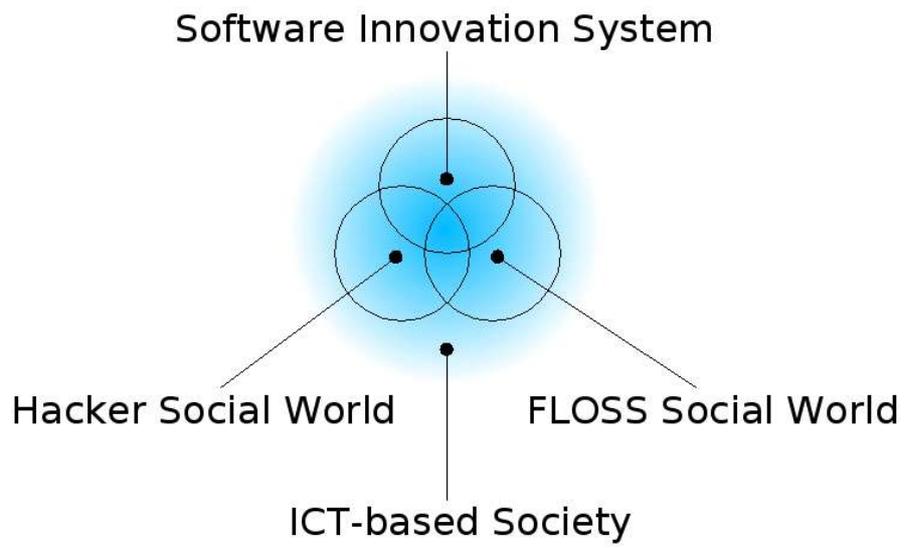
Take the operating system Linux for example. The evolution of Linux from a marginal operating system to a central, mainstream system adapted by many individuals and institutions is a story of conflict between orthodoxy and “deviant hackers” in software production systems. “Hackers” stir up the conventional laboratory pattern of invention, overthrow the conventional path of ICT innovation, and call attention to risk and uncertainty in the ‘digital society’. Returning to the argument of Pinch and Bijker (1989), in the context of a multi-dimensional process of software development, “hackers” challenge conventional software innovation and create a software production subculture that is dynamic and complex. However, this thesis extends Pinch and Bijker’s argument to explore how the system is able to remain relatively unstable, especially as exemplified in open source systems (such the GNU/Linux) whose very rationale is for ongoing change and re-invention.

### **2.10 Conclusion: Innovation and Culture**

Drawing on the literature in innovation studies and software development, I emphasise that the concept of culture and its relationship with technological innovation are crucial in studying socio-technical dynamics in innovation systems, where the collective production of skills and practices emerge that ‘enable social actors to make sense of their lives, articulate an identity, and resist with creative energy the apparent dictates of structural conditions they nonetheless reproduce’ (Brian 1994: 192). Moreover, the development of software is not merely about technological computer architectures, more importantly, it involves an informal and collaborative networking of ideas under strong peer-production, as one will see in the development of FLOSS illustrated in later chapters. Given its characteristics, software technology requires a more effective innovation pattern to overcome its innate problems. The FLOSS innovation system might serve as a socio-technically efficient platform to enrol wider socio-technical resources from the public as well as the private sectors to provide for greater innovation capacity in software development. This will be assessed and discussed in my concluding chapter.

In the text *The Cultures of Computing* edited by Star (1995), the authors explore

a wide range of cultural practices associated with the design and use of computing. To examine “specific kinds of work that people do together, with and around computers, and the ways in which people are brought together in computing practices as computing learners, artists, gatekeepers and scientists—sometimes as insiders, sometimes as outsiders, they adopt Lave and Wenger’s (1992) notion of ‘community of practice’.” (Star 1995: 7). The FLOSS innovation system, as will be discussed later, epitomises such a community of practice where shared knowledge is transmitted through shared engagement in the topic of discussion and how such an embedding practice operates as a culture-creating force (Baym 1995). Such a peer-production process, in its most extreme forms, is often associated with a hi-tech utopianism. However, in a sociological sense, it would be wrong to adopt such a normative position as some insider literatures assume, or alternatively to understand FLOSS as a 'deviant' alternative to proprietary software as some outsider literatures suggest. Instead, I shall seek to explore the actual dynamics shaping practice in the FLOSS social world. As my empirical studies will illustrate, neither the technology, the community, the institutions nor individual users remain the same during the innovation process. The thesis therefore will concentrate on dynamics of an innovation system, struggles of technology evolution, society and identity change, and actor and actant interactions within and across the FLOSS community and see how these processes may shape the overall software innovation more generally (see figure 2.1 below).



*Figure 2.1: Boundaries and exchange: innovation, social worlds and the information society*

## **Chapter 3 The Hacker Social World and FLOSS**

### **3.0 Introduction**

This chapter introduces the research design of the thesis. It presents the principal research questions that are informed by the methodological framework of social worlds theory and the reasons for selecting a qualitative research in the form of ethnography combining several methods including in-depth interview and ethnographic observation, in both virtual and real environments. This ethnographically oriented approach is employed to address the multiplicity, dynamism, and conflictual nature of culture as innovation resources and repertoires. It also describes a methodological shift made during the thesis from a grounded theory to a social worlds theoretical framework. This reflexive presentation of the building of my research design is made to acknowledge that research is never a simple linear process; rather, the research process is continually changing with its surprises, design changes, and reformulation of concepts and hypotheses. Eventually, though, I hope that my final strategy provides for a resilient and theoretically robust analysis.

### **3.1 Previous Research Methodologies for Studying Computer Hackers**

As Taylor (1993) claims in his exploration of the relationship between computer hackers and the computer security industry, it is methodologically problematic to study communities such as hackers through data-sourcing techniques that depend primarily on computer-mediated fieldwork. Because “by its very nature, computer-based communities are not such an easily identifiable social group as their ‘real-world’ counterparts, the researcher does not have a group of people with institutionalised degrees of seniority and knowledgeability with whom they can converse [due to] this lack of physicality” (Taylor 1993: 19). To understand both the virtual and real-life activities/behaviour of participants in the research field, electronic-mail (e-mail) interviews and various electronic-mail publications

became the main empirical data resources for Taylor, supplemented by more traditional face-to-face interviews. Apart from the joint (on-line and off-line) methods adopted in such research, studies on hackers face a conceptual problem (noted in the previous chapter) related to the unstable definition of hackers. For the sake of their research, Meyer (1989) and Taylor (1993) confined their hacker participants to those located in a setting, which is socially deviant and technically subject to security issues. The former text analyses the way in which hackers structure their relations amongst themselves, while the latter tends to look at the motivations and personal characteristics of hackers and addresses the implications of hackers for society as a whole.

Hackers' contribution to and involvement in the development of computing innovation has barely been recognised though their practices have subtly been applied within the computer world, particularly in software innovation. Indeed, from my early investigations in related documents and participation in hacker meetings and conferences, I realised, as I argued in Chapter 2, that it is sociologically inappropriate and makes little sense to try to provide a single definition of 'hackers'. The meaning of the word 'hacker' changes over time and space. To acknowledge the ambiguity of the term, it would therefore be more appropriate to focus on the material practices found among those who see themselves or are seen as hackers and investigate the relationship between hackers and the wider computing world in regard to such practices. Hacking is, thereby, treated as a socio-technical practice that reflects actors' activities and behaviours within the broad context of software programming. The definition of the term is socially constructed, not only by others but, as we shall see, by those regarding themselves as within the social world of hacking. Investigating claim-makings and the shared practices amongst these actors avoids presuming that hacking is intrinsically or solely a deviant or even criminal activity *in its own right*. This also ensures that the thesis can observe diversity within the social world of hacking itself.

In this methodological framework, diverse interpretations of the term "hacker" form different social groups in the field of software innovation. These social groups mutually engage in, interact, communicate and negotiate with one another.

Heterogeneity is incorporated in the hacker social world, while a constellation of collective practices is brought into being, such as the practices of experimenting/challenging existing knowledge paradigms and of sharing information and knowledge. With the development of software technology, the orbit within which hacking practices are found has been extended, and is much wider than was the case for those studies conducted a decade ago, notably in relation to the increasing development of Free/Libre Open Source Software (FLOSS). The collective hacking practices appear to be important factors in the emergence of FLOSS. Hackers in this context are no longer referred to security experts but more conceptually denote an attitude and behaviour within the wider computer world. Heterogeneity becomes the resource that helps mobilise FLOSS innovation. Analysing hacker culture in this regard provides a culturally contextualised understanding of software innovation. Bearing the ethnographically oriented approach in mind, the specific methodological framework of social worlds theory, which will be explained in detail in a later section and subsequently deployed throughout the thesis, allows us to investigate cultural innovation and organisational change in a community of practice at the cutting edge of software technology.

### **3.2 Conceptualising Hackers**

#### **3.2.1 Pilot Fieldwork**

Given the discussion above that there is no singular meaning of hacking and that the hacker community is divided into several groups, a methodology that on the one hand can explore its social and technical aspects, and on the other hand, not lose sight of its more dynamic character, is required. At the early stage of the research when the main methodological framework had not yet been developed, I employed grounded theory (Glaser & Strauss 1967) to let the categories of hackers derive from the initial data, without any preconception. By using a basic ethnographic method in the pilot fieldwork, I hoped that my conceptual framework and theory would emerge from the data. With this point of departure, I would not be obliged to account for all data (Wærn et al. 2001). My aim was to achieve what Silverman (1997) suggested, that is, to ‘both gain access to the collective wholes

that govern behaviour... and focuses on the production of highly individualized accounts gathered in the course of the study' (Silverman 1997: 18). At the beginning of the research, I attended hacker meetings under the virtual organisation of *www.2600.org* held every Friday in London, Hull and Leeds to do observations. In so doing, I got more familiar with my initial research object—hackers. Then, I tried to map out the hacker community to see its demography. I noticed that participants in the hacker meetings were from diverse backgrounds: students, IT workers, teachers, engineers, accountants, social studies researchers, journalists and so on. The meeting was a platform for people with a shared interest in 'hacking' to meet up, to talk about ICTs and security related news, and to exchange experiences in using new ICT tools. Apparently not every participant at the hacker meeting was what conventionally might be regarded as a hacker. Even those who regarded themselves as hackers, also commonly carried another social title such as security consultant or system manager with them. No activity was identified as central or in some sense core at the 'hacker meeting'. The identity of members *qua* hackers at hacker meetings was not central either. The theme of the hacker meetings was broadly based and participants had wide-ranging interests in computing. It was a very informal and loosely defined domain for social interaction.

The mapping of the hacker community became clearer through another early piece of pilot fieldwork at 'Hacker At Large' HAL 2001, a biennial hacker conference in the Netherlands. The setting, unlike most academic or business conferences, was designed to be informal and casual. Participants, mainly ranging from 15- to 35-year-old white males, camped on the campus of the University of Twente. The site combined an advanced high tech infrastructure and a casual social environment, together set up as a digital village. Speeches ranged from security, privacy, cyberpunk, cryptography, copyright, encryption/decryption to social issues related to copyright and licences. Participants had come from many different countries and from different social backgrounds. I met many system managers of commercial companies, from the US air force, and the Dutch police at the site. Many of them came to understand the most recent technologies found in the hacker community. Most important of all, the venue had a strong link to the FLOSS community. The symbols of the penguin that represents GNU/Linux distributions were prominent at the site. Anti-Microsoft slogans were

commonplace. Again, the hacker conference didn't bring people together to “do” hacking. Rather, participants were there mainly to know people, to share information, to explore new ways of living and more importantly, to experience the “freedom”, the most common leitmotif in the free software movement. Though a common background in IT was generally found among most participants, a firm definition of hacker was still impossible to get. Participants described their understanding of hackers in different ways when asked about it. Again, the identity *qua* hacker was hidden rather than self-proclaimed. Participants normally would not introduce themselves as 'I am a hacker'. Rather, they would say 'I am a system manager' or 'I am a father of a three-year-old' to identify themselves. It seemed to be a more subtle sense of recognition of being a hacker. 'Hacker' became an ascribing identity to the individual rather than a clear identity. Unlike formal job descriptions that are assigned, an individual who hacks does not occupy a clearly defined role; it is because of what s/he does -- the practice of hacking. Practices thus come into being as the central issue in our understanding of software innovation process.

The experience of the pilot fieldwork helped me to understand the multiple dimensions of hacking and explore how this might contribute to the ICT innovation system. The boundary of each hacker category was permeable and changeable because of the different perceptions of participants. Some participants did not regard themselves as hackers either because they did not think they were competent or they did not like the term. Some participants who asserted they were hackers emphasised that they were “white hats”, in the argot, or “grey hats”, rather than the malicious “black hats”. The line between white hat, grey hats, and black hats nonetheless, was very fine or blurred. Facing these ambiguous categories, it appeared to be impractical to approach participants in the field engaged with ‘hackers’ as a single subculture. To avoid falling into a reductionist account, it was essential to revisit the fieldnotes and open out and reinterpret my data. In so doing, I was trying to be reflexive to help discover and uncover what was actually happening in the empirical world.

### 3.2.2 Grounded Theory

Dealing with a complex and dynamic field such as hacking, it is at least impractical and probably impossible to ‘develop precise and fixed procedures that will yield stable and definitive empirical content’ (Clarke 1997: 65). In a fundamental sense therefore, the research is better ‘guided by a set of sensitising concepts—less specific suggestive ideas’ (ibid.). In this regard, grounded theory served as a transient solution. The approach, as Orona (1997) remarks, ‘allows for the emergence of concepts out of the data. ... [Q]ualitative research, especially in the grounded theory tradition, is not for those who need a tight structure with little ambiguity’ (Orona 1997: 179). ‘Grounded theory provided the framework for taking observations, intuitions, and understanding to a conceptual level and provided the guidelines for the discovery and formulation of theory’ (ibid.:182). Although the pilot fieldwork was limited, some essential facts had been observed and could be induced into stronger propositions to refine the previous research questions.

Given the diversity and the ubiquitous involvement of FLOSS, it seems to be more promising to relocate the research object in a more definite and established field of software programming. Having said that, the field of FLOSS appears to be an adequate ground for more advanced research given the pilot data showing the overlapping of the FLOSS community and the hacker community. As also observed in the pilot fieldwork, fluid information and capital (socially and intellectually) flows are evident features in the hacker social world. These references are embedded in the mundane interactions between human and non-human. In addition, it will be important to delineate the complex interactions and relationships between actors in the social world, and to anchor these in the im/materialities of technical objects, particularly software and its peripheral development tools. This will enable us to see how the practice of ‘hacking’ expertise contributes to the software innovation process, especially in the FLOSS development.

To determine the research question, as Clough & Nutbrown (2002) suggest, is like applying the principle of the Russian doll to ‘break down the research

question from the original statement to something which strips away the complication of layers and obscurities until the very essence—the heart—of the question’ (Clough & Nutbrown 2002: 33). In so doing, the research is not going to lose sight in the ambiguous definition of hackers and struggle with the legitimacy of their behaviours. Instead, the research question has been sharpened and more defined, concentrating on how the collective practices are formed and employed in the development of FLOSS, and how actors negotiate and communicate their different interpretations over the common doings in this community of practices. Different methodologies offer different ways of interrogating the world. Grounded theory achieved its primary task for me of helping distil my research object. But for this research, a different analytical methodology is more helpful in terms of its ability to answer the questions asked and in terms of investigating what people actually do. To meet this end, *social worlds theory* was adopted as the methodology for the next stage of the research to analyse the dynamics of the software innovation process where diverse actors and actants engage. This methodological framework will be used throughout the research. Data collection will involve both virtual methods such as on-line observation and email interview and more traditional qualitative methods such as situated observation and the face-to-face interview. Furthermore, documentary sources include those in the printed media (e.g. newspapers, hard-copy magazine, government or business white papers etc.) as well as in the electronic media (e.g. webpage contents, discussion threads in mailing lists, conversation in online chat room etc.). Let me now introduce social worlds theory and its value to my thesis.

### **3.3 Methodology**

#### **3.3.1 Social Worlds Theory**

To analyse a dynamic and complex field, Social Worlds Theory (SWT) has been selected as the approach to adopt from among other STS methodologies. To explain my theoretical position, I will begin with a brief discussion of the origins of SWT.

It was Shibutani (1955, 1961) who ‘transform[s] community studies into social worlds studies and initiated explicit social worlds theory development’ (Clarke 1997: 68). He treats reference groups as ‘the organizers of social life and individual commitments’. As Clarke argues, ‘reference groups generate shared perspectives which form the basis for collective action organized through the construction of social worlds’ (ibid.: 68). ‘The theory was then extended for sociologists to understand the concept of commitment as a basis for social action’ (ibid.). Becker (1974, 1986) and Strauss (1978) subsequently define social worlds as ‘groups with shared commitments to certain activities sharing resources of many kinds to achieve their goals’ (ibid.). Strauss’s explanation is instructive here, and I will quote him at length:

In each social world, at least one primary activity (along with related activities is strikingly evident... There are sites where activities occur; hence space and a shaped landscape are relevant. Technology (inherited or innovative means of carrying out the social world’s activities) is always involved.... In social worlds at their outset, there may be only a temporary division of labor, but once underway, organizations inevitably evolve to further one aspect or another of the world’s activities.

(Strauss 1978: 122)

The actors observed in my fieldwork share a typically simultaneous cross-boundaried participation in a number of social worlds, and this phenomenon demonstrates that both the participation in and the structure of the hacker social world are highly fluid.

SWT is widely applied in the research field of health and medical studies. Clarke and Montini (1993) employ SWT to analyse the heterogeneous construction of the abortion pill RU486 by various actors. They begin by attempting empirically to specify all the key individuals and social groups ‘active’ around the technology, around prior or subsequent related technologies, or related social issues. Consequently, they find out not only that the identity of the nonhuman actor—RU486—is unstable and multiply shaped by various social

groups such as scientists, pharmaceutical companies, medical groups, antiabortion groups, women's health movement groups, and other who have produced situated knowledges, but also that, in practice, there are other (previously invisible) implicated actors—the downstream users and consumers of the technology. As Clarke and Montini argue:

Meanings, identities, symbols, and rhetorics that the major social groups in the American abortion arena have constructed for RU486. Many of these constructions are historically embedded, reflecting the continuing controversial status of the reproductive sciences qua science as well as abortion qua practice.

(ibid.: 43)

Star and Griesemer (1989) link the concept of social worlds theory with the idea of 'boundary objects'. This latter notion originates from their analysis of the coordination efforts of members of several different social worlds in building the Museum of Vertebrate Zoology at the University of California, Berkeley (Fujimura, 1992). In response to the actor network theory proposed by Callon (1986), Latour (1987) and Law (1986), which claims that actors' 'interests' are 'translated' in order to enrol them, 'Star and Griesemer shift the focus of the network model to the multiple translations present in scientific work' (Fujimura 1992: 171). In Fujimura's view, 'They use an "ecological" approach framed in terms of understanding science as collective action from the viewpoints of all the actors and worlds involved, and thereby avoid the pre-eminence of any one actor' (ibid.).

The concept of Star and Griesemer's boundary objects is distinct from Callon and Latour's actor network theory (ANT). Whereas ANT focuses on the strategies and negotiations among social worlds to stabilise 'fact,' SWT are concerned with the collective work that members of different social worlds manage to cooperate (ibid.). The focus on different viewpoints and agendas in the negotiation process through which common understandings are created makes SWT a useful tool to analyse diverse viewpoints and concerns of all the participants without presupposing consensus. Social relationships are situated and the identities built

can be better explored in the SWT framework. Hence, ‘reliability across domains and ... information which retains integrity across time, space, and local contingencies can be preserved’ (ibid.). In short, as Fujimura comments:

The strength of the concept of boundary object lies in its attention to multiple and divergent actors, social worlds, meanings, and uses. ... [B]oundary objects are often ill-constructed, that is, inconsistent, ambiguous, and even “illogical”. Yet they serve to accomplish the work to be done as defined by the actors involved. Since the local viewpoints (‘interests,’ requirements, desires, languages, methods) of different groups are usually not identical, rigid or strongly structured entities are less likely to be able to absorb divergent instances and still maintain internal coherence or robustness.

(ibid.)

However, SWT is often criticised because a boundary object as depicted by Star and Griesemer lacks empirical stability. As Fujimura continuously argues,

Though the concept of boundary objects takes the interactions or communications of various actors into account, its ignorance that boundary objects are more easily reconstructed in different local situations to fit local needs, however, hardly reaches facts of stabilization that Callon and Latour’s actor network theory pays attention to. It’s just as Latour declares, they (researchers of boundary objects) have to enrol so many others so that they participate in the continuing construction of the fact.

(ibid.)

To overcome the limits of the concept of boundary objects, Fujimura proposes the idea of ‘standardised packages’, another product of social worlds theory to explain how collective action is managed across social worlds to achieve enough agreement at various times to get work done and to produce relatively (and temporarily) stable ‘facts’. The concept of standardised packages facilitates both

collective work by members of different social worlds *and* fact stabilization (Fujimura 1986, 1987). As Fujimura explains,

A package differs from boundary objects in that it is used by researchers to define a conceptual and technical work space which is less abstract, less ill-structured, less ambiguous, and less amorphous. It is a gray box which combines several boundary objects (in this case, genes, cancer, and cancer genes in proto-oncogene theory) with standardized methods (in this case, recombinant DNA technologies, probes, sequence information) in ways which further restrict and define each object. Such codefinition and corestriction narrows the range of possible actions and practices but does not entirely define them. These properties of a package allow for a greater degree of “fact (and skill) stabilization” and less of the undermining which concerns Latour (1987: 208; 1990). Simultaneously, however, a standardized package is also similar to a boundary object in that it facilitates interactions and cooperative work between social worlds and increases its opportunities for being transferred into, and enrolling, other worlds; it serves therefore as an interface between multiple social worlds.

(Fujimura 1992: 176)

Through this strategy, Fujimura is able to analyse how a scientific bandwagon, a situation where large numbers of people, laboratories, and organisations rapidly commit their resources to one approach to a problem, is formed, ie. how members of so many different social worlds come to agree to participate in or support molecular genetic studies of cancer, and especially studies framed in terms of a single theory of cancer. The concept of standardised packages provides both dynamic opportunities for divergent meanings and uses as well as stability.

Fujimura’s package of concepts and standardised tools is useful for understanding both the stability and the dynamism of innovation (e.g. the disciplinary innovation of the oncogene theory). In her framework, objects that are used to craft an innovation (e.g. the oncogene theory), both less structured

concepts (e.g. cancer, cells, genes, and cancer genes) and standardised tools (e.g. such as probes, the language of sequence information, and sequence databases), provide a way of looking at a theory which appears to be both simple and complex, both static and dynamic (Fujimura 1992: 204). In other words, the idea of ‘standardised packages’ help to explain how innovation can be continuous across time and space through different social worlds.

### **3.3.2 Methodological Rationale**

Given the discussion above, I would like to suggest that, the FLOSS phenomenon can be seen as a “standardised package” in the terms Fujimura uses, and consists of a repertoire of artefacts (e.g. software and programmes) and less structured concepts (e.g. freedom, privacy, intellectual property). Given that FLOSS is a collective and dynamic interface that acts to translate the interests of actors from different social worlds (e.g. governments, corporations, individuals, and organisations), in scrutinising the dynamics of the software innovation process where diverse actors are engaged, it is essential to employ SWT to understand the socio-technical construction of the FLOSS arena in terms of heterogeneity and contingency. In the FLOSS ‘package’, boundary objects (be they material, conceptual or social practices) are studied in order to understand the emergence, stabilisation and dynamics of the development of FLOSS where diverse interests and perspectives across social worlds are translated and enrolled to construct the larger social arena.

Fujimura’s approach helps us to conceptualise and analyse the socio-technical heterogeneity of practices. Although it is difficult to fully understand and represent the processes of producing science and technology in different situations and from different viewpoints, Fujimura still endeavours to examine this malleability and dynamism of truth matters. As she says reflexively, “While we will never be able fully to understand and represent the views of the other, this does not mean that we should not even attempt to “explain” science’, says she reflexively” (Fujimura 1991: 218).

Furthermore, it is also the case that SWT can bring the formation of identity into

its analyses. As social worlds and identity are interactively constructed, perspectives and aspirations emerge dynamically from this interaction. Identities are formed with meanings and interpretations that are both culturally created and mediated. While views, opinions and reactions of others significantly influence actors' conceptions of themselves, these interpretations or perspectives are based in communities or social worlds. This is exactly why one could never ignore the identity of actors when telling the story and why the social worlds theory is therefore a useful methodology to explore the identity of the actors (the sense in which they are 'hackers') in this research. It is worth noting that the emergence of identity is a complicated process because communities are not clearly circumscribed and defined, and individuals usually participate in many different communities simultaneously. Moreover, individuals also participate in the ongoing construction of social worlds. Thus, identities and perspectives are "multiple, processual, and dialectical" (ibid.). There is no simple one-to-one mapping between perspective and community membership. This focus on the multiplicity, fluidity, temporality, and processual nature of identity, perspectives, and social worlds is the advantage of SWT. The analysis of the tension between indeterminacy of situation and structure, identity and sociohistorical location, and political choices and collective memory, provides no neat categories, no certain way to classify people, ideas, actions. This makes SWT superior to other social theories when studying the complex interactions and relationships in society.

One might ask why not use ANT as the methodology. Indeed, there are many affinities and compatibilities between ANT and SWT, but they differ in degrees, stances, and goals (ibid.). While both theories are constructionist, relativist, and focused on relations among actors, SWT 'attempts to view the constructed world metaphorically over the shoulders of all the actors' (Clarke & Montini 1993: 45), rather than following a dominant actor in a network. As Clarke & Montini explains,

[SWT] emphasizes process, plurality, flexibility, novelty, the generation of many theories and concepts, and attention to local situations, and thus prevents us from falling into the rigid structural frameworks of functionalism and Marxism. Strauss argues that

phenomena are so rich that we should mine them for more stories, more concepts, more “grounded theories”, more ways of “seeing” phenomena rather than limit ourselves to one set of concepts, theory, or way of seeing.

(ibid.: 226)

Given the diversity in the field shown, ANT would have been less useful than SWT as there would have been difficulty in locating the central actors in the FLOSS arena. It appears instead that a software developer can also be a user, an informed one. The boundary of each social world/subworld formed is malleable and cross-boundary activities often occur. In this regard, it is problematic to concentrate merely on one single actor/actant. SWT provides an overall perspective on actors from different social worlds engaging in a collective arena. This methodology offers each actor or social group an equal chance to express themselves and the researcher is able to analyse the complicated interactions between them and the heterogeneous constructions of the technology.

There are always some other actors that the researcher hasn't picked up to discuss playing around in the arena, and these actors may come up to challenge the neutral view of the researcher. And indeed, researchers are constructed by other actors (ibid.). Drawing upon Clarke and Montini's argument, 'relativism and reflexivity do not preempt advocacy or action on the part of those we study – or ourselves' (ibid.: 69). Moreover, they have reflected this thinking in the title of their paper, 'The many faces of RU486', to show that what they see depends upon where they stand. Their ground also mirrors Fujimura's argument in her paper 'The sociology of Science: Where Do We Stand?' (Fujimura 1991: 237). She answers this question by acknowledging the need to be reflexive about her position:

There is no finally stable ground on which any of us—feminists, sociologists of science, people of color—base our stories. . . . When I write, I must take responsibility and hold myself accountable for the final perspective. The point is to make explicit to myself and to my audiences just where I stand, my operating perspective, and the

ground on which my concepts are constructed.

(ibid.)

I find Fujimura's argument especially helpful in differentiating my work from previous research on hackers. From my perspective, hacking is not an activity that has a single or stable meaning in the digital world. Hacking involves a range of practices and can be found in different contexts. Those who engage in some or all of these practices also very usually occupy diverse social worlds at the same time. The concept of 'hacker' therefore is an analytical notion that points to these hacking practices that is expressed through and embodied in real-life social actions. In this research, when I use the term 'hacker' here, I actually refer to an unstable social category linked to particular practices, rather than a readily identifiable, empirically given type of person or subcultural group of people. This contrasts with much of the existing literature on hacking that has presumed hackers exhibit a very specific pattern of behaviour through membership of a strong hacker subculture (Turkle 1995; Taylor 1999; Thomas 2002). While there may be circumstances where hacking practices come together and help to engender strong cultural networks that shape and inform hacking itself, the boundaries of these networks are much more permeable and mutable than subcultural theory would allow. While noticing that there is no static 'hacker social world' consisting of groups who hold particular and homogeneous 'ideologies', in employing sociological concepts and categories, we shall remain 'sensitive' to possible misfits during the research process (Fujimura 1991) in order to keep the sociological reality depicted updated.

### **3.3.3 Validation of Research Scenario and Core Research Questions**

What might this constellation of 'hacking' practices look like and so serve my fieldwork? I began by drawing on my pilot fieldwork and constructed a provisional set of practices.

Thus, in light of available documentary and my early empirical data, I suggest that to engage in hacking, one has to have a basic competency, that is the ability to program computer software. Hacking requires coding. Some codes/scripts are

novel and might be shared with other users. Other codes are challenging and designed to explore the vulnerabilities of software, as is the case, for example, with computer viruses. To obtain "analytic generalisation" (Yin 1994: 30-32) with the concepts of "hacker" and "hacking", I suggest that hacking practices shall be seen as a pattern that contributes to software engineering and which include:

Practice 1: Writing creative codes/scripts and sharing them.

Practice 2: Writing challenging codes/scripts to explore software vulnerabilities.

Practice 3: A strong interest in decryption, code-breaking.

Practice 4: Developing novel hardware and sharing the proprietary information on which it is based.

Practice 5: An interest in tackling software problems and resolving them.

These practices, commonly observed in the "social world" of hacking, can be seen as a repertoire of actions that might come together in their entirety as a constellation of hacking activity, or that might be pursued more variably and unevenly by social actors.

I sent out this constellation of hacking practices to those mailing lists [Cyber-And-Society] and [softwarestudies] where scholars doing research on ICTs and their social impacts can be focused. I mentioned in my message that I was trying to develop a classification of practices related to how people engage with mainstream commercial software. I showed the list of practices and encouraged any ideas/comments, especially if anyone could see whether these practices might be regarded as a constellation that points to a particular type of behaviour? Many responses characterised such practices as a concern for 'users' at different levels in the ICT system. However, one of them pointed out that my descriptions 'obviously sound very much like what many might call "hackers"'. This seemed to confirm for me a blurring of the demarcation between 'users', particularly competent ones, in ICTs and what might be considered to be "hackers".

My other observations in the field of software security also appear to validate my approach towards membership of the hacker social world. The codification and formalisation of hacking tactics and knowledge in network and system security,

such as book publications (e.g. *Hacking Exposed*) or conferences (e.g. *Blackhat* or *DefCon*), make a wider engagement, particularly public participation, in hacking possible. From these sources, participants can learn how to design and implement Internet threat modelling that will help to determine appropriate countermeasures, discover the tactics “hackers” use to get into one’s systems and learn proven techniques for thwarting them. The hacking practices and knowledge are shared among a range of actors from different social worlds. A governmental security manager at a hacker conference might not regard himself/herself as a hacker, but s/he is doing exactly what a hacker does. In adopting the hacking practices, a delegate has inevitably shared the same interest as a hacker in a collective field. In taking up the hacking practice, a delegate shifts his/her identity from an outsider to an insider in the hacker social world. While the cross-boundaried performance might ascribe hybrid identities to the actors (as will be discussed in later chapters), the collective practices adopted by either actor, be it industrial, governmental delegates or individual self-proclaimed hackers, all have impacts on the development of ICTs. Therefore, it proves to be sensible to study the relations between ICTs and the hacker social world, and how a variety of socio-technical meanings are given to the collective hacking practices, instead of prefixing a hacker identity to any presumed social groups or individuals<sup>xv</sup>.

In light of this, I prefer to suggest that the concept of ‘hackership’, with connotations of the term ‘craftmanship’, be adopted to point to a collection of activities among diverse actors who are good at programming and problem-solving. The hacker social world denotes a competence and display of hackership, where people sharing diverse interests in hacking, with dual or multiple memberships across the hacking boundary interact. The concept of ‘subculture’ is essentially flawed due to its attempt to impose a hermeneutic seal around the relationship between hacking practices and members. Indeed, several subcultural scholars have argued that ‘groupings which have traditionally been theorised as coherent subcultures are better understood as a series of temporal gatherings characterised by fluid boundaries and floating memberships’ (Maffesoli 1989, 1996).

As noted above, there might be some circumstances where hacking activities

coalesce and create visible social worlds that are occupied by a range of social actors. Under certain circumstances, this social world might form of stronger hacker cultural network, especially, as in the process of ‘deviancy amplification’, hacking practices become the focal point for computer users as a response to regulatory authorities’ attempts to restrict these very practices. But if such circumstances or conditions change, the network may dissolve or be re-defined, such as, I want to argue, occurred over the past decade or so with the emergence of FLOSS as a form of institutionalised ‘hacking’, inasmuch as it is based on a number of the practices noted above (viz. 1,2 & 4).

To put the argument in a nutshell, this research studies the hacker social world where actors come from different social backgrounds and occupy a variety of social positions and statuses with a common engagement in hacking. As the hacking boundary has a close connection with other social worlds, it is not accurate to understand the hacking group through reference to one boundary. Therefore, the proposition, based on SWT, is that it is essential to conceive of and study a range of interactions, to see how hacking practices can be found in a wide range of social settings, to observe how hacking practices are built into different social worlds and become institutionalised, and to map the social structure behind the hacking practices.

### **3.4 Methods and Meanings**

Before selecting the research methods for the next stage of data collection, one needs to think of more practical matters: what kind of data is required in order to answer the research questions? It is suggested that the FLOSS social world would be the best field to serve the purposes of the research. This would prevent the research from falling into endless debate on the controversial issues of hacking with attributions of legal and illegal activities ranging from legitimate creative programming techniques to illicit lock-picking and the manipulation of worldwide phone/computer systems. In addition, case studies of the GNU/Linux community will be of especial value to the thesis. Archival data has shown that the GNU/Linux community has the largest population of all FLOSS social groups, which widely overlaps with the hacker social world. As an established field,

various documents about the GNU/Linux community are available for the research as well, though these documents should be treated as the products of the context in which they were generated.

Research methods ‘cannot be true or false, only more or less useful, depending on their fit with the theories and methodologies being used, the hypothesis being tested and/or the research topic that is selected’ (Silverman 1993: 2). To explore how the FLOSS arena is constructed, to understand actions and meanings in the context of each social world, and to emphasise that software innovation is a process rather than a state, qualitative methods are more helpful than quantitative ones. The research field is also heavily dependent on tracking on-line activities, while the boundaries of the organisations that compose them are becoming increasingly permeable (Dutton 1999: 474). As a result, virtual methods (computer-mediated communication (CMC)) are required for this thesis and will be integrated with traditional qualitative methods. In more detail, the research methods used in this research are as follows.

### **3.4.1 Ethnography**

Observation is fundamental for this research especially to help me to answer the research question that *How are we to understand the heterogeneity of the social world of hacking?* Only through adopting a sociological ‘gaze’ that treats its subjects of interest symmetrically can we ensure that the diversity of social actors within a broad social world is captured. This will enable me to get a picture of the characters and composition of those engaged in hacking practices and the wider FLOSS social world. Following on this fieldwork, the next fieldwork would take place at a Linux conference, LinuxTag 2002. The site was divided into 3 milieus: the exhibition, the conference and the social activities. The multidimensional settings allowed me to approach various actors in the FLOSS social world including delegates from industry, governments, academia and community. A special session named ‘hacking competition’ especially caught my attention. This session at the Linux conference was to help confirm my hypothesis about the close relationship between the FLOSS community and the hacker social world. Another interesting sign of this was that I saw several participants at the Linux conference

wearing the T-shirt from the hacker conference “HAL 2001”, where my pilot fieldwork was initially conducted.

Apart from observation at the conferences and meetings, virtual ethnography was employed to observe the actual practice of participants in the FLOSS social world. For one of the case studies, as will be explained below, I observed the mailing list of the York Linux User Group (YLUG) for over a period of 15 months. Located in York, this online fieldwork accompanied with off-line observation of the group meetings taking place once every 2 weeks on the campus of the university of York provided me with rich on- and off- line narratives of the actors’ situated practices. As research done by Silvonen (2002) on the Finnish LUGs (FLUGs) demonstrates, ‘FLUG hosts on-line and real-life activists as well as radical and moderate hackers. The meaning of ’Linux’ is being constructed in continuous debates between these groups.’ (Silvonen 2002). Silvonen is right to argue that LUG is constructed virtually and in physical encounters. Face-to-face meetings still serves social purposes that virtual communication cannot achieve alone. Therefore, my participation in the group meeting enhanced on-line observation, so for example, I could know someone posted messages frequently in person. As a sociologist, I bear the basic assumptions of ethnography in mind that “the ethnographer should be an informed outsider, should avoid ‘going native’ and should work through ‘writing the culture’, i.e. both annotating and interpreting the data in the form of text” (Wærn et al. 2001). I declared my identity at the beginning of the fieldwork and tried to overcome the anxiety of participants that my fieldwork was an invasion of privacy. Additionally, apart from a notice message I sent out to invite the members to come to my talk based on the data of YLUG, I did not get involved elsewhere in their regular discussions.

Though observation is useful at the exploratory stage of the research to get a general view of the general practices associated with hacking, it is difficult, however, to get the data of the actual practices of individuals from different social worlds. Observation was limited in this study due to time constraints, the relatively large number and different settings of organisations considered, and the potential for gaining access, particularly to firms, for this kind of research. Additionally, ‘qualitative researchers also argue that observation is not a very “reliable” data-

collection method because different observers may record different observations.<sup>7</sup> (Silverman 1993: 9). To address the other research questions, interviewing was chosen as another primary instrument for data collection.

### 3.4.2 Interviews

The interview, either face-to-face or E-mail, is a useful and effective tool to gather particular forms of data, and it was used to answer several research sub-questions: 1) what are the practices of the programmers in the FLOSS community; 2) how do their practices influence the development of software and the wider society; 3) how is the identity of the programmers in the FLOSS community constructed according to their practices; 4) how is the wider software innovation system being constructed? While it may be difficult to measure the degree of creativity and the influence of hacking practices on general software innovation, interviews helped understand the subtle communication and social relationships among actors when the diffusion of new ideas, and new practices occurs.

Face-to-face interviews were obtained via snowballing among participants at European hacker/Linux conferences and meetings where potential interviewees appear. These face-to-face interviews were semi-structured and took an average of 45 minutes. The interview schedule was designed for technical experts in the fields, however, the degree of expertise was hard to measure and sometimes an experienced Linux user was considered as qualified for interview (see the interview schedule in Appendix 1). Due to the time constraints and limits of different settings, I also collected more E-mail addresses of potential interviewees from the fields and contacted them to do interviews via E-mail after the conferences. A snowball sampling technique was also used to collect more interview sources. Moreover, I cross-posted an abridged version of the interview schedule (see the abridged interview schedule in Appendix 2) onto targeted mailing lists and discussion groups<sup>xvi</sup> to get responses randomly. Specifically selected interviews were conducted at VitaNuova software company at the York Science Park and a number of academics at the university of York. The total number of interviews, including face-to-face and E-mail, is 40 (see Appendix 3 for the list of interviewees). At this number of interviews, theoretical redundancy was

reached.

Face-to-face interviews in this research, unlike those in the framework of Taylor (1993), were largely used to begin to open up the area for explanation, as were E-mail interviews conducted with other respondents. In other words, face-to-face and e-mail interviews enabled me to explore the boundaries of the field with a range of respondents generated through snowball or network sampling. An E-mail interview however, is different from a face-to-face one in the data collection process. As Bloor (1997) notes, 'all data are shaped by the circumstances of their production, and different data produced by different research procedures cannot be treated as equivalent for the purpose of corroboration' (Bloor 1997: 38). Hence: 'What is involved in using different research methods is not the combination of different kinds of data *per se*, but rather an attempt to relate different sorts of data in such a way as to counteract various possible threats to the validity of [the] analysis' (Hammersley and Atkinson 1995: 231-2). There are some evident differences in the two instruments. As Taylor (*ibid.*) notes, the email interview is low-cost and geography-boundless. Moreover, 'the methodological implications of e-mail are ambivalent to the extent that it allows the respondent more of a chance to reflect upon issues, and so to refine their responses. This may increase the accuracy of replies, but it may also reduce some of their spontaneity.' (*ibid.*: 25). It is also the nature of email that allows respondents to answer the email whenever they want. This aspect of email, in Taylor's view, may facilitate apparently greater willingness to communicate than by face-to-face or phone methods. But on the other hand, without the social pressure from the interviewer, without 'visual or aural cues [that] can be used ... to intimate the limits required for the response to a particular question' (*ibid.*: 26), the efficacy of the exchange might be reduced. Overall, email correspondence seemed to be less burdensome for both the interviewer and the interviewees, but judging by the detail of the replies from the interviewees in this research, this doesn't mean the authenticity and efficacy of material derived by email interview were reduced. And the particular nuances and thrust of a question overlooked would be addressed in a second message sent back to the correspondent. Both Face-to-face and e-mail interviews serve the same end in this research, to help ensure the validity of the analysis. A group interview was also conducted at the hacker conference but this form of interview is not central to this research.

### 3.4.3 Documentary analysis

A qualitative method is more promising in terms of its ability to secure full answers to the questions asked and in terms of investigating what people actually do. However, the criticism of a divergence between reported and observed behaviour often levelled against surveys can also be applied to interview research. Thus, ‘a full sociological analysis cannot be restricted to interview data, it must also consider the material traces’ (Hodder, 1994: 395). Combining interview research with material traces of behaviour in the form of documents can add other dimensions to the data collected and give important and different insights from that provided by interviews. It allows ‘new light to be shed on topics and ... different facets of problems to be explored’ (Bloor 1997: 1). Documentary analysis therefore serves as an important method in the research. It is clear that documents should never be treated as neutral texts. Documents, whether they are produced intentionally to record the social world or not, all denote the values, interests and purposes of those who commissioned or produced them. “Such creations may be regarded as ‘documents’ of a society or group which may be ‘read’, albeit in a metaphorical sense.” (MacDonald 2001: 196). Documentary researchers, therefore, should acknowledge that “documents which are intended to be read as objective statements of fact are also **socially produced**” (ibid., emphasis is made in the original text). “They are produced on the basis of certain ideas, theories or commonly accepted, taken-for-granted principles, which means that while they are perfectly correct – given certain socially accepted norms – they do not have the objectivity of, say, a measure of atmospheric pressure recorded on a barometer.” (ibid.)

A number of documents related to hacking activities and the development of FLOSS, in either virtual or printed form, were identified as key sources. They include public records (e.g. governmental and industrial white papers), media reports (e.g. newspaper and magazine clips, textbooks or tool books in software engineering), private narratives (e.g. programmers’ todo-lists, chat room dialogues, discussions on mailing lists, web blogs and wiki pages), and biographies of a number of self-proclaimed hackers and programmers. Apart from checking the “accuracy” and “objectivity” of the documents, as said, a documentary research

needs to be aware of the account which reveals “the teller’s interests, perspectives, and presuppositions” (Hammersley and Atkinson 1995: 160). While more formal or official documents – such as ECIDA (European Commission Interchange of Data between Administrations) open source migration guidelines – provided instructive information about agencies and a means to check the accounts given by informants’ interviews, they should not be used to check the validity of the data collected. To use different research methods is to get the accounts of different participants located differently in the setting. ‘Even if the results tally, this provides no guarantee that the inferences involved are correct. It may be that all the inferences are valid, that as a result of systematic or even random error they lead to the same, incorrect, conclusion.’ (Hammersley and Atkinson 1995: 231-2).

### **3.4.4 Case Studies**

A case study is an empirical inquiry that “investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident” (Yin 1994: 13). One of the major strengths of the case study as a research method is its thickness, that is to say, richness in both the quantity and variety of information it includes. Because of the thickness, a case study provides us with a means of retaining “the holistic and meaningful characteristics of real-life events” (Yin 1994: 3). The contextual thickness makes a case study appropriate for “how” and “why” research questions because answering these questions deals with operational links needing to be traced over time (Yin 1994:6). In this research, the detailed investigation of FLOSS phenomenon through close attention to its context by using multiple sources of evidence and various methods of data collection will help to examine the innovation process by which new FLOSS technologies are created, and see how this is ongoing and involves diverse groups who give the technology different meanings. (Robson 1993: 146; Yin 1994: 12-13). In chapter 6, the case study of EMACS serves to explain how actors from different backgrounds contribute multiple ways of knowing, understanding and resolving problems that arise in the FLOSS innovation process. In chapter 7, the case study of YLUG serves to demonstrate the values of local tinkering, skills and tacit knowledge in FLOSS innovation. In light of data collected from YLUG, I investigate how locally

defined software problems and locally crafted solutions towards the problems are codified and translated into expert knowledge within FLOSS innovation through the display of particular practices and ongoing debate. For each of these case studies, various research methods are employed, namely documentary analysis, interview and observation.

### **3.5 Research Ethics and Reflection**

To deal with the issue of confidentiality and to protect individual identities, consents were acquired from each interviewee. I assured my informants as far as possible of commercial confidentiality and anonymity in published work, even though most interviewees have said that they do not mind having their names revealed. This was primarily given in the consent letters shown at the start of the interviews describing the study and asking whether they would take part, and that was also reiterated at the end of the interviews. (In an email interview the issue is addressed at the start of the mailing.) Additionally, all respondents' names have been changed to protect their anonymity.

Confidentiality and anonymity can only be guaranteed 'as far as possible' because the extent to which absolute confidentiality and anonymity, especially in my smaller case studies, can be retained is questionable. There is always the chance that insiders may recognise themselves, others or specific organisations. Organisations and firms, for instance, may be familiar with specific incidents of non-compliance and can therefore recognise the firm or individuals in the interview. As to the fieldwork, as claimed earlier, before entering the field, I declared my identity at the beginning of the fieldwork and tried to ease the feeling of participants about my presence in conducting research on them and their activities. Perhaps because the field is such a heterogeneous milieu, and because most FLOSS and hacker activities are open for all, the accessibility problem has been less difficult. While the fieldwork in the FLOSS community was seemingly less an invasion of privacy, the disciplinary difference and divergent interests (my interest in the social issues vs. their interests in plain technical issues) appeared to be more conflicting.

This became evident when I was writing up the case study of EMACS. To get a richer picture of the proclaimed functions of EMACS, I posted a brief survey onto YLUG's mailing list asking about the advantages and disadvantages of various versions of EMACS. Many members kindly offered their accounts, though a particular one disapproved of posting this type of question on the list. It was said that,

I don't know if it's just me, but I'd have been happier if a request for information list this had either come from someone who has posted here before and had more detail on what it's being used for. So far at least it's not set off the emacs v vi thread. This might just be me being grumpy of course. I get fed up with the amount of "please do my homework for me" questions I get elsewhere so can be overly sensitive.

(AC120104)

As pointed earlier, apart from a notice message I sent out to invite the members to come to my talk based on the YLUG data, I did not get involved elsewhere in their regular discussions. I did not seek to 'contextualise' myself in the fieldwork. My objective was to get detailed data, not to provide the group with a new member. As guided in various guidebooks of sociological research methods, "one must maintain a certain detachment in order to take that data and interpret it." (Fielding 2001: 149). One might argue that my observation is likely generate the problem of 'not getting close enough', "of adopting an approach which is superficial and which merely provides a veneer of plausibility for an analysis to which the research is already committed" (ibid.). However, I reckon my contribution to the FLOSS community is symmetrical, rather than just being a one-way receiver. My research is conceptually as well as empirically constructive to the understanding of FLOSS phenomenon. Perhaps it is more difficult to take care of each member's feelings when conducting observation in a group, particularly a local, relatively close group. But on the other hand, my research interest based in the disciplines of the social science differing from the interviewees' mainly based in natural science or engineering is probably the other reason for this frustrated outcome that happened frequently during my data-collection process. When being

asked of their own definition of ‘hackers’, very often my interviewees would just refer me to read the same book, the ‘New Hacker’s Dictionary’. It seems for them that everything about hackers had been documented in this book. Then I had to tell them that, ‘Yes, I’ve read the book, which is a kind of compulsory homework before entering the research field. But I’d still like to know how you think of yourself; I’d still like to hear your voice.’ As an email dialogue between me and one of my email interviewees shows below, very often there did exist a huge perceptive gap between social scientists and computer scientists. Perhaps this is the main reason why it is important for social scientists to go into the field of the FLOSS community to understand such a unique innovation pattern.

My identity as a female Taiwanese also brings me a different experience of collecting data. This personal experience derived from my experience of an essential difference of a face-to-face interview and an email interview. In a male-dominated software world, conducting face-to-face interviews allowed me to get more data than when I stayed anonymous on the Internet when adopting virtual methods. In a face-to-face interview, on the one hand, I seemed to get more detailed explanations from interviewees because I was regarded as both a naïve to the field and a woman. On the other hand, without knowing my gender, the E-mail interviewee was both more open and blunt compared to respondents during face-to-face interviews.

### **3.6 Conclusion**

In this chapter, I explain the research framework based on SWT. I argue that SWT enables me to depict the heterogeneous and dynamic FLOSS innovation system as a social world surrounded by and contained with soft boundaries within and between the social world itself. This theory shares some similarities with the actor-network theory, but differs in the sense that the permeable boundaries, which might be considered as a weak factor in the ANT, serve as a positive element in the FLOSS innovation process to allow actors to move around, and to exchange innovation resources more fluidly. The social worlds theory thus helps catch the complexity and dynamics of cross-boundary activities. I also introduce a range of qualitative research methods employed in this research including interview,

## Chapter 3

---

ethnographical observation, documentary analysis and case studies, both in the virtual and real environments, and the data gathered. In the following chapters, I will explore the research issues raised in Chapter 1 in great details by studying the heterogeneity, various cross-boundary activities, and the dynamics of innovation process in the FLOSS social world.

## **Chapter 4 Hacking, Heterogeneity and Cross-boundary Practices**

### **4.0 Introduction**

This chapter brings my detailed examination of the social world of hacking and what I refer to as ‘pan-hacking’ supported by evidence from my fieldwork. As noted in the previous chapter, I want to investigate the socio-technical relationships and structures built on a constellation of practices found within and outside the hacker social world. In light of the fieldwork reported below, I argue that “hackers” and “hacking” are given many different meanings by different social actors engaging in a range of activities, which I want to suggest are best seen as a constellation of hacking practices. In the hacker social world, one can see the meaning of ‘hacking’ is described, inscribed in, confronted, challenged and negotiated along with participants’ activities. The hacker subculture, if there is one, is not predetermined or prescribed. Instead, it is constructed by diverse actors moving between different social worlds and embedded in various practices, some of which are collective. Moreover, participants’ identity as hackers is not simply a self-description or ascription. The identity is inscribed in their material practices, which are adopted, translated into, and found in diverse social worlds. Various FLOSS projects that are co-developed both in the hacker social world and the mainstream social world (e.g. the state, the corporation, and the public sector) serve as boundary objects that bring the two social worlds into connection, and moreover enable a degree of cross boundary interaction and integration to occur. In investigating the interaction between the two social worlds, one can see how the technologies in regard to “hacking” (in a sociological sense) are socially and technically constructed, and how marginalized hacking technologies are incorporated into the development of ICTs more widely.

In the first section, the simplified but prevalent view of the hacker subculture is challenged in constituting the hacker community under the framework of a social world. In the second section, the fieldwork is drawn on to explain the

heterogeneity of the hacker social world and the recalcitrant meaning of ‘hackers’. But the data also show that there is a range of practices shared in this social world that can be generalised as a constellation of hacking practices. Based on this constellation of hacking practices, the hacker social world is characterised. The social meanings of these collective practices inscribed in actors’ interpretation, coordination, confrontation, and negotiation within and outside the hacker social world are examined in the third section. The chapter concludes that within the social world, no single membership can be defined. The identity of actors is highly hybrid and fluid given the frequent cross-boundary activities. Some exemplars of this process appearing in formal and informal documentation are also provided.

#### **4.1 The world of hackers: A community of practices**

A community is a locale where human beings gather, interact, form their identity and share a broad social-historic structure. The community through which people interpret and understand social relations has been a key area of interest within sociology. Ever since the early days of social theory, defining basic types of social configuration and analysing the source of their cohesion and boundaries have been central concerns. How people identify with others and their senses of belonging to social groups continues to be the subject of much academic and popular debate in social science. Following on from the sociological analysis of social classes (Marx 1867), societies and communities (Tönnies 1887), groups formed through mechanical solidarity based on similarity, versus organic solidarity based on complementarity, occupational groups (Durkheim 1893), open and closed groups, interest groups (Weber 1922), a “community” was defined in terms of social group sharing a specific place, common ties and regular social relationships (Wenger 1998: 283, n. 8; Bilton et al. 2002). For example, UK research on the working class in the 1950s argued that the workers such as miners, steel-makers, ship-builders or textile workers lived and worked together, and they shared the common experience of adversity and subordination (e.g. Johnson 1971; Kornblum 1974). In light of these factors, the working class formed a strong sense of community with a specific place, common ties and social interaction. A strong sense of identity also emerged out of the distinction between insider/outsider, worker/employers, blue/white collar. However, later studies challenged such a

homogeneous view of the working class and pointed out a diversity based on various social resources held by different groups (Lanzara & Morner 2003).

The work of Ferdinand Tönnies (1855-1936) explored the impact of urbanisation on the character of social contacts. Tönnies argued that there is ‘a shift from *Gemeinschaft* or community, characterised by close-knit, personal and stable relationships between friends and neighbours and based on a clear understanding of social position, to *Gesellschaft* or association, based on transitory, instrumental relationships that were specific to a particular setting and purpose and, in this sense, did not involve the whole person.’ (Bilton *et al.* 2002: 38). The growing complexity and diversity of modern society have impelled scholars to develop a more flexible and useful approach to analyse the dynamics of social structure and relations at both global and local levels. Over time, the notion of a community has been interrogated by postmodernism, where diversity, individuality, fragmentation and detachment are celebrated. The conventional concept of community is consequently questioned. Fluid memberships, blurred boundaries and multiple identities become common in late modern communities, without being ‘rooted [in the] soil it occupies’ (Driskell & Lyon 2002: 375). Many late modern groups, particularly in cyberspace, are found in loosely coupled and self-selecting and networks (e.g. Wellman 1999). New notions of communities such as virtual communities and a global village are generated along with the advent of ICTs. As much research has shown, ‘the concept of community has been continually redefined and remains extraordinarily slippery’ (Driskell & Lyon 2002: 375).

The hacker “community” shares many postmodern features of the form and character discussed more generally elsewhere by post-structuralist scholars such as Derrida (1978), Lyotard (1984) and Baudrillard (1988). The complex process of interpretation and negotiation does not enable ‘hacking’ to have a stable and self-evident content and meaning. As one shall see later, the value of hacking and what shall be defined as hacker ethics are constantly negotiated through various claim-makings of participants coming from different social worlds. Their expressions and performances are embedded and embodied in the problem-solving process as their everyday programming practices, as will be analysed in chapter 5 and 6.

In light of my visits to various UK hacker meetings, I observed that the membership of the hacker community is highly mobile. These hacker meetings were virtually organised by 2600.org but take place physically around the world on the first Friday evening of every month. It was Dutch hacker Tineke's second visit to the London hacker meeting<sup>xvii</sup> when I met him at the Trocadero shopping centre, where the participants in London usually gathered. He told me that he had come to the meeting during his 6-months stay in London for a part-time job as a network security consultant. After that, he would go back to the Netherlands. For Tineke, the London hacker meeting was a temporary sojourn where he could find people sharing the same interest as he. He might not know whom he was going to meet on that night; all he knew was that those people there were interested in computer and telecommunication security and the impact of technology on society. Reports over the last nine years have shown that participants at London hacker meetings were 'from both sides of the fence, no matter what age or level of skill and experience', including 'computer hackers, phone phreakers, cyberpunks, performance artists, systems administrators, cybergoths, military intelligence officers, mobi chippers, skip trashers, hacktivists, network gurus, anti-virus programmers, penetration testers, multimedia artists, internet entrepreneurs, newbies, cybercriminals, warez d00dz, old skool, movie script writers, 31337, civil liberties activists, lawyers, radio hams, students, cool hunters, wannabes, djs, corporate security professionals, academic researchers, privacy campaigners, journalists' (<http://www.london2600.org.uk>).

The meeting had an informal agenda. The background of participants is diverse, and so are the topics. Some people have regularly appeared at meetings while occasional attendance is common, as in Tineke's case. Apart from digital gadgets, participants sometimes bring 'toys' unrelated to ICTs, such as YoYo balls or freewheel to play with. The machine is just part of their activities, not the entire world. The hacker meeting serves as a platform where people sharing the same interest interact with each other. The atmosphere is so relaxing and informal that you can do whatever you want. As described on its 2600.org webpage, 'Meetings exist as a forum for all interested in technology to meet and talk about events in technology-land, learn, and teach, [for] anyone of any age or level of expertise.' (<http://www.2600.org/meetings>). What I observed at the London2600 meeting,

however, does not completely mirror other 2600 meetings in UK. Meetings at different places are seasoned with a local flavour. For example, participants at the Hull2600 meeting are mostly linked to the university of Hull (the meeting venue is right opposite the university). Apart from the common interest in computer technology, they share a collective link to the university life and local events. As a result, I have seen couples dating at the meeting, local people (university students or staffs) coming to the table to chat, alumni regularly gathering to see each other. No identical environment can be found among meetings taking place in different cities. Each city meeting such as that at London, Leeds and Manchester, has its own character. Speaking of the local character, what is written on the London2600 webpage may serve as an illustration: ‘We are mostly British and therefore somewhat shy in public, but it is easy to strike up a conversation with most of us.’ This statement, perhaps tongue-in-cheek about their ‘British’ and ‘reserved’ character, does nevertheless send out a local signal as to the character of the event. Local traits such as this are particularly in evidence at meetings of local Linux User Groups (LUGs), as I discuss in chapter 7.

The cultural, social, and technical diversity found at the 2600 meetings should not come as a surprise. Given the heterogeneity of the hacker social world we might expect this. The term ‘hacker’ has been contested (Taylor 1993, 1999; Thomas 2002; Skibell 2002) for decades in various discourses. As has been argued, ‘the term “hacker” has its own historical trajectory, meaning different things to different generations’ (Thomas 2002: ix). “[H]acker” has been stretched and applied to so many different groups of people that it has become impossible to say precisely what a hacker is. Even hackers themselves have trouble coming up with a definition that is satisfactory, usually falling back on broad generalizations about knowledge, curiosity, and the desire to grasp how things work.’ (ibid.: 5). Despite being aware of the ambiguous nature of the term ‘hacker’, Taylor confines his research to the territory of computer security at a certain time and place, and examines the relationships between the computer underground where ‘hackers’ reside, and the computer security industry where a mainstream ICT culture dominates. In emphasising and focusing on this antithesis, his research object, hackers, is inevitably allocated to a deviant position in society. The dichotomy he draws between the computer underground and the computer security industry locates hackers on the dark side of the development of computer science. Taylor’s

account fails to avoid stigmatising hackers; instead, it offers another rather reductive account of hackers as “outsiders”.

Thomas, subsequently, gave this overly reductionist view on hackers a postmodern twist. He argues that ‘hackers both adopt and alter the popular image of the computer underground and, in so doing, position themselves as ambivalent and often undecidable figures within the discourse of technology.’ (Thomas 2002: xx). Being aware of the complexity of the hacker field, he considers hacking from various aspects, biographically (a dichotomy of 50s-60s old-school and 80s-90s new-school hackers), socially (e.g. social positions, social roles), technically (e.g. security, game, kernel programming) and culturally (e.g. cyberpunk and social engineers). Thomas’ strong belief in the historical categorisation of hackers<sup>xviii</sup>, however, leads him to position hackers in the 90s, in the context of network security, as Taylor and others do. For the purpose of his research, he gives hackers a broad definition: “a group of computer enthusiasts who operate in a space and manner that can be rightly defined by a sense of boundless curiosity and a desire to know how things work, but with the understanding that such knowledge is further defined by a broader cultural notion: secrecy” (ibid.: 3). Despite Thomas’ view being more open to the diversity of the hacker world, the term “subculture” he also ascribes to hackers suggests a tight, coherent social group. In treating the hacker culture as an already existing entity referring to a solid and stable reality, ‘the concept of subculture tends to exclude from consideration the large area of commonality between subcultures, however defined, and implies a determinate and often deviant relationship to a national dominant culture’ (Jenkins 1983: 41). The concept of “subculture”, as I argued in chapter 3, is essentially flawed due to its attempt to impose a hermeneutic seal around the relationship between hacking practices and participants in the hacker social world. As Bennett (1999) argues elsewhere, “[T]he term “subculture” is deeply problematic in that it imposes rigid lines of division over forms of sociation which may, in effect, be rather more fleeting, and in many cases arbitrary, than the concept of subculture, with its connotations of coherency and solidarity, allows for.” (Bennett 1999: 603). The contemporary hacker community, in fact, is an assemblage of practices—techniques and tools, political rationales, expert discourses, user customs—that constitute ‘its’ manifold components in diverse contexts, as we shall see in the

following sections. In my view, if there exists a subculture in the hacker community, what it maintains is a postmodern lifestyle in which notions of identity are constructed rather than given, and fluid rather than fixed. The hacker community in fact, is built around a constellation of practices shared by diverse actors. In other words, “hacker” is de facto an umbrella term which enables a wide range of actors coming from diverse social worlds (including teens, college students, programmers, scientists, free speech advocates etc.) to take a part in a common platform built around the shared practices/interests among these actors. Thomas is right to argue that ‘hackers and hacking are much more about a set of social and cultural relations’ and ‘hackers cannot be understood solely in terms of the technology with which they are intertwined’ (ibid.: 4). But while studying the cultural and relational forces that define the contexts in which hacking takes place is essential, one should not overlook the manner in which hacking is done, the tools used, and the strategies that actors deploy. It is these material practices that hacking is grounded in and that are found adopted, converted and integrated within the hacker social world. Hacking is a dynamic and complex process, and the practices are the crucial site for my research if I am to understand how actors utilise these material resources in the specific setting of a postmodern ‘community’.

It is within this context that the notion ‘a community of practice’ (Lave and Wenger 1991) is drawn on to characterise the hacker world, a highly complex and dynamic field, and to investigate the material practices grounded in the field, instead of categorising and presuming a rather tight subculture of hackers. ‘The concept of community of practice focuses on what people do together and on the cultural resources they produce in the process’ (Wenger 1998: 283, n. 8). This does not mean we will ignore the ideologies of and attitudes towards “hacking” in the community. Rather we are trying to find out how a constellation of practices existing in the software innovation system (e.g. compiling the kernel of an operating system, writing patches for programmes, reporting bugs etc.) forms a specific social world. Hacking is not an activity that has a single or stable meaning. Instead, it involves a range of practices and can be found in different contexts. Those who engage in some or all of these practices also very usually occupy diverse social worlds at the same time. The concept of ‘hacker’ therefore is an analytical notion that points to these hacking practices, and that is expressed

through and embodied in real-time social action.

While there may be circumstances where hacking practices come together and help engender strong cultural networks that shape and inform hacking itself, the boundaries of these networks are much more permeable and mutable than subcultural theory would allow. In fact, rather than seek a single definition of ‘the hacker’ it is more appropriate to examine hacking-type practices as they are found *within* and *outside* of the conventional world of computing. Among diverse practices found in the hacker community, it is notable that some of them are also found in the broader software innovation system. For example, potential attackers and system managers use the same tools to execute a penetration test of a system. The security tool they deploy and share, usually FLOSS that has been tested in hostile ICT environments, becomes a boundary object that in some settings, as I will show, enables diverse actors from different social worlds to come together (see section 4.3 below). Apart from the increasing awareness of the security issue, many programmers nowadays believe that sharing source code, the hackers’ rule of thumb, can itself facilitate software innovation. In this context, the open source code also becomes a boundary object in the software industry. It is not then a coincidence to find the hacking practices being adopted in the broader software innovation system. The research thus endeavours to understand the process through which specific practices emerge out of a diversity of practices and grow to be collaborative (see section 4.3 below). This points to the socio-technical construction of software technologies where “hacking practices” originally marginalized, are taken up. Consequently, I will argue the translation of hacking practices by wider ICT players has shaped the software innovation process.

I now want to look more closely at the notion of a community of practice that I discussed briefly above, hereby reporting on my material and data gathered at the HAL 2001 hacker conference held in the Netherlands.

### **4.2 Hackfest**

Since the first hacker conference Hacking at the End of the Universe (HEU)

held in 1993<sup>xix</sup> in the Netherlands, it has been the European hackers' tradition to have regular festivals combining hacking and technology outdoors with camping, barbecues or hiking and so on. Well-known European hacker events in the last decade have included HEU 1993 in the Netherlands, Hacker In Progress (HIP)<sup>xx</sup> 1997 in Amsterdam (cf. Savage 1997), CCC (Chaos Communication Camp, organised by Berlin-based CCC, Chaos Computer Club) 1999 in Alalandsberg Germany, Hacker At Large (HAL) 2001 Camp<sup>xxi</sup> in Enschede the Netherlands, and the CCC 2003 in Alalandsberg Germany. Parallel to these strongly informal hacker events are the various Linux or free/open source software conferences in Europe<sup>xxii</sup>. These events, unlike many industrial or academic conferences, have been organised with local customs in mind and so each has its own characteristics. For example, when German hackers organised the first CCC (Chaos Communication Camp) in 1999, it was seen to contrast with the strongly Dutch event, HIP 1997 that took place outside Amsterdam two summers previously. As Ine Poppe, a Dutch documentary filmmaker and artist who worked HIP as a journalist, told Wired News, an on-line newspaper, the first CCC was regarded as very German:

For me, [CCC] is more German than HIP. They learned a lot from the festivals before. From my point of view, HIP had more of a scene of chaos: tents close together and cables all over the place and dance parties into the night. Maybe we will have those later.

(Wired News August 07, 1999)

Perhaps through the help of some Dutch hackers who had organised the previous Dutch event, the Germans had sought to learn from their experiences. However, even so, a typical German ethos was apparent in the organisation of the CCC festival. As Stephanie, a 22-year-old from Amsterdam who works as a consultant on network security, told Wired News during her visit in CCC 1999, she found HIP 1997 more informal and less programmatic: 'HIP was more happening. It's cool to camp, but at HIP there were more interesting people and more diverse people.' (Wired News 09 August 1999).

Another comparison has been drawn between the European hacker community

and the American one. As Andy Muller-Maguhn, one of the German CCC organisers, puts it:

The American hacker community is organized very differently than ours. I find it strange. Some [US] groups are very political. Some are very technical. I have the feeling there is a very little in common between them. I don't even think they like each other. In Europe we try to be both. We consult with politicians on censoring and so forth, and of course we are in a way a public institution. We try to provide information, freedom, and transparency of technology.

(Wired News 5 August 1999)

Muller-Maguhn's account suggests a socio-technical strategy informs the European hacker events where diverse technologies are celebrated to draw the public's attention to a specific political philosophy on which the hacker community is built. As Muller-Maguhn went on to say:

[I]t's important to give the normal people—and also politicians and journalists—and understanding of how the tools work. In America, more people have email, yes, but technology is driven by big corporations that think about profit and things like customer profiling. For us it's more important to give all groups an understanding of how computers and networks work. Compared to the US, the European public has very critical discussions about technology. Maybe that's one reason why technology is not integrated so rapidly. [European] people are not as careless as in the United States. They ask, 'What if?' They think about *1984* and *Big Brother*. That's always on our minds, so we don't have computers that can be switched to fascist mode.

(ibid.)

As John Gilmour, a lobbyist for open cryptography code which he and others regard as being under the protection of US First Amendment, claimed in Wired News:

## Chapter 4

---

I see the camp as basically educating people about what their rights are. It's not about breaking anything, because the people who break something and make people pay, asking for ransoms or something, those people are outcasts. Or that's how it is in Germany anyway.

(Wired News 9<sup>th</sup> August 1999)

Some of my respondents at HAL2001 who had travelled to the American hacker conference DefCon beforehand offered me similar comments. For example, Tenyen from England, told me that he preferred HAL to DefCon because HAL was 'more alternative' and it was something 'more like a hack'. Rudolf, a masters student at the University of Twente majoring in public relations also thought HAL had a more friendly and relaxed atmosphere than DefCon:

People are very nice and kind to each other. They chat and relax in the nature, not just stay in an air-conditioned hotel, sit in front of computers and eat junk food. Hacking does not only mean intruding computer systems, but more about creation and challenging. And not only computer, you can hack for food as well.

(RVDB070801)

Unlike the American hacker conferences such as DefCon that are normally held in hotels, European hackers choose to have "warm-beer-fuelled mud fests" to bring technology to life, to enjoy a hybrid lifestyle combining the natural and the artificial. It seems that what the participants search for is not an advanced technology per se, but the extra something that "hacking" can bring. In a sense, such an open-aired, free-for-all event suggests the "coming-out" of "underground" hackers. Taking off their "hats", participants seem quite happy to be open about their practice. Driven by curiosity, journalists travel to hackfests, meet hackers in person, and give reports on what they have observed rather than partial stories based on common prejudice. As we will see in the following subsection, such events demonstrate the socio-technical diversity celebrated in the hacker community, and provide a channel for hackers to communicate and interact with each other as well as with the lay public.

### 4.2.1 HAL2001

This next section describes the campsite of HAL2001, where I conducted my pilot fieldwork. This enabled me to get a good idea of how the hacker community of practice is built and what participation within it entails.

Sharing the same name with the powerful computer in the film ‘2001 A Space Odyssey’ that was taken from each preceding letter of the computer manufacturing giant IBM, HAL (Hacker At Large) 2001, a three-day open-air event, took place on 10-12 August 2001 at the University of Twente in the Netherlands. About 3000 campers brought their tents and computer equipment to Twente. The campsite was provided with electricity, an Ethernet connection with 1 Gb of bandwidth coming almost all the way to the participants’ tents, and wireless facilities. Over 2 km of fibre and 15 km UTP<sup>xxiii</sup> supported the event. Inside the 300-foot main tent, long tables held hundreds of PCs and laptops brought by participants to connect to CAMPnet, which supported 2000 hosts and carried an aggregate Internet bandwidth of 1Gb bit/sec. This is a self-contained campsite. In this tented city, participants exchanged information, swapped security tools, ate waffles, viewed fire-eating demonstrations, discussed encryption politics and went swimming in the swimming pool. The campsite was divided into 3 areas including a silent camp where noise was not allowed. The local facilities also included basic requirements such as showering, a launderette, banking, postal services and shopping. Apart from accepting the Dutch gilder, HAL had its own currency (scratchcards) for purchasing food and drinks at the campsite. HAL was not only seen as a fun and relaxing holiday for hackers, but also an intellectual energy station. Seminars and workshops containing various topics were running day and night for 3 days either indoors in the lecture room or outdoors on the grass, depending on the speakers’ interest<sup>xxiv</sup>. To make all this work, volunteers were crucial to complement the few hired people who worked full-time to make HAL a success. The volunteers were teamed up: the catering team in charge of food and drinks, the security team taking care of campsite security, the power team setting up cabling and light facilities, the info desk providing a 24 hours inquiry service, the stage management team making sure the speakers got equipments and locations, the tent builders building up the main tents, the entrance and cashiers team in charge of registration, the networking

crew designing, building and maintaining the network, and many more including the most rewarding but unwanted work of all teams – the sanitary team making sure participants could go to the toilets without fainting from the smell or sight. This division of labour among the volunteers was seen as important. It was interesting to see that female volunteers usually ended up in the catering team, info desk, or cashier teams, while male volunteers were involved in building the actual information infrastructure, either the hardware or software.

The participants were mainly European. But I wondered whether these 3000 participants were all hackers? This was difficult to determine because they all carried different titles: students, security consultants, system managers, self-employed programmers, academics, CEOs, members of the US air-force, snack vendors, journalists (the journalist from the Frankfurter Allgemeine Zeitung newspaper) and even the police. Some showed up as families or with boy/girlfriends (many ‘hackers’ brought their children or even babies along), while others were professionals. To ‘protect’ participants, everyone at HAL was asked to wear a wrist strap classified in different colours as a sign of registration. Normal visitors wore light green wrist straps; volunteers in fuchsia; reporters/press in light-blue; the police in bright fluorescent-red (but the organiser gave a kind warning that the police didn’t always wear red wrist straps). Though the event was open for all, this classification scheme subtly showed some differentiation within the community. The hacker stereotype of the white single male was quickly revised when seeing many hacker families on the campsite with nursery equipments such as swings, slides, or hobby horses around. A crèche was even suggested to add to the next hacker event. The demography of the HAL meeting shows the diverse composition of its participants, which would, I expected, shape the sort of interaction I would be likely to find.

### **4.2.2 Lifestyle: social diversity**

So, what did the participants do at HAL? Unlike a traditional conference, HAL participants had a very wide range of choices and considerable flexibility in regard to their daily activities. Topical conference sessions or workshops ran intensively from 10.30 to late evening in 5 lecture halls simultaneously, meeting the diverse

intellectual needs of the participants. If the sessions were not of interest, participants could always find ways to enjoy themselves. Some sat in front of their PCs in the tent hall while others did so outside their tents in the sun. Sporty participants played football or went swimming while music lovers played guitar and listened to music in their tents. Or one could have a quiet sun bath (the sunshine however was limited because of poor weather during those 3 days). In the evening, the aroma of BBQs permeated the campsite. Friends had food and drinks together. Talent shows took place by the campfire. A tent-cinema was also provided. Crowds gathered in the bar. A Welsh visitor described his stay in HAL as follows: ‘Well, you know, [I’m here] just to meet people, listen to some gossips, stories and enjoy the atmosphere’ (HAL0801). Indeed, HAL, like other hackfests, appeared to be an extension of existing virtual communities. Many participants mentioned that they had known each other from mailing lists. When they met up, they introduced themselves by their alias. The social extension from the virtual world is important. As the German CCC member Andy Muller-Maguhn has said:

It’s one thing to be on a mailing list. But sitting in a campground and having discussions all night gets people networked more closely together and helps develop solutions to problems we haven’t even faced yet, while computer companies sell solutions to problems we wouldn’t have without them.

(ComputerWorld 10 August 1999)

Interviewees I brought together in a semi-structured focus group at HAL also reinforced the view that the social activities were what they looked forward to the most. Here is an extract of my conversation with Stefan and Astrid:

Y: [W]hat do you expect from this conference?

S: Sunshine. (all laughed) A lot of fun. I think for many people this is the place to meet each other, because many people only talk on line. Some groups here are close groups but they live in different countries, different cities or whatever.

Y: So they know each other before?

S: Yeah.

## Chapter 4

---

A: Yeah, but they only know each other as still never growing possible dot on the screen.

S: So this is the place to meet them to have fun and exchange ID and listen to other people.

(GI090801)

The social function of HAL, parallel to the 2600 meetings mentioned earlier, serves as a platform for people sharing the same interests to meet up physically and enhance the trust built previously by their virtual contacts. But sometimes it went further. RGB, a hacker who engaged in many intrusive actions in his teens, now an estate agent, described the hacker parties he attended:

We have parties where you meet all the people you used to hack with, and very interesting. That's not all about technology, most of the time it's just a party, hanging out, dancing, drinking. It happened to be just a lot of people hacking away on computers on Friday night and in small steps it changes into more social things about parties. Artists and all kinds of people are coming. So it went more into a bigger thing and eventually that's what it ended it up here (HAL).

(RGB100801)

Such a social event for hackers is also an ideal place to meet girls, as RGB explained that

My girlfriend is not into hacking, that kind of thing, but she came to the party a few times. So that's more a social event. That's better for her. She doesn't like computer so much. She uses it for the Internet, for the email, everything, but that's it.

(RGB100801)

Later, the sleepless night for hackers started. The tent hall was crammed full of people. Loud digital music could be heard in the 100m<sup>2</sup> area until 4am. When most people were having fun, volunteers on the security and info desk crews however,

had to maintain their shifts during the late night to make sure this self-contained hacker city went well. There was a practical reason for this, which was also unique compared to other conferences. As Jaco, the Dutch hacker in charge of the HAL cashier team, commented during my interview: ‘One of the things making [HAL] different [is] by not hiring security guys but keeping it grey suits, low profile itself. It's really limited budget, we are not talking about millions of guilders.’

(JK100801). Rop Gonggrikjp, who published the first Dutch hacker magazine and was the organiser for previous hackfests, also noted the efforts of the volunteers from the participants: ‘There is no professional security attendant because there is no budget for it. But because of people around like you and me help with the organisation, we have made it all.’ Jaco agreed with Rop that things had worked well: ‘I think it's very good for the atmosphere here. If there are people in uniform walking around here for security, this would not be a nice festival I think.’

(JK100801). But why would these people volunteer to do these challenging jobs and how does the volunteer network operate? I suspect such a volunteer phenomenon imply that there exists some the intrinsic organisation to the hacker community. And this volunteer phenomenon relate to either the Internet gift culture or the free software gift culture.

### **4.2.3 Intellectual diversity**

What kind of conference programme could meet the diverse intellectual and technical needs of a wide range of enthusiastic computer users? There was a programme committee responsible for inviting speakers, collecting and reviewing proposals, and creating the final schedule (through peer review). The conference programme was based on four themes: privacy & security, public understanding, digital rights and content encryption, and ‘weird science’. And the main idea of the conference was to show that ‘We are not living in a safe virtual world’, as Gerrit Hiddink, the event coordinator put it. The topics included the most up-to-date techniques, social and legal issues, panel discussions and practical workshops where the lay participants could learn basic ‘hacking’ skills. Unlike some meetings held by very restricted hacker groups, which lay audiences were excluded from, since the very specialist topics they handled could not be understood without expert knowledge, the HAL programme sought to close the gap between the lay

and the expert.

The keynote speaker was Emmanuel Goldstein, the editor of the 2600 hacker magazine. He criticised the US law against free access to information such as the Digital Millennium Copyright Act (DMCA), which even forbids individuals to make a backup copy of their own CD. He claimed that citizen's rights had been eroded. With the DMCA, the industry determines what digital material is allowed to be viewed and copied, where one is allowed to view or copy (using regional management), and on what devices one is allowed to view such material. Goldstein drew attention to a similar law, which was to be introduced in Europe. He was especially concerned that 'DMCA serves as an example to many other countries, and it can be introduced through out the globe.' Following on from Goldstein's presentation was a talk by Tom Vogt on DeCSS--one of the most controversial of programmes, that was and still is banned in the US, and panel discussions on the methods and effects of the circumvention and anti-circumvention of security. Users' privacy and network security was another concern. There were many detailed talks about hacking techniques and security measures, ranging from a panel about DDoS (Denying Distributed Denial of Service Attacks) to a workshop demonstrating DIY Linux Security. Mobile security, PGP, digital signature, voice cryptography, biometrics all had a place in the programme. Additionally, an Orwellian concern was evident in the so-called 'International Big Brother Awards' and a session of camera and video surveillance. Instead of teaching audiences how to exploit the vulnerabilities of software to conduct cybercrime, most sessions seemed to encourage audiences to be aware of the risks in the current computer world. Even in sessions such as 'Hacking digital watermarks' or 'Writing exploit payload for Risc based Architectures', the speakers appeared to be more keen to share their experiments and experiences rather than to promote any malicious form of hacking activity.

Apart from these technical sessions, a small part of the programme covered social, economic, legal and psychological issues, such as panels on cybercrime, hacker ethics, cyberactions, drugs and thought crime, hacking motives, and a non-proprietary economic scheme. Stories about hacking, such as the 'B92' project that used the technology against political repression, or the population project of the

hacker island Sealand/Havenco, or the history of the Dutch hacking magazine *Klaphek Magazine* were told. Some special panels offered very particular types of events, including entertainments such as a Hacking Sound performance (concert), a Hacker Quiz, a Hacker cinema showing a documentary film about the ‘Free Kevin’ movement that recounted how the *2600.org* team endeavoured to get the hacker Kevin Mitnick released, and a very special memorial session dedicated to the German hacker Wau Holland who found the Chaos Computer Club and devoted himself to advocating freedom of information.

Shared a similar ambience as previous events (see e.g. Wired News 10 August 1999), the conference theme of HAL 2001 also reflected a cultural movement concerned with data security, privacy and the free flow of information. However, there was a more hidden conference stream, which related to FLOSS. In fact, it was understandable that the name of FLOSS had not been raised explicitly in the programme. FLOSS was considered as part of the hacker community. FLOSS artefacts were so much part of the hackers’ social world that there seemed no need to make specific reference to them. Though it was said that the Windows system was more vulnerable and had more security holes than any other operating system, it would have been bizarre to see a session on how to secure the Windows system at the hacker conference. The Windows trademark hardly made an appearance at the campsite except for being decorated on the tent of the anti-Microsoft groups with a bold red cross on it. In contrast, the penguin ‘Tux’ representing Linux was the most popular symbol on the campsite. When IT security consultants came to the hacker conference, they had expected to find open source security tools that had been tested in hostile environments, rather than other proprietary tools. While sessions such as ‘Linux 2.4 firewalling’ and ‘Open BSD—overview from a security point of view’ presented the problems and solutions surrounding the systems, workshops such as ‘DIY Linux security’ and ‘FreeBSD security’ taught users how to implement the systems securely step by step. ‘Fun & Games with FreeBSD & Solaris kernel modules’ was particularly interesting because it demonstrated how hacking could be played with FreeBSD and Solaris kernels. Apart from these, the development of FLOSS was still of central concern in the hacker community: ‘The strategy of software quality in OS/GPL systems’, ‘Introduction to the FreeBSD operating system’, ‘Technical Introduction into the GNU/Hurd Operating system’, ‘Demonstrating Hercules running Linux/390

(Debian)', 'GNU Radio, a free software defined radio' 'Designing an economy without intellectual property' – all indicate the commitment to FLOSS.

The openness of this programme reflects, I suggests, the intellectual heterogeneity of the hacker community of practice both technologically and socially. The HAL talks were unconventional: as Jaco observed,

[U]sually a conference is in a stupid building, which is large glass tower. You pay 2000 pounds for breakfast and listen to some important guys who say he is important. For example if you go to a lecture of Bill Gates it's not interesting, because this guy is so busy, he can only talk briefly about a very few obvious things and here people have time to talk to each other and you really explain something. It's not like some business guy who is reading some obvious speech because all the speeches are the same. So that's how we make it different.

(JK100801)

Because there are diverse tools (programming languages, software) used in the hacker social world, the conference programme was not narrowed down to one single approach or system. This suggested that FLOSS software engineering contains various ways to meet design goals and user requirements; users have diverse options to choose which technique meets their requirements the best. Most of the tools recognised are FLOSS-based. As the next chapter will explain, FLOSS is regarded as a more flexible, context-sensitive system. Practically speaking, open source code allows the possibility of different designs to exist apart from a proprietary one. Furthermore, given the vulnerability of software, open source code allows a more effective debugging process enrolling a large number of users to detect security holes and write patch programmes, which creates safer and more useful software. But as will also be shown later, the situation is more complex than that. The meaning of FLOSS is constantly being interpreted and negotiated within and across the hacker social world, rather than being defined according to some established consensus.

#### 4.2.4 Law and order: Hacking vs. Cracking

Imagine this: A teenage white boy sits in front of PCs with wires surrounded by many printouts of passwords and IP addresses, pizza and coke cans on the ground, restlessly typing on the keyboard to get unauthorised access to systems. If one expected to see this at HAL, one would be disappointed. It was clear that there were informal proscriptions in place at HAL: so-called ‘script kiddies’ who use scripts to effect unauthorised access to systems were not welcome here. In the Acceptable Use Policy (AUP), which outlined the principles of use of the internet connectivity available during HAL, it was written that ‘The use of any program/script/command, or sending messages of any kind, designed to interfere with a user’s session, by any means, locally or over the Internet, is prohibited’. It was made clear that:

Any activity which adversely affects the ability of people or systems (on HAL or anywhere else) to use the services provided by HAL is prohibited. This includes, but is not limited to, “Denial of Service” attacks against any servers, network hosts or individual users, be it on HAL or anywhere else. Attempting to circumvent user authentication or security of any host, network, or account (“cracking”) is strictly prohibited. This includes, but is not limited to, accessing data not intended for the participant, logging into a server or account the participant is not expressly authorized to access, or probing the security of HAL servers and networks. Machines designated by HAL as targets for a hacking contest are exempt from this rule.

(Acceptable Use Policy HAL2001)

What happened if one was caught infringing the AUP? HAL claimed that they had some of the worlds’ best system administrators monitoring the Internet access to HAL services, as part of the normal course of its network operation. ‘Should HAL discover a participant engaged in prohibited actions as outlined in AUP or engaged in actions which otherwise adversely affect HAL’s ability to provide services, HAL reserves the right to (temporarily) suspend the participant’s access to the HAL Network.’ (ibid.). Indeed, in one particular case at CCC 1999, a

camper who attempted to attack the network was subject to local justice and found himself cleaning the conference toilets (ComputerWorld 16 August 1999).

Apart from the local provisions in place during HAL, the information disseminated through the HAL2001 systems was subject to Dutch Law as well since the systems were physically located in the Netherlands. According to Dutch Law, it is illegal to gain unauthorized entry, or “crack” into other people’s equipment, either inside or outside the Netherlands. Certain countries also claim jurisdiction even outside their national borders. For example, France claims the right to regulate information on foreign servers; Italy assumes jurisdiction over sites directed to an Italian audience, and the US reserves the right to prosecute offences against American interests, irrespective of where they take place. However, HAL did not recognise this wider range of foreign jurisdiction and only obeyed instructions from the Dutch authorities. Nonetheless, HAL warned participants that as an individual user, one should consider the possibility that s/he could be sued or prosecuted in another country. Additionally, if one was physically in a country other than the Netherlands when disseminating information through the HAL2001 systems, s/he was probably subject to Dutch jurisdiction.

These legal controls however, could only scare “script kiddies” away from hackfests. Eventually, as some interviewees said, “script kiddies” had to grow up. Rop Gonggrijp who wrote a pamphlet called ‘Script Kiddes’ Guide Book to HAL2001’, adopts a position of the elder hacker speaking to the younger generation. He draws attention to the key differences between the older hacker and the script kiddie:

When you arrive at HAL2001 and look around you, you may feel this is an ideal place to do script-kiddie things. ... You may have also noticed all these other people around you. Most of them seem to be in some kind of different world. Most noticeably, they’re not constantly bragging about how many machines they have installed Stacheldraht on. When they talk about computer security you often don’t understand, and they keep talking about vague political things a lot of the time. That’s us. We are the rest of the hacker community. We’ve

been here for a while now, so you would probably just refer to most of us as “these old people”. That’s OK.

(p. 1)

Gonggrijp clearly distinguishes between “us” and “them” in his writing: “us” are the older hacker generation while “them” are the younger script-kiddies; “us” are mature and responsible while “them” are immature and irresponsible; “us” are concerned about the moral issues of computer security while “them” are blind to these ethical issues. Gonggrijp in fact stresses the need for younger hackers to see how their behaviour is likely to damage hacking activity overall, not least by the hostility it creates:

We weren’t all that good when we were kids. But right now, powerful people all over the world would like to paint a picture of HAL2001 as a gathering of dangerous individuals out to destroy. While it may seem cool to have powerful people think of you as dangerous, you’re only serving their purpose if you deface websites from here, or perform the mother of all DDOS attacks. You’re helping the hardliners that say we are no good.

(ibid.: 1-2)

Gonggrijp’s text is clearly one that is assigned to champion a collective position within the hacker social world against those opposed to it whose leadership, he believes, “doesn’t even know how to hold a mouse” (ibid.: 2).

However, such a public display of unanimity would in fact hide the actual diversity I found in the hacker social world. Things are more complex inasmuch as I discovered that the way people at HAL characterised their own behaviour varied considerably. For example, in the hacker’s argot ‘white hat’ and ‘black hat’ are terms given to define who is an ethical hacker and who is not. But many interviewees at HAL told me that these descriptions were too simplistic. Frequently they wore many different hats. As Tom, a German chief system manager, said:

[This “white hat” and “black hat” distinction] is not a perfect thing. There is nothing like the white hat. It's very difficult to say actually. It's putting a label on people, a label it's never entirely correct, but it helps to distinguish various categories and various tones of views on how to act. So it's useful but you can't describe someone just by saying he is a white hat, or he is a black hat.

(TV120801)

Tom thought the real situation was more complicated:

There would be very very few people are actually really white or really black. And all the others will be mostly grey, but more black than white, or more white than black. But I don't think that you can really describe someone's attitude just by saying that he is white hat that's too, it's a high level extraction. It's like 'he is a scientist', which doesn't say much about him but you know roughly what kind of work he does.

(TV120801)

Tom went on to offer a more detailed commentary on his classification of hacker activity:

Firstly, people are not simple. Most people are very complex. The reason why hackers do whatever they do has a lot of reasons. There's curiosity there but of certain kind and very a drive like... I want to know everything about TCP/IP, or other people are driven by how many webpages have I defaced this month. That's also changing over the time because you are always... Hacking is about learning. When you learn, you change. So the more you learn, the more you drive into different direction than you are moving before, then you are taking different things and you re-evaluate what you have done. A year later you are in different things again and you are again moving what you do. It's very dynamic.

(TV120801)

Tom's words, echoing Gonggrijp's notion of generational 'maturation' also see this as a learning process, and one involving 'dynamic' change over time. As many interviewees also noted, they stopped hacking during the period of getting a job or after the college and university.

When being asked to consider their identity in terms of the "white hat" and "black hat" labels, many interviewees put themselves in a more ambiguous position. For example, Rob claimed that 'I am in the middle, "grey hat" as they are called. Well, I guess so.' (RGB100801). But he also denied this demarcation was useful: 'Well, I don't think it's good to say "grey hat" or "black hat" or...I am just a hacker and I've never destroyed anything, so I would consider myself "white hat".' (RGB100801). Though his intrusive behaviour in breaking into systems might be regarded as on the dark side of hacking, Rob justified what he did by saying that he did not destroy or manipulate any data in the systems. Therefore, he suggested that he was more "white" than "black". Stefan was aware of the dual role he played in the hacker world as well. He said that

[As a system manager myself], I fight against hackers because I don't want them to own my own systems. But I also like to hack myself. So I see from two sides, not only against hackers but I am also part of the groups of hackers.

(GI090801)

Stefan also regarded himself as more "white" than "black":

I don't consider myself as a malicious hacker. I think I am a hacker because I am creative, and I can do things with computers [that] people thought that's not possible and things like that. That's the more important hacker spirit I think. I don't like breaking into other people's website. ... It can be very difficult. But I don't learn anything from that.

(GI090801)

Denis, a senior security consultant who used to hack in his college days, claimed that ‘My different hacking activities (in computers, mechanics and lock-picking) could be viewed as an appropriate application of a technology to solve a problem in another technical field.’ (DG120901)

Torsten, who saw himself as a “white hat” hacker, defined hacking as ‘a way of exploring, a desire wanting to go somewhere no one else has been before’ (GI090801). When asked whether he had ever broken into any unauthorised systems, Torsten said he had never done so because that was not what a hacker should do; that was “cracking” rather than hacking. But even though his hands were “clean”, Torsten hesitated to admit that he was a “white hat”. He said: ‘it depends on the term [in which] you define them.’ (GI090801).

What was particularly interesting was his direct association between white hat hacking activity and innovation in software:

The white hat hacker, the good guy, in general [is someone who produces] science and technology, because any contribution to that as in a clever innovation counts as a hack, or most of them count as a hack, or some certain definition of hack. ... I haven't seen a black hat hacker coming up with some true innovation.

(GI090801)

Torsten also said that he broke into his *own* system to challenge it. The same idea informs a hacking contest, ‘Linux Death Match’, at HAL that has also become a tradition in all conferences. Teams of network administrators tried to halt one another’s network services. Teams built up their own systems, and the winning team was the one who could secure their system until the last minute. Some chose not to attack but instead build stronger walls while others chose to use intrusive tactics. Some employed Linux tools while others preferred FreeBSD tools. By having such a contest, participants could express their hacking skills but did no actual harm. At the earlier CCC 1999 event, one participant remarked on the skills

used in such contests:

It's pretty amazing to see the knowledge these people have, all these people sitting here in front of these machines will never have a problem finding a job. Everyone around here knows how useful it is to find vulnerabilities, and most of these people don't destroy systems, don't crack systems, they just look at them.

(ComputerWorld 10 August 1999)

The match once again demonstrated the technical skills (techniques, tactics etc.) and a competitive element to the hacker community.

The material I gathered through observation and interviews at HAL 2001 indicates a number of things: first, that there is a strong interest in positioning hacking as a more positive and creative endeavour; second, that within the hacking social world the binary "white/black hat" distinction is highly provisional; and third, that hacking skills are of use to those in the mainstream ICT world.

Jaco, a co-organiser of the event, regarded hacking as "building something", which was precisely why he gave his time to organise the event: "HAL is about building something. You spend your time and you leave something behind. Hacking [in a cracking sense] is not compatible with my personality anymore, so I am not deeply involved in that." (JK100801).

### **4.2.5 Multiple identities**

As suggested above, diverse actors are found in the field of hacking and my respondents offered different ideas about how to interpret the meaning of hacking. Hacking technology or techniques are employed and shaped by diverse actors technically and socially. This social and technical diversity are co-produced and shaped mutually. That is, the social diversity allows multiple groups and individuals to contribute to technical innovation while the technical diversity engages diverse social actors in the field. Apart from diversity, by which I mean

the engagement of various social actors in a common platform, hybridity or multiplicity is in evidence where a single actor can inhabit various social worlds when s/he engages in the common platform. This is reflected in my fieldwork where most participants felt they wore different hats when they conducted hacking. They claimed that there was no “pure white” or “pure black” hat in the hacker community; most participants were actually “grey hats”. This narrative gives an idea about multiple identities in the hacker community that entails no single identity can be ascribed to hackers.

Even if the competences and interests of hackers locates them as insiders in the computer world, this world is divided into various lines of work and not every hacker is good at everything. Given the constellation of practices that I have suggested define an ideal type of hacker, which are 1) Interest in tackling software problems and resolving them, 2) Writing challenging scripts to explore software vulnerabilities, 3) A strong interest in decryption, code-breaking, 4) Writing creative scripts or codes and sharing them, 5) Developing novel *hardware* and sharing the *proprietary* information on which it is based. (see also chapter 3 and section 4.3 below), few interviewees claimed to be good at all of these. The hardware and software division is particularly clear. A few interviewees who claimed to be in hardware design hardly knew how to program, and vice versa. Most interviewees in the hacker community claimed they were capable of either exploring security holes (#2) or writing codes (#4), while a few of them mentioned their experience in both. This suggests that hackers may be regarded as experts within the computer world in general, but when we look more closely, they have different skills and different interests. Hackers possess a hybrid identity as both insiders and outsiders. But this hybrid identity may shift as their display of knowledge and interaction with other actor shifts. As a result, their identity is not only hybrid or multiple, but also fluid. Here is one example.

Torsten, a system manager, regards himself as an “insider” within the hacking community: “With regard to hacking software and hacking the Internet, I think I am an insider because I know a lot about [these] and because my friends do, we share information and we try things together and say 'hey, look at this, what do you think?’” (GI090801).

An insider is ‘the one who knows lot of things, familiar with technology, being part of a large group of people doing the same thing, for example a lot of my friends share the same attitudes and do the same things.’ (GI090801). In Torsten’s example, though he did not regard himself as an insider of programming, once he learnt the knowledge either on the Internet or through other forms of communication, he then shifted his identity in this specific area from an outsider to an insider. For instance, Torsten states how cross-boundary knowledge changes the identity of an actor from outsider to insider:

When a guy finds a security hole somewhere, he tells his friends about that, and these are acknowledged as insiders of this specific group. Everyone else is still an outsider because he doesn't know about the particular hole. Now, some people in this group will tell their friends and so on eventually someone might use that to break into someone's machine, so at some point it will come out to certain security mailing list, and that point of time, I would count myself an insider because I am subscribing to some of the mailing lists. Now I own the information and I am an insider [of this episode].

(GI090801)

Torsten considers himself as an insider when he knows what is going on in a social group. But he becomes an insider of this episode through getting second-hand or even third-hand information, rather than having an initial affiliation with the core group whose members found the security hole. Torsten’s comment defines the notion of insider/outsider in regard to one’s position in the circulation of peer knowledge, which is likely to vary in different contexts (i.e. who has the peer knowledge and then who is an insider). In short, when knowledge flows in the hacker social world, the identity of members as insider/outside also changes. Participation in the networks is crucial if engagement with a hacking activity is sought, since “the problem”, is defined iteratively as the network kicks in.

Apart from participation in order to establish one’s membership, continuous learning is also important to keep oneself as informed as possible as an insider. “Learning by doing” is regarded as the key principle to understand the sort of ICT

knowledge characteristic of the hacker social world. In an era of knowledge explosion enabled by the Internet, learning by doing is faster and makes new demands on the hacking community. The advent of the Internet facilitates the rigid transmission of knowledge. For example, Torsten had heard that “Hamburg CCC” (a hacker group) was very active and its members made new discoveries about software vulnerability regularly. Residing in Munich, however, meant that he could not physically join a hacker group located in Hamburg. If he did not spend time participating in the Hamburg group, he felt he would remain being an outsider of this specific group. With the aid of the Internet, Torsten subscribed to mailing lists to get information about the group and then got in touch with the group via the Internet. The more diligently he studied their new techniques/technologies, the more he interacted with people in Hamburg (virtually or physically). As a result, Torsten’s virtual activity overcame the barriers of time and space, and eventually he could see himself becoming insider in this Hamburg group despite his being based at Munich. Conversely it was easier for him to drop out the group if he did not keep up with the news generated from time to time. The networks (social or technical; on-line or off-line) provided him with a lot of resources for his own hacking activity. Apart from hybridity, the transmission of and variable engagement with peer knowledge also entails the transition of one’s identity as insider or outsider.

It is appropriate to end this section about heterogeneity with a remark from one of my respondents, Tom. When asked to describe his understanding of what a hacker is, Tom replied as follows:

No, you really can't because they are very very different people. There are really socially responsible people, a lot of them who care much more for the career group than they care for themselves. But there are also a large amount of fame-driven people who want their names on the websites or who want their names on the piece of software or want their name [acknowledged by] the next employer "I just cover this hole". It was good when all these people work together.

(TV120801)

But Tom had observed a common feature about hackers when he claimed:

One thing that all hackers have in common is their interest in finding out how things work, taking things apart, and learning. And learning by doing. Learning by taking it apart, by breaking it. You don't know how strong it is so you break it. I think that's the strong common interest. But it's usually like different things. Some people do it differently than others; some people break into systems, others just analyse it from the radical point of view. It's basically the same tribe [finding] out how things work.

(TV120801)

This notion of the hacker community as a “tribe”, is not used to emphasise a homogeneity but, on the contrary, a heterogeneous and dynamic community. This heterogeneity enables the hacker community to sustain its wide participation and broad range of skills and competences. Many alternative tools and techniques can survive and be converted into other uses across diverse boundaries and networks. Though there are some restrictive hacker groups able to limit membership by publishing their agenda on restricted websites, which are not accessible for users of Windows Internet Explorer<sup>xxv</sup> or require a password, and by having their own ‘rituals’ in a private place, it seemed that HAL characterises the broad philosophy of hacking. Once the technical gap is conquered (arguably, by participation and learning), the barrier of the exclusive group will be dissolved (it is just a matter of time). My data suggests that the hacker community, where participants move in and out without a fixed membership, is a community of practice. The hacking technologies, and their wider application therefore, are in a dynamic and energetic course where actors coming from different social worlds engage and interact via a common platform. Over time, however, the technologies may become institutionalised inasmuch as practices are extended into a wider, more codified or formalised pattern of use, as I shall discuss later in the thesis.

Indeed, it is difficult to give a black and white verdict to hacking practices in

that there are N sides to every question. 'It's hard to tell the difference between a police academy and a terrorist training camp if you don't know the social structure they are in. They both learn target practice and "how do we defeat things that are coming at us?" These are things that are common between the good guys and the bad guys.' (Clarke & Montini 1993: 45). Caezar, parallel to the Gonggrijp's argument cited earlier, tries to distinguish "us" and "them" in the hacker social world. For him, "us" are good guys doing ethical hacking whereas "them" are bad guys hacking to vandalise systems. However, given the fluidity in the hacker social world, it is hard to distinguish a "good" hacker from a "bad" one. As argued earlier, many interviewees position themselves as "grey hat" in their argot, rather than as clearly "white" or clearly "black". Many system managers/administrators also take advantage to play these dual roles (e.g. Stefan). Chris Wysopal, director of research and development for digital security firm @Stake, mentions that 'Ethical hacking means different things to different people. To some, it means hacking for security's sake. For others, it is more hacktivism. Then there is hacking for the pure pursuit of research.' ('Hacking their image' C|NET News.com August 2, 2002). There are disagreements on the word used as well. The term "ethical hacking", to Denis, is better rephrased as "ethical cracking" in that it is actually more factual. But as he notices that 'everybody is using that term [hacking], as for Black, Grey and White Hats Hackers' (DG120803). In such a heterogeneous social world, tensions and negotiations over what one is doing are endemic. Whereas there are hacker groups such as GhettoHackers trying to change the culture from within as well as educate the public at large, hacking in the form of deconstructing software will be likely to prevail. As Tom remarks,

[H]ackers are some of brightest people on the planet ... The best way to understand how something works is taking it apart. A lot of [hackers] are only 15 or 16 years old, hanging around [at HAL], playing games, and running the scripts against some websites. Probably. I don't hope they do that here, but. Well, at 15, I don't think they have enough responsibilities to stop themselves, so they probably do. Someone would be angry that his webpages are defaced. They are not doing anything really constructive, but they are learning. In a few years, they will, well, not all of them, but some of them will be really bright, will really know how things work because they take them

apart. They really uncover it and check it out. Then they will be able to be really constructive, to do things that they couldn't have done otherwise-- cause they have to learn how it works first.

(TV120801)

Indeed, many interviewees admitted their unauthorised hacking actions when in their teens. However, these actions were typically justified in terms of curiosity, creativity or both. Likewise, as seen above, Caezar, a self-described ethical hacker, revealed his phreaking past playing around with a telephone card scheme that let him make unlimited calls without paying a penny (C|NET News.com 2 August 2002). The life course of a hacker appears then to involve a shift of the meaning of as well as the knowledge embedded in, the hacking practice.

### **4.3 The pan-hacking phenomenon**

#### **4.3.1 The hacker social world**

In light of the initial fieldwork reported above, it can be argued that the meaning of hacking is constantly negotiated and the field is socially and technically co-constructed. The former finding confines my earlier argument that we should not look for a hacker culture in the community whereas the latter suggests that the hacker technical culture is dynamic and constantly shaped by diverse actors through the practices and competences they deploy. Diversity is in short endemic in the hacker community. A 'community' in this regard is not a conventional one with a physical place, firm boundary, close relationship and central identity, as defined by scholars ranging from Tönnies to Giddens. The hacker community, instead, is quite loosely defined, and its boundary is malleable and permeable depending on the range of the interests engaged. This kind of community is better characterised as the more flexible notion of 'a social world', or 'a community of practice', where multiple practices belong to the same social world, and that creates special but also often individualised relations among them (Wenger 1998: 291 n.2). To study the dynamics of such a community, it is not useful to follow a single actor/actant. As explained in the methodology chapter, social worlds theory

enables us to analyse the interactions and socio-technical relationships between actors from different social worlds. It allows one to move away from the investigation of a technology which follows the translation of a dominant meaning from one world to another (Star 1991) and instead looks at various meanings given by actors associated with the technology and their negotiation. Additionally, one can explore the meanings and discourses held in common by different groups allowing them to work together and to work with the technology to achieve their specific ends (Star and Griesemer 1989) such as is the case with conventional mainstream software and unconventional hacking activities (as I shall show later in this chapter).

Deriving from social interactionism, social worlds theory deals with ‘social configurations created by a shared interest’ (Strauss 1978; Star 1991; Fujimura 1992; Clarke 1992; Gieryn 1995; Garrety 1997; Wenger 1998). Strauss (1987) describes social worlds including occupational worlds and others which have at their core common activities taking place around defined technologies. He remarks, ‘Likewise every social world is characterised by intersection processes, wherein it exchanges, negotiates, conflicts, and so on with other social worlds and subworlds.’ (Strauss et al. 1987:287). In this regard, we may define a hacker social world where diverse actors share a constellation of practice as described in the methodology chapter. The repertoire of practices defined has been developed and grounded over time in the social world and hence becomes central to the form of participation. In Wenger’s account, ‘With the notion of practice as a point of departure, it becomes necessary to pay attention to mechanisms of belonging beyond affiliation, and salient social categories are only part of the story.’ (Wenger 1998: 283 n. 8). Practice, in this respect, ‘yields a more tractable characterization of the concept of practice—in particular, by distinguishing it from less tractable terms like culture, activity, or structure’ (ibid.: 72). Implementing some or all of the constellation of practices in activities is one of the ways that actors manifest their membership. However, as argued earlier, a practice may be expressed in different ways. In fact, the multiple claim-makings to a collective practice have been commented on elsewhere. When Gieryn argues that ‘A social world is a group with shared commitments to the pursuit of a common task, who develop ideologies to define their work and who accumulate diverse resources needed to get the work done.’ (Gieryn, 1995:412), the term ‘ideologies’ is used to suggest how such

commitments are translated into various repertoires that support different forms of work.

Studying the constellation of practices and how such a constellation of practices is taken up in different social worlds, enables one to examine the granularities of ‘hacking’ (in a conceptual sense). In this research, the concept of practice, in line with Hutchby (2001, 2003), serves to illustrate the mutual interactions between social actors and technologies by examining the ‘affordance’ of technologies. In many ways, hacking technologies are treated as a means of problem solving in the software engineering field. As quoted earlier in this chapter, Denis regards his different hacking activities as “an appropriate application of a technology to solve a problem in another technical field.” (DG120901) (this problem-solving attitude will be explored more fully in the next chapter). While solving the problem at hand is considered as the technical goal of the practice, the process of problem solving is especially intriguing in sociological terms. It illustrates how a practice chosen to solve a specific problem can contribute to different ‘shared commitments’ across different social worlds. While the plasticity of practices does not preclude their being used in ways that are common across different social worlds, one shall see from the case of hacking technologies how an informal, marginalized practice and its associated tools and techniques can be deployed in formalised, mainstream settings. In the following section, I explain -- what I want to call -- this pan-hacking phenomenon in two ways: hacking for protection and institutionalised hacking, both in regard to an innate software problem, software vulnerability, and the need for security.

### **4.3.2 Shared practices**

Before discussing the two exemplars, it is necessary to discuss a common practice, which will play an important role in the process of problem solving. Given the constellation of practices (see 5.2.5 above), the most common practices found among the actors in the hacker social world are exploring software vulnerabilities or security holes, and sharing codes or information. As suggested by most interviewees, they attempt to solve problems in writing creative codes and sharing them, or writing challenging codes to explore software vulnerabilities and

reporting the bugs, and in so doing to fulfil their interest in tackling software problems and resolving them (i.e. #1 & 2 are done to achieve #5). Both the practices of writing challenging codes to explore software vulnerabilities and developing novel hardware and sharing the proprietary information on which it is based (#2 & 4) have a symbolic meaning in demonstrating the central belief in freedom of information in the hacker social world. Whereas finding security holes and exploiting vulnerabilities to get access to computer systems is to subvert the legal restriction on accessing information, writing software and sharing its source code is to challenge legal proscriptions about intellectual property rights that restrict copying or accessing information. However, my fieldwork, reported below, suggests how actors give different meanings to their actions, and in so doing justify their actions in distinct, indeed in conflicting social worlds. A good example of this is seen in ‘application buffer overflows’.

Whilst system administrators might use buffer overflows to effect a password recovery for clients, the same technique can be exploited by a potential intruder to shut down the application or gain high privileges on a server machine (i.e. take control over a system). Cross-site scripting (CSS), a more neutral technique than buffer overflows in the sense that it has little to do with a penetration test, targeting the browser rather than the server, is obviously manipulated for different purposes. Klein, a security manager for Sanctum Inc. explains the hybrid consequence of employing this technique (see Box 4.1 below): ‘Cross-site scripting allows a potential intruder to manipulate a link to a valid web site so that one of the parameters of the URL or maybe even the referrer will hold a script. This script will then be implanted by the server into a dynamic web page and will run on the client side. The script can then perform a ‘virtual hijacking’ of the user’s session and can capture information transferred between the user and the legitimate web application. The user activates the malicious link when s/he crawls through a third party site or by receiving an email with the link in a web enabled email client’ (Klein 2002). For potential intruders, CSS relies on an abuse of trust to obtain and transfer restricted data. But for system administrators, CSS can be used as an ‘ethical hacking technique’ to build a “transparent” single-sign-on (SSO)<sup>xxvi</sup>, in so doing to add a 3<sup>rd</sup> party web site in an existing authentication infrastructure without paying a lot of money for an SSO software, such as the

'Passport' design available from Microsoft.

This example shows JavaScript embedded as the value of one of the parameters of the login page. Once the link is pressed, the JavaScript residing on the third party site is activated and has full control over the client's browser.

### Correct request

```
POST /login.pl HTTP/1.0
```

...

```
title=Home%20Page
```

### Attack attempt

```
POST/ login.pl HTTP/1.0
```

...

```
title=<script
```

```
src='http://www.evilsite.com/evilscrip.js></script>
```

### Function

```
Function display_title ()  
{  
    global $title;  
    print "<B>Document $title</B>";  
    ...  
}
```

### Result of correct request

**Home Page** is displayed to the client

### Result of attack attempt

**Evil Script** is running on the client

*Box 4 1: Example of Cross-site scripting (Source: Amit Klein, ©Sanctum, Inc. 2002)*

CSS is just one of the many examples in our daily lives that an artefact (i.e. a technology, a technique, a tool, a tactic etc.) is adopted by a variety of actors for various purposes. The case is used to emphasise the multiple *raison d'être*,

interpretations and manipulations behind individual adoptions. In so doing, it helps to understand the mutual interactions between diverse actors and also between actors and actants. While intrusive hacking, such as buffer overflows or CSS, is encouraged to help find security holes and fortify network security, it is considered 'dangerous' from another point of view. The techniques can be used to break into a vital system, such as a hospital infrastructure. As Denise says, 'Intrusive hacking is only useful when the targets ask you to do it to them (such as the clients). So, they could monitor your activities and understand why, or why not, you broke in their systems.' (DG120901).

The techniques such as buffer overflows or CSS illustrate the concept of a shared practice that can be adopted, crafted, interpreted, and translated into different contexts. Not only are diverse actors found participating in the process, but also is it possible that one actor resides in more than one social world (e.g. a webmaster who knows how to use a buffer overflow to perform a password recovery conversely knows how to exploit it to gain control over an unauthorised system as a potential intruder does). The meanings involved in the collective practice might be complementary or contradictory to each other at some point, but they are all embedded in the common practice/artefact. It is one of the reasons why an understanding of the material practice/artefact is essential when investigating a dynamic and complex field. Though one might argue that the constellation of hacking practices given is not complete, this repertoire of practices is a conceptual category that at some level reflects the shared experience among actors. These practices might come together in their entirety as a constellation of hacking activities, or they might be pursued more variably and unevenly. What is interesting is that such a constellation of practices is not peculiar or specific to the activities typically ascribed to hackers, but might be said to characterise computer innovation more generally. In this way, while such a constellation of practices may not provide quantitative indicators to measure the degree to which hackers influence the development of software, they do allow us to map the pattern of activities that have played a central role in the ongoing development of innovative software (and indeed operating systems), and where and when these activities are found *within* and *outside* of conventional, mainstream computing. The technologies developed in line with the constellation of hacking practices, in this sense, are more dynamic and more complex than those developed in the

conventional environment, in that the shared practices enable diverse actors to engage in the innovation systems. To take this argument further, I suggest we can identify a pan hacking phenomenon, which arises when certain hacking technologies facilitate the problem-solving process, and do so in two contexts: hacking for protection and institutionalised hacking. In other words, the hacking technologies are shared in order to strengthen, to enable, and to make more efficient software functionality.

### **4.3.2.1 Hacking for protection**

In the software innovation process, meeting the requirement of users is usually regarded as the core objective of programming. How to meet diverse needs of users in a changing society has been the main concern of software engineering. Aside from functionality, security has become another main problem in software production, but long been ignored. It is observed that computer evolution has evolved from the PC paradigm to the network centric paradigm (Jackson, Mandeville & Potts 2002; Miles 1999). In the era of the network, innovation is a social process involving the cumulative growth of knowledge from many sources. The network of computers has enabled people to communicate faster and closer. Though the network has brought us the convenience of communication, the weakness of the network technology delivers risk to the network society. The vulnerability of software provides potential attackers the possibility of intruding into the computer system and breaching the data and privacy of users. Given many emerging computer crimes, President Bush's special adviser on cyberspace security, Richard Clarke, has said that 'software makers and Internet service providers must share the blame for vulnerable networks' (ZDNet August 2, 2002). The major issue that Clarke attempted to raise is that companies and organizations that create the hardware, software and services that make up the Internet are not doing enough to secure their products. In laying the blame for the vulnerabilities on the Internet, he pointed not only to software makers and ISPs, but also to those who create and use wireless networks, to the absence of a group responsible for securing the Internet, and to the government itself. Sociologically speaking, what Clarke brings up here is the concept of social construction of technology – a computer system is de facto constructed by diverse actors such as software and

hardware makers, ISPs, users etc.. When a breach occurs, all parties involved inevitably are concerned because there are so many points of potential vulnerability in the vast and complex systems of financial operations: hosting companies, Internet service providers, databases, transaction software and all manner of hardware.

This social diversity within the arena of software production makes the risk of human error hard to eliminate. Technical aspects however, exacerbate the situation. It is acknowledged that complexity, connectedness and extensibility are major factors making it much harder to create software that behaves (McGraw & Felten 1997). As McGraw observes, software is much more complicated than it used to be:

For example, in 1990 Windows 3.1 was two and a half million lines of code. Today, Windows XP is 40 million lines of code. And the best way to determine how many problems are going to be in a piece of software is to count how many lines of code it has. The simple metric goes like this: More lines, more bugs.

(C|Net News.com November 28 2001)

Vulnerability is an innate problem of software. It is said that there is no bug-free software in the world. Apart from its complex nature, the connectivity makes software more precarious. As the Internet is everywhere, and every piece of code written today exists in a networked world, software vulnerability is more easily detected and exploited. Additionally, the situation gets more unstable because of the trend towards building an extensible system. McGraw illustrates this when he notes that:

A perfect example of this is the Java Virtual Machine in a Web browser, or the .Net virtual machine, or the J2ME micro VM built into phones and PDAs. These are all systems that are meant to be extensible. With Java and .Net, you have a base system, and lots of functionality gets squirted down the wire just in time to assemble itself. This is mobile code. The idea is that I cannot anticipate every

kind of program that might want to run on my phone, so I create an extensible system and allow code to arrive as it is needed. Not all of the code is baked in. There are a lot of economic reasons why this is a good thing and a lot of scary things that can happen as a result.

(ibid.)

In the case of an extensible system, it is difficult to predict how, when, and by whom the system is going to be incorporated and integrated into what. Sometimes a set of code arrives with a short notice that changes the environment dramatically. To combine diverse technologies through rapid change, programmers usually do not have time to go through every line of code to determine whether it presents a vulnerability. An extensible system therefore is usually exposed to immeasurable risk of attack.

Many episodes can be used to show how mainstream ICT groups draw on the hacker social world to help deal with these questions of security. Clarke, Bush's cyber security consultant, issued a call for participation to those who can help secure the Internet to 'step up to the plate' if and when they detect any core software vulnerability. Recent coverage shows 'hackers' are usually targeted (e.g. Kevin Mitnick) to cooperate with public agencies in fighting against security flaws. For example, after the 911 attack in New York, the US government recruited many talented "hackers" in the name of fighting cyber terrorism. Banks and financial services firms suffering from sabotage, or system crashes often turn to hacker groups for help if their own security teams are not capable enough to deal with very skilled attacks. Apart from establishing security firms to handle frequent electronic break-ins, many hacker groups, being aware of their technical advantage, also try to educate their peers on how to learn about network security and on how to find ways to improve it without doing any harm: some publish security books, others organise security conferences (e.g. the *BlackHat Conference*), others open 'security schools'. Being insiders in a specific field, hackers have their own tacit knowledge. Insiders know how an attack is normally done with tacit knowledge and therefore are more likely to provide an efficient counter plan to stop the attack. Caesar, a 28-year-old security consultant and founding member of GhettoHackers (a self-claimed hacker group) claimed that

‘The Ghetto are good guys, So I guess the way to look at us is (as) the boot camp for the people growing up to protect the world’ (‘Hacking their image’ C|NET News.com August 2, 2002). These hackers describe their actions as ‘ethical hacking’. But hacking practice, as insiders are aware too, serves as two sides of a sword: one can use it for the good but the other can use it for the bad. And the meaning of hacking is viewed differently in different circumstances.

#### **4.3.2.2 Institutionalised hacking**

In light of the argument above, not least because can we find widespread evidence of activities that express a constellation of hacking practices within the mainstream computer world, practices typically associated with hacking that are usually ascribed to hackers appear to make certain positive contributions. Heterogeneity in this regard is neither completely unconstrained nor completely unstructured. That is, we need to ask who is engaged in the practices, what label is applied to the practices, and how actors from different social worlds enact the practices in different social settings? In some circumstances, when practices are deployed by social groups with greater power in the wider society they will be likely to be framed quite differently. The exemplar below demonstrates how unauthorised hacking was deemed to be legal when employed by the Recording Industry Association of America (RIAA) and the Motion Picture Association of America (MPAA) both of which enjoy considerably social and material capital.

The recording and movie industry trade groups have been troubled with certain file-sharing networks such as Gnutella, Morpheus, and KaZaA whereas many users think the advent of these file-swapping technologies is a blessing. For their sake, RIAA tempted to secure the privilege to legally hack into peer-to-peer (P2P) users’ computers to prevent or thwart the distribution of free mp3 files. On 25 July 2002, US Democratic Congressman of California Howard Berman proposed a Bill in Congress which would allow the recording industry to legally hack into systems suspected of sharing copyrighted material. The **Bill** does not specify what techniques, such as viruses, worms, denial-of-service attacks and domain name hijacking, would be permissible. It does say that a copyright hacker should not delete files, but it limits the right of anyone subject to an intrusion to sue if files

are accidentally erased. Berman argued that:

[This Bill] does not allow copyright owners to send viruses through P2P networks, destroy files, hack into the personal files of P2P users, or indiscriminately block lawful file-trading. [But it does allow] disabling, interfering with, blocking, diverting, or otherwise impairing the unauthorised distribution display, performance, or reproduction of his or her copyrighted work on a publicly accessible peer-to-peer file trading network.

(AntivirusAbout 30 July 2002)

Though the Bill includes a number of provisions, including a requirement to notify the Department of Justice seven days prior to engaging in an attack, it still raises immense controversy about the legitimacy of hacking technology. This proposed Bill hands copyright owners substantial new control over the distribution of their works by curtailing a consumer's right to copy material under a doctrine known as "fair use". If this bill gets its way in the Congress<sup>xxvii</sup>, Americans are no longer allowed to record a TV programme or radio segment that generally may sell or otherwise dispose of that analog recording or digital files as they wish. The bill is crafted to level the playing field between copyright holders and so-called "file traders", as Berman put it:

In other words, while P2P technology is free to innovate new and more efficient methods of distribution that further exacerbate the piracy problem, copyright owners are not equally free to craft technological responses. This is not fair. ... Songwriters, photographers, film producers, karaoke tape makers and other copyright owners are experiencing massive piracy of their works through P2P networks. Billions of P2P downloads every month constitute copyright infringements for which these creators and owners receive no compensation. There is no excuse or justification for this piracy. Theft is Theft, whether it is shoplifting a CD in a record store, or illegally downloading a song from Morpheus.

(AntivirusAbout300702)

It was thought that this Bill was supported by RIAA and MPAA, following their attempt to attach an ‘anti-piracy amendment’ (source) to an anti-terrorism bill in October 2001, and their lawsuit against three prominent file-swapping companies for infringing their copyright in the Los Angeles federal court: Morpheus parent StreamCast Networks, Grokster and Kazaa BV<sup>xxviii</sup>, the Netherlands-based company that originally created the Kazaa software. While RIAA and similar companies feel that their right cannot be infringed, P2P users also feel that they have the freedom to share information. Some voices from the civil right groups criticised the proposal saying it would encourage profiling of and vigilantism against users. It is reported that unknown parties have launched a Denial of Service (DoS) attack against the RIAA following the announcement of the proposed bill, making the site virtually inaccessible to legitimate traffic (AntivirusAbout300702).

Those behind the Bill are not the only party in favour of hacking in the name of a self-defensive strategy against file sharing networks. P2P networks constantly bring security risks in that viruses and worms are easily transferred through the networks. Network security companies also advocate a similar strategy to that of RIAA to stop virus as from spreading. At DefCon2002 the US hacker conference, Timothy Mullen, chief information officer of AnchorIS and a columnist for SecurityFocus.com, suggested a technique for machines that have been attacked but not infected with a virus to trace the worm back to the attacking machine and prevent it from spreading the worm to other computers. Using his techniques, the computer that launches an attack is paralysed and requires an administrator to restart it, but it stays online and is not otherwise harmed. The rationale for this approach is that the current way of dealing with virus attacks – contacting the administrators of infected and attacking computers – is not effective. Mullen claimed it that ‘This after-the-fact stuff clearly doesn’t work. I’m still getting Nimda<sup>xxix</sup> attacks, often from the same person.’ (CNET News.com 4 August 2002). But Mullen is also aware of the illegitimacy of this technique: ‘What we’re doing, [according] to the letter of the law, is illegal. I would like to see the law changed... We’ve illustrated not just a reasonable recourse, but a minimal responsibility.’ (ibid.). Nonetheless, Mullen’s idea has not gained much momentum elsewhere and

surprisingly it has been the US officials who have questioned the ethics of the idea. Mark Eckenwiler, senior counsel at the US Justice Department's computer crime division says, 'You have trespassed on their system. There are more legally acceptable ways to deal with the problem than what is essentially hacking into their system.' (ibid.). Another commentator from the US Department of Defense's Command, Control, Communications & Intelligence office says, 'There also is the possibility of hacking back at the wrong computer. It is the DoD's policy not to take active measures against anybody because of the lack of certainty of getting the right person.' (ibid.). But this type of hacking is claimed to be understandable by Jennifer Stisa Grannick, litigation director at the Center for Internet and Society at Stanford Law School: 'Mullen's idea may be protected under a self-defence provision. This is a type of defence of property. There is a lot of sympathy for that [kind of action] from law enforcement and vendors because we do have such a big problem with viruses.' (ibid.).

### **4.3.3 Cross-Boundary practices**

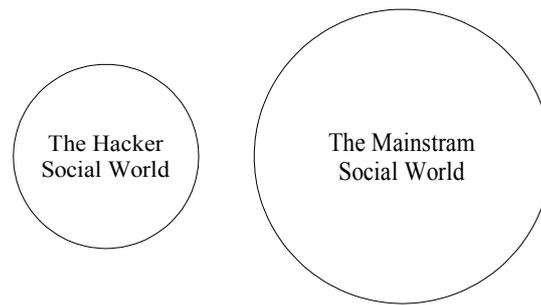
In my analysis so far, I have sought to show how the core concern over protection of and vulnerability in computer software becomes a socio-technical site that saw increasing opportunities for mutual engagement between the hacker and mainstream ICT social worlds. Here, the latter social world refers to a formally institutionalised, professionalised, mainstream world of computing based on legal convention. In this next section I want to develop this argument more fully in terms of suggesting a series of engagements through which the two worlds come more closely together. I go on to suggest that where overlap does occur, anchored in the constellation of practices described earlier, this is an arena where we are most likely to see new pattern of ICT innovation. As my later chapters show, this is not restricted to questions of security. My support for this analysis is material and for the latter part or primary data that I explain much more fully in later chapters.

#### **4.3.3.1 Two independent social worlds**

In my review of hacking for protection, the technology per se is regarded as

useful to the state and the corporation in that it helps to find security holes and to fortify system networks. But this was not a presumption nor was it a given fact; it was realised in the process of negotiation between actors from the mainstream social world and the hacker social world. The demarcation of both social worlds formed at the time when the security problem started emerging in the software engineering, roughly just after the end of the Second World War when the first electronic mathematical computation machines were constructed. By then, the state and the corporation (especially the American military who mainly led and sponsored the research projects) did not pay attention to the risk of software vulnerability and network security addressed by the computer scientists (the hackers in the 50s and 60s) (Abbate 2000; Hafner & Lyon 1996). Since the very first computers were large mainframe machines controlled by the government and were used for classified military and defence purposes, unauthorised abuses occurred at these highly secured installations during the early days of the computer revolution have never been reported. The mainstream considered the world they live as a safe and sound milieu and did not care about the security problem in the wider computer networks.

For almost 3 decades (1946-1976), ‘the instances of computer-related deviance were treated as peculiar news events regarding this strange and exotic technology’ (Crime, Deviance and the Computer: xviii). At this time, only a handful of people recognized the importance of the computer to our society and were capable of assessing the potential risk of the innate software vulnerability. The hacker social world was isolated and had little contact with the outside world. As fig. 4.1 illustrates, the hacker social world and the mainstream social world were (almost) independent from each other.



*Figure 4.1: Two Independent Social Worlds: the hacker social world and the mainstream social world*

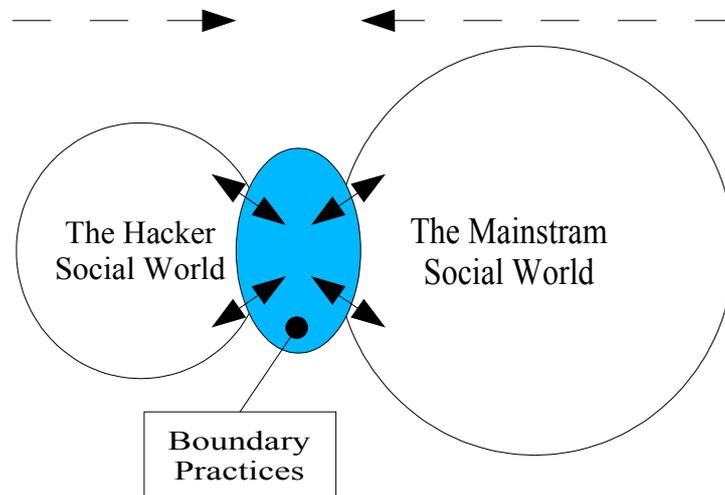
In spite of a clearer demarcation, one should not presume the hacker social world as a tight subculture. Apart from the joint interest in computing, “hackers” were connected to various lines of works in the development of computation. At the developing stage of computation, security was just one of the many challenges “hackers” were facing. But with the development of computation, the positions of the two social worlds and their relationships were to change.

#### **4.3.3.2 The opposition of the social worlds**

After the advent of personal computers (PCs) and the computer networks, security glitches gradually became problematic. Having little knowledge in the new technology, the issue of security was beyond the awareness of users in the mainstream social world. The innocence led the public to treat all computer-related flaws as deviancy, regardless of the possibility of good will (i.e. to raise the vigilance of the public to the security problem). Emerging out of a fear of new technology, social stigmas were labelled on those who spent a long time in front of the PCs and named “hackers” (e.g. Turkle 1985). This was during 1977-1987 when the PC and the Internet were emerging. During the period of the rapid popularisation of the PC and the Internet, digital flaws were commonplace. Meanwhile, there was a mixed feeling of phobia and philia towards the new prevailing technology. While the machine and wider connectivity empowered lay users, the affordance of the technologies, in Hutchby’s terms (2001, 2003), also enabled potential intruders to embark on attacks more easily in an exposed security environment. Facing various threatens from viruses, webpage defacement and

system intrusions etc., the mainstream social world was, on the one hand, scared of, and on the other, hostile towards “hackers”. Most coverage on hackers in the mass media was negative and stereotyped. Symptomatic of this, perhaps, ‘computer ethics’ became a new module in Departments of Computer Science.

When the censorship of computer activity arose during 1990s on (the most notable one is the US Communications Decency Act of 1995), the opposition between the hacker social world and the mainstream social world was redefined, as illustrated in the fig. 4.2.



*Figure 4.2: The connection of the two social worlds provided by the boundary practice*

Originally the hacker social world was unconnected with the mainstream one, but the impact of hacking had connected these independent social worlds and brought conflict as well as negotiation. Hacking practice served as a boundary practice through which two social worlds could encounter each other. The relationship between the mainstream social world and the hacker social world had shifted from remoteness to reciprocity, though coming from opposing positions. Apart from exploring and exploiting security flaws, the free software movement was initiated during this period to demonstrate the utility of the free exchange of information. The establishment of the Free Software Foundation (FSF) in 1985

had been the benchmark for this. Hackers, in fact, widely engaged in various lines of activities. Hacking practices might be used to show off their skills as well as to attract the attention of the mainstream social world.

In the beginning, mainstream ICT did not care about hackers and what were they doing. But suffering from attacks, the mainstream, without understanding how hacking actually took place, considered hackers as enemies. Hackers and their technologies were marginalized and criminalized. When hacking was exploited by “hackers” primarily to connect the two social worlds, the initial contact was conflictual. The intrusive elements of the technology were accentuated. Worse, hackers were demonised in the mass media, for example, the notorious Kevin Mitnick (e.g. the 15 January 1990 AT&T system crash), the widespread viruses and worms (the first Internet worm was born in November 1988 when a graduate student at Cornell University unleashed a system “worm” on the telecommunication network which connected all major research mainframe computers in the world), and the potential cyber war (e.g. a number of computer buses happened in the American military networks are done by the East German or Soviet KGB, etc.).

It might be useful here to characterise this phenomenon as similar to Young’s (1971, 1973) analysis of drug takers. In his view,

Drug taking is not an essential prerequisite of membership. Rather it is used instrumentally for hedonistic and expressive purposes and symbolically as a sign of the exotic ‘differences’ of the bohemian. Drugs are thus an important although not central focus of such groups. Drugs hold a great fascination for the non-drug taker, and in the stereotype drugs are held to be the primary—if not exclusive—concern of such groups. Thus a peripheral activity is misperceived as a central group activity.

(Young 1973: 353)

However, when the society defines the drug-taking practice as deviant, ‘the drug-taking group creates its own circumstances to the extent that it interprets and

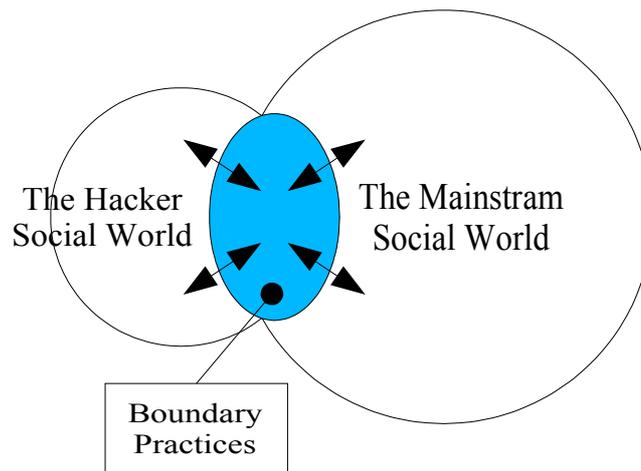
makes meaningful the reactions of the police against it' (ibid.: 351). Hacking in a sense experiences something similar to the labelling of bohemian groups as 'drugs takers'. The hacker social world has always been more heterogeneous and diverse than the deviant and reductive label implies. If there was a subculture, it was socially constructed as a pseudo-subculture that might be result of the deviance amplification effect of the social control that consequently excludes hackers from "normal" society.

However, the antagonism did not last long; since confrontation and negotiation had also accompanied the conflicts between the mainstream and hackers. Hacking knowledge/practices turned out to be the medium for reconciliation.

### **4.3.3.3 The encounter of the social worlds**

As the hacker social world is formed around a constellation of common practices, given the versatility of the practices, the boundary of the social world is permeable. As argued earlier, the same technology/practice might be exploited for other purposes by other actors from different social worlds. For instance, buffer overflows can be used to gain unauthorised access to systems and destroy or steal the restricted data, whereas vigilant hacking (or "ethical hacking") can be done through a penetration test of the buffer overflows. Being a locus for mutual engagement between the two social worlds, hacking practices/techniques bring the two social worlds into encounter. When they were moving towards each other, their boundaries started to collapse and converge and negotiation between them occurred after the first encounter. All parties have conceded at some level: Hackers open up the gateway to their world by revealing their tacit knowledge either via the Internet or through conferencing or lecturing, whereas the state and the corporate pay more attentions to the issues raised by hackers and to learn hacking knowledge. Other forms of exchange and communication occurred: hackers translate their tacit knowledge into a form that is easier to be understood by the outside world, whereas the mainstream integrates hackers and their practices into routine work. Through reciprocal interactions, hackers and the mainstream both gradually find a way to communicate. After mutual engagement in the technology, the distinction between "Us" and "Them", as seen in the

narrative of Rop above, began to blur. The two social worlds are still distinct but the boundary between them is more open to mutual crossing. By taking up the practice, both the mainstream and the hacker move forward to each other. More and more they have common ground, a pan-hacking language and practice. The concept of “others” may still exist, but each entity realises that “others” could be someone to cooperate with. Hackers are no longer “strangers” to the state and the corporation (independence, Figure 4.1), neither are they the rival party (opposition, Figure 4.2). Instead, hackers are more often regarded as companions or partners who might contribute to the state power and the corporate business. At this stage, the two social worlds have overlapped by sharing the same hacking practice (Figure 4.3). Hacking practices, originally regarded as deviant, is found widespread in different social worlds. The shared practice serves as a “boundary object” for diverse actors to interplay, to achieve different purposes (Star 1989).



*Figure 4.3: Cross-boundary: overlapping social worlds*

In the process of negotiation, hacking practice and hackers are assimilated into the mainstream social world whereas the hacker social world absorbs certain schemes from the other. Conflicts and contradictions still exist within or outside these social worlds (the two social worlds are still distinct), but the interface area of these worlds has widened (the boundary of each social world becomes blurred). I want to argue that this overlap is the most dynamic place in each world where

cross-boundary activities and co-construction of hacking technology are found and where the specific contribution of the hacker social world to wider ICT innovation is likely to be most evident.

Periods (year)	social situation of hacking practice	Technical situation of hacking practice	The position of the two social worlds
1946-76	The discovery of computer abuse	Marginalized and ignored	Isolated and remote
1977-87	The criminalisation of computer crime	Consultation	Connected but opposite
1988-1992	The demonisation of hackers	Acknowledged	Encounter
1993-1999	The censorship period	Developed	Convergence
2000-present	Censorship + coordination	Institutionalised	Integrated

*Table 4 2: Focal periods of the sequence of interactions between the two social worlds*

The sequence of the motion of the social worlds associated with the hacking technologies and the peripheral socio-technical situations are summarised in the Table above (Table 4.1).

I do not want to claim however that ‘all’ hackers move in a style direction. As a result, groupings which have traditionally been theorised as coherent subcultures are better understood as a series of fragile and temporal gatherings characterised by fluid boundaries and floating memberships, in line with Young (1971) as argued above. While the diagram depicted and the explanation given in the narratives of hacking seem to make sense, one should not overlook the heterogeneity in the hacker social world as shown earlier. The actual situation hence is more complex than the tidy diagrams offered here. Multiple meanings are given to the practice and the technology: while some hackers consider their action as ethical and are willing to be absorbed by the mainstream, others may consider them as losing the self-claimed hacker mentality; while some officials suggest cooperation with the hacker social world, others in the mainstream maintain a

strong reluctance to do so. As a result, the uncovered area of each social world in Figure 4.3 represents the unassimilated world while the overlapping area represents a milieu where episodes based on collective practices take place.

The analysis also suggests that hackers often occupy hybrid socio-technical positions in the social world. As shown earlier, many interviewees consider themselves as “grey hat” rather than purely “white” or purely “black” in the hacker social world. They also acknowledge that it is difficult to be ascribed or avow a firm title. Not least because they may occupy dual socio-technical positions in the mean time, the rapid mobility of knowledge often drives hackers in and out between several subworlds subordinate to the wider hacker social world, as discussed in 4.2.5. The hybridity of identity can be explained from another socio-technical angle. Technically speaking, hacking technologies are considered as very insider knowledge with high expertise. However, socially and legally speaking, the technologies were (and still are) deemed as outsider behaviours with deviant intentions. The socio-technical hybridity renders a profound situation described in 4.3 where it introduces two exemplars of hacking technologies being applied to different sectors. The hybrid view on hackers does not mean to complicate the heterogeneous field. It offers a sociological perspective on the complex socio-technical interactions that resonates the essence of seeing “hacker” as a conceptual term.

The degree of hybridity and fluidity increases when the hacker social world encounters other social worlds. Considering the diagram shown above (Figure 4.3), the space for actors in the hacker social world has enlarged when the hacker social world comes across the mainstream social world. More social actors have been included in the hacker social world because the boundary of each social world is no longer distinct. It is more possible for actors to have dual identities, moving easily from one world to the other by engaging in the shared practice. Standing right on the border of the hacker social world, a self-described ethical hacker in fact occupies both the hacker social world and the mainstream social world. Meanwhile, knowledge in each social world also flows across the boundary. The tacit knowledge in the hacker social world, after being brought into the mainstream social world, transforms some actors in the mainstream from

outsiders into insiders of the hacker social world. It is also possible that hackers get involved in the policy or business directly in the mainstream social world. The blurred boundary enables the knowledge to travel across social worlds and to cause peripheral effects on both social worlds. Outsiders have opportunities participating in the counterpart social world. The grey overlapping area is the most dynamic, most hybrid and most complex milieu in the wider computer world.

By ‘possibility’ or ‘opportunity’ when speaking of the transition of insider/outsider is reflexive. Take myself for example, when I entered the open field of HAL2001, at times I felt like full participation. But once a while something would happen that reminded me that I was an outsider: it might be a jargon I did not understand, a mistrusting look from the interviewee, a reference to a past event, a name of the person known very well, and so on. My experience reflects the importance of both peer knowledge and participation for being an insider, as discussed in 4.2.5.

### **4.3.3.4 The translation of hacking practices**

Whereas the narrative of hacking for protection illustrates the cross-boundary feature across different social worlds, “legal hacking” will take the argument further to demonstrate that hacking is actually contextualised. ‘Hacking’ is translated and imposed to different social meanings in different social settings.

In light of the studies on social problems and the mass media (Cohen & Young 1973), it is arguable that hacking and hackers are usually biased presented. In the mass media, usually hacking denotes malice or deviancy, and hackers are labelled as subversive individuals or gang-organisations. As seen earlier, hacking has been imposed to many different meanings and serves to achieve diverse social and technical end, for instance finding security holes, satisfying the curiosity and adventurous mentality for some “users”<sup>xxx</sup>, or being used for political purposes to articulate diverse voices on-line. While hacking is considered as offensive and aggressive technology, it is cynical to see that RIAA and MPAA, the two very illustrative figures having poignant relationship with the media, lobby to employ hacking, a controversial technology, to secure their profits. It is also odd to see that

hacking, developed by those who come from in the same social world and develop the file-sharing technology, which contradicts RIAA and MPAA's interest the most, are used by RIAA and MPAA to fight back. It is paradoxical that the media social world, where information is considered as commodity and proprietary, shares the same practice – break-ins – with the hacker social world, where the freedom of information is the rule of thumb. The same practice is imposed to different meanings when being used by different parties. It is definitely illegal for an individual or a group hacking into any system of a media agency to see what are they doing, but oddly it is claimed to be legitimate for RIAA and MPAA to hack into a user's system to see whether or not is there a file-swapping programme installed in his/her machine. The identity of the technology has transformed dramatically from a marginalized and criminalized position to a crucial and influential one. A knowledge originally regarded as deviant has attracted attentions from the mainstream in that it copes with the mundane utility for certain users who happen to be one of the powerful agents holding robust resources in the society. Practically speaking, the practice of break-ins, for organisations both in the hacker social world and the mainstream, functions to gain unauthorised access to the counterparts' systems. But the socio-technical consequences in different contexts are different. Hacking technologies, on the one hand, serves as a boundary object and provides a platform for the two social worlds to interact (see 4.3.3); on the other hand, the technologies (the boundary object) is imposed to different meanings and given different social identities in different contexts. This narrative of RIAA and MPAA illustrates how hacking technologies are contextualised and socially constructed. Moreover, considering diverse actors in the wider society, each actor has his/her own say to hacking and hackers. As a result, it is sociologically inappropriate to prescribe a definite meaning to hacking or hackers.

As the above narratives serve to illustrate, rather than assuming a quintessentially fixed subculture, the membership in the hacker social world is quite loose and the boundary is soft. When we study the hacker social world, we are studying people coming from a variety of social backgrounds and occupying a range of social positions and statuses with the common engagement in hacking. As exemplified in the hackfests, there might be some circumstances where hacking activities coalesce and create visible social worlds based on shared practices that are occupied by a range of social actors, though not all participants should be

assumed to engage in. Additionally, as the hacking boundary has a close connection with other social worlds, it won't be accurate to understand the hacking group within only one boundary. Therefore, a cross-boundary research is necessary to conceive of and study a range of interactions and consequences. This allows us to avoid pre-judging people as hackers, to see how hacking practices could be found in a wide range of social setting, to observe how hacking practices are built into different social worlds and become institutionalised, and to map the social structure behind the hacking practices. Because these practices can be readily found among the majority of insiders in the computer world, this might give us a chance to observe what I would like to call 'pan-hacking phenomenon', a practice-oriented fact, an activity that is much more extensive than conventionally assumed, and suggests points of contiguity and overlap between mainstream and 'outsider' innovation. This overlap has, in fact, been increasingly seen to be important in a number of areas where new technologies are involved, notably in the case of computer security (software patches or network security). In terms of another common practice in the hacker social world, sharing source code, this phenomenon also applies to what I want to explore in the next chapter, and do so through the development of free/open source software as a field that occupies a dual site *within* and *outside* mainstream computing.

### 4.4 Conclusion

This chapter investigates the heterogeneity in the hacker social world in light of the fieldwork and interviews done at HAL2001, a hacker event took place in the Netherlands, and at 2600 hacker meetings in Hull, London and Leeds. The diversity found resonates the idea that an 'identikit portrait' of hacker is not pragmatic because in reality there is no a firm sociological 'photokit' of hacker. This finding leads one to move away from a dominant hacker culture that would derive from the subcultural theory, to a more ecological point of view on the heterogeneity. The finding also suggests that the identity of hacker is fluid and the boundary of hacker social world is soft and flexible based on a range of practices. The chapter suggests not to review on the essential position of hackers but a range of collective practices influencing software technology, in that 'hacker' is a conceptual notion; it is a metaphor, which acquired a plurality of meaning in

sociological discourses. This will provide us a view on how things come together in different social worlds. Two major practices are central to hacking: exploiting software vulnerabilities and sharing software source code. Based on the former practice, two narratives are given to demonstrate the negotiation between the hacker social world and the mainstream social world. One can see this practice being incorporated into a wider computer world and becoming the collective practice across social worlds. In the process of negotiating the meanings of the shared practice between diverse actors, the relationships of different social worlds have undergone a periodical change, from isolation, opposition, connection to incorporation. Owing to the structural factors, the practice may well be institutionalised in the mainstream. I have coined a new term “the pan-hacking phenomenon” to analyse the structural change of the social worlds through examining the cross-boundary activity of the hacking practice. The pan-hacking phenomenon illustrates a 'co-fabrication of knowledge and identities' in Callon & Rabearisoa's account (2003). It proves that knowledge and practices deemed to be marginalized and deviant can contribute positively to the innovation in the wild by mutual engagement in the shared practice. In this practice-oriented view it is to show that one's contribution to the technological system does not depend on one's identity, but one's practices.

Now I would like to explore this further by examining a specific case to see how the relationship between hacking and FLOSS-based technologies develops and evolves.

## Chapter 5 A Case Study of the Problem-Solving Process in Developing FLOSS

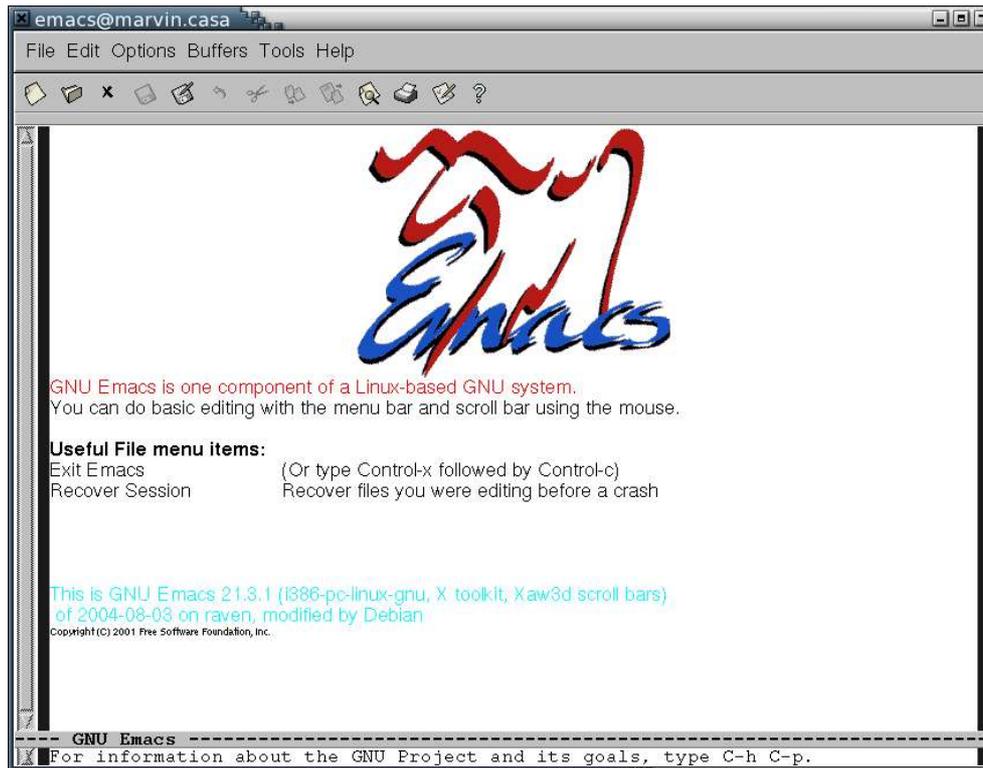


Figure 5.1: A screenshot of Emacs

### 5.0 Introduction

In Chapter 4 I explored the socio-technical heterogeneity of the hacker social world, based primarily, though not exclusively, on my empirical fieldwork at the hacker conference. I argued that it is methodologically inappropriate to define and categorise “hackers” and “hacking” according to clearly defined boundary identities or sub-cultures, as the majority of previous literature has done (c.f. Taylor 1999). Some actors within the social world of computer hacking assign a

negative meaning to the term “hacker”, while others proudly proclaim themselves to be “hackers” yet simultaneously occupy a position in mainstream computer innovation, and still others find the term too ambiguous to even define when asked to do so. In short, the different interpretations of the meaning of the term “hacker” reflect the complex composition and structure of the hacker social world. Hence, this calls for a sociological analysis of the variety of socio-technical relationships between actors from different social worlds and the communication processes through which they negotiate and handle their software needs.

To investigate this situation, the research treats ‘hackers’ and ‘hacking’ as available repertoires of meaning and practice in order to analyse how actors engaging in the hacker social world interpret and make sense of them through sharing a range of socio-technical practices. It is suggested that these serve as ‘boundary practices’ to enable diverse actors to occupy a position within and across the hacker social world. The shared practices provide a platform for diverse actors to interact and negotiate their readings of artefacts and metaphors. In the longer term, the affordance (Hutchby 2001) of the technologies invites more actors to engage in these practices and expand the social world of hacking activity itself, and bring new interests to it. As Henwood *et al.* (2000) have argued elsewhere, “[T]echnologies are the material embodiment of the values and interests of particular social groups or classes. ... Designers, engineers and their managers and financiers have values, goals, assumptions and even prejudices that are built into technologies.” (Henwood *et al.* 2000: 11). Through a mutual process of socio-technical shaping, technologies and actors influence each other reciprocally. Technology is a medium through which actors’ practices are enabled, but it is also shaped and directed along particular paths of development, as has been suggested by Callon’s (1999) notion of irreversibility. In other words, the affordance of a technology is contextualised and influenced over time by cultural, economic, organisational, political and technical factors. As the exemplars given in Chapter 4 illustrate, a shared concern found within the social world of hacking—exploring software vulnerabilities and reporting bugs—is itself given expression in different ways across the hacker and mainstream social worlds, but which, over time, converge or overlap around particular interests and patterns of activity.

Technological devices associated with software innovation, such as network/system-scan/penetration techniques, are developed to perform specific functions. However, the examples of CSS or buffer overflows in Chapter 4 have also shown that an identical technique or software tool can be used in different contexts to produce quite different results. In the area of network scanning tools, diverse actors adopt the same tools for different purposes that are regarded as either beneficial or potentially damaging to current software. In other words,

[T]echnologies are not primarily material objects but constitute an arena for contesting meaning. The physical capabilities of the artifacts are not paramount; rather the cultural meanings given to them or read into them are important. Because these meanings are contested and fought over by different social groups, the same artefact will be understood differently over time and across cultures.

(Henwood *et al.* 2000: 11)

Another example that I explored in Chapter 4 was the interaction between RIAA, MPAA and the hacker social world from which decryption tools (e.g. DeCSS) and file-swapping tools (e.g. P2P [peer-to-peer] technology) emerge. To tackle the pirate CDs and illegal downloads more effectively, RIAA and MPAA changed their tactics: apart from adopting legal means, they deployed and exploited network/system penetration technologies. As these technologies are celebrated within and indeed have been developed in the hacker social world, it is something of a paradox that RIAA and MPAA make this strategic move to strengthen their technical resilience. This is not to suggest that RIAA and MPAA are intrinsically against P2P technologies, nor does it imply that RIAA and MPAA are against freedom of speech/information. It is just that decryption and file-swapping technologies have posed threats against RIAA/MPAA's and their counterparts' interests. And it is ironic to see RIAA and MPAA adopt rival technologies to fight against their rivals. This episode also points to the question of moving across boundaries and the need as analysts to avoid making or seeking clearly defined categories of actor/behaviour in trying to understand the social

world. Rather we might be better adopting the notion of hybridity: as Latour (1993a) puts it,

The dual mistake of the materialists and of the sociologists is to start with essences, either those of subjects or those of objects ... Neither the subject, nor the object, nor their goals are fixed for ever. We have to shift our attention to this unknown X, this hybrid which can truly be said to act

(Latour, 1993a: 6)

In light of Latour's methodological insight, I propose to analyse technological innovation systems (here, the FLOSS system) by allowing some sense of technological determination yet embracing too the notion of the socio-technical construction of technology. The former leads one to consider the affordance of a technology in the context that would provide solutions for problems at diverse actors' hands and determine users' behaviours. The latter suggests the need to recognise the socio-technical construction of a technology whereby economic, political and social values inform its innovation, shape it and direct it along a certain path. It is worth also asking why certain technologies (such as software) appear to enable a higher level of affordance than others.

Following on from the discussion of Chapter 4, this Chapter explores the interactions and relationships between actors and actants that play important roles in the development of FLOSS (Free/Libre Open Source Software), which is regarded as the principal contribution towards computer innovation that has emerged from the social world of hacking. The FLOSS phenomenon illustrates how the socio-technical potentiality of technologies can be developed in a highly open and heterogeneous environment, where various sources of innovation are brought about to mobilise the ongoing development of FLOSS itself. The FLOSS story reinforces my argument that there is no self-defining, linear cause-and-effect process of innovation. Technological innovation is a contingent and heterogeneous process particularly in terms of highly individualised and plastic socio-technologies such as software.

The Chapter is structured as follows: it begins with the story of EMACS (short for Editing MACroS), an editor programme originally written for TECO language and PDP-10 machines in the MIT AI Lab by Richard Stallman, from which various more sophisticated versions have been developed. I analyse how the innovation of EMACS took place over time as a socio-technical process. The EMACS story serves to illustrate how the innovation process in the FLOSS community occurred, but one that is then adopted and deployed in other social contexts, including the commercial sector. The analysis of EMACS is especially useful since it spans the period that saw the origins of the free software movement and the subsequent development of a broader FLOSS social world.

Apart from this, there are a number of reasons why EMACS provides a valuable illustration of the heterogeneous and contingent FLOSS innovation system. Firstly, EMACS signifies the beginning of the General Public License (GPL), one of the most important contributions that Richard Stallman made to FLOSS innovation. EMACS's connection with GPL date back to the TECO EMAC written by Stallman and colleagues at the MIT AI Lab in the late 70s. It had been designed to be used and developed via an explicit social contract – a sort of ‘innovation contract’ - between users and developers for their mutual benefit, enabling those involved to notify each other about the modifications they proposed to elements of the system. Secondly, GNU EMACS had been in use at more than a hundred sites (Stallman 1998: 16) and had a number of related editor software in place. Hence, EMACS will, on the one hand, illustrate the emerging innovation process from invention, through implementation, to diffusion, and on the other hand, show at work the complex web of classification and standardisation that began to shape and define the system as it became established and embedded in the wider FLOSS context. It will also allow us to explore the affordances of EMACS and how this diverse potentiality is exploited in different ways such as gaming, web browsing, editing, compiling and testing programmes, to name just a few. Furthermore, EMACS, an editor programme that can edit codes for crafting other programmes, functions as one of the essential programmes in the GNU/Linux operating system. Studying EMACS therefore provides a rich site through which we can investigate the relationships and interactions between actants and actors in the innovation system.

## 5.1 Problems and solutions: the *sine qua non* of the software innovation process

A problem can be seen as the inauguration of an innovation. Many technologies are initiated when problems come into scientists' or engineers' sight. This point is made in a text about 'inventorship': "To succeed at inventorship, [one] must understand that the existence of a solution does not imply that there is no longer a problem to be solved. It may simply mean that a single, orthodox way of dealing with a situation has existed for years and has never been questioned." (Greene 2001: 74). However, perceiving problems may initially at least, be an individual matter; something regarded as a problem for someone could be an ordinary everyday event or phenomenon for another. As Borgman (2003) observes "From a cognitive perspective, all problems have three basic components: 1) A set of given information or a description of the problem; 2) A set of operations or actions that the problem solver can use to achieve a solution; 3) A goal or description of what would constitute a solution to the problem." (Borgman 2003: 99). In this regard, defining a problem can be linked to the process through which people organise a difficulty they confront into something that can be defined, classified and thereby framed according to a 'problematic': in this sense, we can speak of problem-definition as a process of classification. In seeking to deal with the problem, innovators typically try to classify the problem in relation to an existing paradigm (cf. Kuhn 1970). When one is making a classification, s/he has taken the first step of drawing her/his boundary around the issues such that whoever perceives the same in the same way, as this problem, may join forces, and become part of a group. In other words, a defined problem may be seen as a temporary alignment, which has been recognised as a legitimate problem by others for them to become shared concerns or issues. However, seeking solutions for defined problems may bring divergence to this temporary assemblage. Members who agree and endeavour to render the same solution form a social group under this frame of the defined problem, whereas others may come up with different solutions and draw other boundaries around and so classifications of 'the problem'. A solution may be regarded as being so useful (with thereby a higher sense of affordance) that various actors come to share it. This solution thus becomes a boundary object to be crafted by these compartments. The solution sought may not be perfect yet, but as it is

more easily accessed, actors are more able and perhaps willing to craft it together. The boundary object therefore provides an interface through which diverse actors may interact. In short, the process of defining a problem or seeking solutions for it is a process involving the classification of things and the building up of social groups around boundary objects (problems/solutions/artefacts).

Classification is ubiquitous and of great importance in scientific and technological innovation systems, as it is in our everyday activities (Bowker and Star 1999). In the process of classifying objects, ‘even where everyone agrees on how classifications or standards should be established, there are often practical difficulties about how to craft them.’ (ibid.: 46). Classification is a process through which standardisation, naturalisation, interdependence, integration, and interoperability come together. ‘[Classifications] are layered, tangled, textured’ (ibid.: 38). Bowker and Star employ a socio-technical perspective (p. 13, 24, 39, 42) in their methodology treating classifications and standards as material, as well as symbolic (p. 39). In analysing materials that embed classifications, one is able to understand the social processes through which these materials are given meaning and functionality. This is especially true for the ways in which software problems, *as socio-technical classifications*, express searches for a better digital ‘ordering’ and the search for greater functional utility.

Understanding how programmers perceive and classify a problem and find a solution for it is then a key starting point in investigating the software innovation process. As was stated in many of my interviews, the ability to define problems is seen as the essential skill and in one sense the objective of software engineering. As a programmer working in a commercial closed-source company observed, his idea of writing software came from “find[ing] something that irritates me” (TS200602). Another said that software innovation starts from “see[ing] an interesting software problem and try[ing] to find a good solution for it” (PC0602). Problems can be framed in different contexts (and are thereby situated) depending on the material environment where the problem and its identifiers are found. As Borgman (2003) notes, “Multiple types of problems exist, as do multiple types of knowledge that may contribute to solving them.” (Borgman 2003: 99). The following account of the EMACS innovation process will illustrate how problems

were defined and solved by a variety of actors interacting with artefacts to develop a software product/project. But I also show how the establishing of EMACS turned out to have quite unexpected outcomes in the longer term.

## 5.2 EMACS

EMACS (short for Editing MACroS), an editor programme, can be seen as an excellent illustration of the ongoing innovation found in FLOSS innovation over the past two decades. In 1976 Richard Stallman, as noted above, an employee at MIT AI Lab, and his colleagues, wrote the editor EMACS to upgrade the previous editor TECO (Text Editor and Corrector;) on an ITS (Incompatible Time-Sharing System), which was the software running on the AI Lab's Digital PDP-10 mini-computer. In the text-based pre-graphical world that existed before the Apple Macintosh and Microsoft Windows, the editor was a programme crucially important for creating and manipulating text (Moody 2001: 16). Instead of typing commands when editing texts, the TECO editor enabled users to employ macros, command strings for a cluster of TECO programs, which provided a more immediate onscreen feedback for users. TECO had already had the 'WYSIWYG' (What You See Is What You Get) feature named Control-R, written by Carl Mikkelson, which allowed users to enter macros (command strings) and discarded them after entering them. Borrowing the idea from another WYSIWYG editor named 'E', Stallman then brought additional functionality to TECO to make it possible to save macro shortcuts on file and call them up at will. It is said that this improvement was subtle but significant in that this raised TECO to the level of a user-programmable WYSIWYG editor, which later on enabled innovation at another meta level that became the progenitor of FLOSS (Williams 2002: ch.6, p.5). The amended macro function in TECO permitted users to redefine their own screen-editor commands, pass them around and improve them, make them more powerful and more general, and then the collections of user-redefinitions gradually became system programmes in their own right (ibid.). In so doing, users extended the original TECO system by adding or replacing functions based on their self-defined definitions of 'the problem'. Users were not, then, limited by the decisions made or problem-solving approaches taken by the original innovators (Stallman 1998: 2). The extensibility made TECO more flexible for use and in turn attracted

a larger number of users to incorporate the macro function into their TECO programmes.

However, a new problem emerged along with this new feature. While the new full-screen capabilities were embraced vigorously, various customised visions of TECO also led to over-complexity. The most obvious example was that one had to spend more time than before figuring out what macro commands did what in terms of an individual's self-definition of 'the problem' when improving each other's work. Guy Steele, a colleague of Stallman's at the AI Lab, recognised this problem and sought to address it. He firstly gathered together four different macro packages and began assembling a chart that he believed identified and organised the most useful macro commands (Williams 2002, ch. 6: 6). In the course of implementing the design specified by the chart, Steele's work attracted Stallman's attention and led to the latter's participation in this project. Together with David Moon and Dan Weinreid, the four tried on the one hand, to develop a standard system of macro commands, and on the other hand, to still keep the command set open-ended to enable ongoing programmability/extensibility by others. The programme was named EMACS, (or, as noted, Editing MACroS). The name not only 'signified the evolutionary transcendence that had taken place during the macros explosion two years before, [but also] took advantage of a gap in the software programming lexicon short of programmes on ITS starting with the letter 'E'. It is documented that 'the hacker lust for efficiency [i.e. to make it possible to reference the programme with a single letter E] had left its mark' (ibid.).

The distribution of EMACS marked another milestone in the software history. In response to the prevalence and technical opacity associated with the practice of entirely self-defined commands, and to endorse the hacker tenet of sharing information, Stallman set the terms of on which EMACS could be used in the statement linked to the source code when distributing the editor. EMACS, as noted in Stallman's biography, served as a *social contract* that rendered communal sharing the basis of its distribution. Users, on the one hand, were able to modify and redistribute the code; on the other hand, they were asked to report back the extensions they might have made to Stallman so that he could incorporate and distribute those again. In so doing, Stallman strengthened the functionality of

EMACS, making programming with macros more standardised through creating a reciprocal understanding of the written codes through sharing problem solutions, as well keeping the extensibility that macros afforded. Consequently, a library was created for users to load new or redefined functions and to publish and share their extensions. ‘By this route, many people can contribute to the development of the system, for the most part without interfering with each other. This has led the EMACS system to become more powerful than any previous editor.’ (Stallman 1998: 2). Since then, an archetype of EMACS had been established.

### **5.3 Affordance and EMACS: Extensibility and Customisation**

The earlier generation editor had been successful because it provided flexible use with an embedded programming language, TECO. Because of this feature, editing commands could be written in that programming language and users could load new commands into her/his editor while s/he was editing. EMACS resembled a system that was useful for things other than programming, and yet one could program it while s/he was using it. It was claimed to be the first editor that could operate in this way (Stallman 2002: 1). Parallel EMACS-like editors were written for other programming languages or for different machines in the few years following its first release in 1976. For example, EINE (EINE Is Not EMACS) was written in Lisp (the first editor written in Lisp) in 1976, the same year EMACS was released; Multics Emacs was written in MacLisp in 1978; Gosling Emacs was written in C language in 1981; Hemlock was written in Spice Lisp in 1983. These various developments inspired Stallman to write a new version of Emacs in 1985, named GNU Emacs, a part of his GNU project, meant to be a clone UNIX operating system that would be distributed for free.

GNU EMACS, XEMACS and their sort are well regarded today in the social world of FLOSS or in the wider software world (e.g. support for XEmacs has been provided by Sun Microsystems, University of Illinois, Lucid, ETL/Electrical Laboratory, Amdahl Corporation, BeOpen, and others).

This popularity does not come overnight. The wider development of EMACS

has undergone a process of socio-technical construction. The socio-technical network of EMACS has expanded mainly because of the affordance it allows. The features of customisation and extensibility have provided mutual benefits to both the users and developers. For example, the extensibility of EMACS enables one to go beyond simple customisation and write entirely new commands for programs in the Lisp language to be run by EMACS' own Lisp interpreter. In so doing, once the new commands are written, they will be stored in the library so that all can access it and implement it. However, it is worth noting that '... only a programmer can write an extension, [though] anybody can use it afterward.' (Stallman 1995: 13). This statement implies that the powerful features of EMACS depend upon a specific level of expertise and are so limited to competent users. In other words, not every user has the same ability to modify or manipulate the software. To be a principal user of EMACS, one has to understand the language (its technical jargon) referred to in its documentation (e.g. batch language, syntax, interpreters etc.; note that the definitions of these terms in computer science are totally different from what they mean in everyday speech). The practices (of using the programme/software) can only be conducted if one understands how to practise them. One might argue that users with limited knowledge in computing can do some simple customisation, which allows users to change the definitions of EMACS commands, but it was noted by Stallman that this change can be done only "in little ways" (ibid.). Stallman gives two principles of customisation as examples in the manual. One is manipulating comments, and the other is the rearrangement of the command set. The manual on the one hand helps users to get around the software and find solutions for commonplace problems, but on the other hand, it also tells users how to customise the software in certain ways and this shapes the practice of users, especially for beginners. In other words, the software is likely to be used in the way suggested in the manuals for most users. A manual is not dedicated to encouraging improvisation of the software, unlike the motivation driving hacking practice.

The question of how much lay-users contribute and benefit FLOSS innovation is not straightforward. It is actually not so easy for outsiders to access the core FLOSS innovation system, as it requires expertise derived from in/formal training in programming as well as experience gained 'on the job' or learning-by-doing. The following exchange I observed between a chief financial officer (CFO) and a

chief technical officer (CTO) in the same company providing close-source package software based on Linux system illustrates this difference between insiders and ‘normal’ users:

Lewis: I am a beginner [in using Linux]. I am lucky having great colleagues to help me get through my problems. I don’t think it’s easy for an outsider to access the core innovation system. Linux community actually it’s quite skilful. Only skilled persons can contribute something to the community. But I do recognize the idea of Linux. From this perspective, I think I am a member of the community.

Simon: I think a normal user can contribute something to the community as well. For example you can translate documentations...

Lewis: But it’s difficult if you don’t have much motivation [to be a part of the community], and it is also difficult if one do not have enough technical knowledge to write a well-written document.

(SO060201)

Nonetheless, a well-written manual can inspire beginners to learn to programme and to try something out. Multics Emacs written by Bernie Greenberg in Lisp language was said to have such a capability in its enthusing end-users. It is said that

[Multics Emacs made] programming new editing commands so convenient that even the secretaries in his office started learning how to use it. They used a manual someone had written which showed how to extend EMACS, but didn’t say it was a programming. So the secretaries, who believed they couldn’t do programming, weren’t scared off. They read the manual, discovered they could do useful things and they learned to program.

(Stallman 2002: 2)

Here, it is interesting to see how users’ presumptions about their technical

competence and use of software tools could be changed by the manuals. It also conversely suggests how software users might be easily put off from engaging with new systems. For example, when users are told that Linux is more difficult to use than Microsoft Windows, they will probably believe this to be the case. Yet without this warning, they may be more likely to take up and find Linux as usable or even easier than handling the Microsoft Windows operating system. EMACS in particular, in its affording diverse users, is seen to encourage a much wider range of users, even those who do not believe they can operate such a complicated software without programming knowledge. Multics Emacs “gives [people] a chance to write small programs that are useful for them, which in most arenas they can’t possibly do. They can get encouragement for their own practical use—at the stage where it’s the hardest—where they don’t believe they can program, until they get to the point where they are programmers.” (ibid.)

The efficacy and utility of EMACSen have been generally recognised. As an XEmacs user puts it,

XEmacs is more flexible than any other editor I know: it allows transparent multifiles editing; it is easily extendable -- I can modify or create modes if I need to; and has an enormous range of tools such as integrated source code control (CVS, clear case) and programming language specific syntax aware operations. For example: auto-completion, code-block matching, and block commenting.

(AL110104)

Consequently, the socio-technical networks of EMACSen have been expanded. GNU Emacs for example, is now available for Unix, VMS, GNU/Linux, FreeBSD, NetBSD, OpenBSD, MS Windows, MS-DOS, and other systems. GNU Emacs has been re-configured more than 30 times as part of other systems. Other variants include GOSMACS, CCA Emacs, UniPress Emacs, Montgomery Emacs, and XEmacs. Jove, Epsilon, and MicroEmacs are limited look-alikes. These systems on the other hand also have requirements, needs, and visions that differ from the original GNU Emacs. This phenomenon widens the range of what we might see as ‘digital epistemologies’ (ways of ordering and knowing software) and their

expression through software artefacts in the FLOSS social world.

The pattern of affordances linked to EMACS software is not, however, the only factor affecting its social network. There are some specific features of the system that appear important in explaining the popularity of EMACS. As is noted by Bowker and Star, problems and the way they are ordered, managed and classified are not to be defined independently from a specific time or locale, rather, they are defined through ‘a process of assembling materials close to hand and using them with others in specific contexts.’ (Bowker & Star 1999: 288; see also Lave & Wenger 1992). EMACS appears to be a system whose material features enable it to have this ‘close to hand’ quality for users. Users’ preference for EMACS reflects the way in which users see it providing specific types of answers to specific types of problems or needs that they have:

I prefer using emacs for complicated editing/coding jobs because of its features. But I prefer it for simple and quick jobs like config file editing, hacking scripts, etc So it's horses for courses.

(JJ110104)

Because my fingers are friendly with all the commands I need!! I picked up EMACS first and haven't seen the need to change, I especially like the way it handles copy/paste/delete of rectangles within a text file.

(GH120104)

In meeting these specific needs, certain functions of EMACSen become increasingly polished or honed, and developed as users engage with and reshape the software. The following discussion explores the way in which EMACS, its library and its code, are shared among these users, being shaped by as well as shaping them.

## **5.4 Constructing problems and crafting solutions: framing questions (classification) and building a social network of expertise**

[I]t is increasingly clear that knowledge is the constitutive identity-defining mechanism of modern society.

(Stehr 2001: 20)

In light of the EMACS account above, problems play a crucial role in the innovation process. Triggering a problem or perceiving a problem and dealing with it are important tasks for scientists and engineers, and through their resolution help generate innovation. The confronting of a problem provides an opportunity of coming up with something new or different. A problem *de facto* denotes one's perception of the situation, one's knowledge, and skills. Accordingly a problem signifies one's identity as an expert or a novice, for example. As Borgman (2003) notes, the professional or the experienced often demonstrate better abilities in addressing well-defined problems because technical terminology, information resources and material resources and demands can be better identified. Providing a solution makes the boundary of the problem even firmer precisely because the solution implies the drawing in of co-solvers who share a similar approach to the problem at hand. Furthermore, the solution afforded will lead to a sequence of socio-technical effects that will spread beyond the original group.

A problem, as noted above, is an opportunity; and in a basic sense where there is no problem there is no opportunity. This problem-solving orientation is said to be a specific feature of open source innovation:

Software like everything else is driven by a need to solve a particular problem. There are some developers who are innovative for the sake of being creative. I think commercial software is mainly driven by the market and the need to sell a product while open source software is driven mostly by the need of a solution.

(DY011202)

In defining and solving a problem, one employs those materials that are available and negotiate or cooperate with actors within or across the boundary set by the problem itself. There is no standardised path to be followed; each resolution is a result of situated practices and knowledge. This is evident in many areas of everyday life: Lave (1988) observed that instead of applying conventional mathematical algorithms found in textbooks, people ‘perform highly abstract, creative mathematical problem solving’ when shopping in a supermarket (Bowker & Star 1999: 288). Another legend about how ancient Taiwanese calculated prices when making a business deal also illustrates how materials at hand make the problem-solving process practical and versatile<sup>xxxii</sup>. In this regard, each episode of problem-solving is materially-grounded, textured and situated.

Macro commands stemmed from the requirement for a real-time display editor<sup>xxxiii</sup>, which was unavailable in the 1970s when many users found that typing and editing commands with the original editors were too complicated and too inconvenient. This problem had been noticed by a number of programmers. They came up with different solutions to make programming more efficient. Consequently, a few editor programmes with their own WYSIWYG features had been developed, including E. E turned out to be a valuable innovation route through which Stallman could update TECO’s WYSIWYG feature with macros, a set of command strings for a cluster of TECO programs. Subsequently, by managing a variety of macros commands in the programming language TECO Stallman established the basis for EMACS. Here, efficiency was an important parameter in the process of defining this problem. A problem would not exist if users did not recognise the existing practice (composing code with printing terminal editors) as inefficient. As noted earlier, efficiency is regarded as key within the field of engineering; engineers were and are still taught to design efficient technologies. Efficiency is not however, self-evident: Stallman reports that he did not have a strong sense of the need for a real-time display editor until he encountered the ‘E’ programme when he visited the Stanford Artificial Intelligence Lab in 1976. He was inspired by the function E afforded and sought to expand TECO’s functionality in the same way and helped form the group that was to work on a real-time display system. Meanwhile, there were other parallel groups providing solutions for real-time display systems, and their work could become

complementary to the work that Stallman's group were undertaking. Stallman's macros improvement to TECO enhanced Mikkelson's earlier WYSIWYG feature for TECO. As a result, with this greater affordance and functionality, TECO became more popular. The TECO socio-technical network expanded when more people accepted the macro innovations and incorporated them into their own versions of the TECO programme.

It is worth noting that Stallman did not sit down and write the editor system programme immediately after his encounter with E. Instead, he looked up the database and found that Mikkelson had made a WYSIWYG feature for TECO. He then integrated his idea into that. If Mikkelson's work had not existed, we may have seen a different technical option taken, as the 'problem' may have been defined differently. This points to the contingency and localised nature of the innovation process. This process is reflected in many FLOSS developers' own reflections on their systems as seen in Stallman's and Torvald's biographies (e.g. Torvald said he probably would not have started the Linux kernel project if the GNU Hurd had already existed; Stallman said he would not have started to write the GNU C Compiler (GCC) if Tanenbaum had agreed to share his work). Looking for existing material and tools is another common practice in solving problems in software innovation. Software engineering, as in other fields, is built on existing technologies. Programmers typically explore existing databases and see whether any tool is available; if not, they are likely to try to create one to solve the problem.

Access to problems is another issue that needs to be discussed in light of the data from my fieldwork. The accessibility of problems measures the relative ease with which problems can be understood. If one problem entails an opportunity for innovation, an active innovation field should welcome more problems. In a less open innovation system, problems are less accessible, and the boundary is relatively impermeable to new entrants and new ways through which the system can be enhanced. In such an innovation system, problems are less likely to be seen to appear, innovation options likely to be more pre-defined, and innovators sharing a consensus on 'what needs to be done'. On the contrary, I want to argue that in a heterogeneous field where diverse innovator actors are found, more problems arise

or are triggered. If the boundary of the social group centring on the problem is soft, more diverse actors will be included in the circle. There is a positive correlation between the elasticity of the boundary of an innovation field and the momentum behind the pace of innovation because the more accessible the problem is, the higher the level of multivocality existing in the innovation system.

The elasticity of the boundary can be manipulated through a range of educational, legal, political, economic, social and technical means. In the (early to mid) 1970s, software problems were more accessible in the sense that fewer regulations were applied to restrict programmers to access key materials (codes peculiarly). There was a so-called ‘collaborative hacker’ approach, sharing knowledge and improving each other’s work, that encouraged programmers to continually to redefine the boundaries of the problem (e.g. conducting reverse engineering to deconstruct a software to understand how a code was written). As a result, a wide range of software programming tools (languages, editors, compilers etc.) was created in the 1970s (Ceruzzi 2003).

However, the generation of too many problems may become counterproductive to innovation. The ability to solve problems is key to innovation. The more a problem is accessible, as noted above, the more diverse actors will be invited to participate in the innovation group centring on the problem. As there is no single perfect solution for a problem, multiple voices and silences should always be welcome (Bowker & Star 1999: 41). If a problem is presented in an intelligible/perceivable/accessible way, it will encourage more participants to craft solutions, (though there may be nothing wrong with asking a ‘dumb’ question, as the dumbest question can sometimes produce the best answers.) As noted by a number of commentaries, well-defined problems in which the given information, operations, and goal are clearly specified will more likely to have solutions than ill-defined problems (Glass, Holyoak, and Santa 1979; Borgman 2003). Furthermore, such sources argue that an expert can articulate the queries more specifically and completely than could someone new to the domain. This ability to articulate problems becomes one of the parameters that defines expertise, which will be discussed later in this chapter.

As I noted above, while Stallman's innovation was celebrated because of its extensibility and flexibility, it did however create new problems. One that many saw was the sense of a growing *confusion* derived from the plethora of self-defined macro commands, which was seen to eventually work against a more efficient process of programming. In response, another member of the AI group, Steele, tried to provide a solution by charting the macros. Steele's solution encouraged Stallman, Moon and Weinreid to participate in a solution-crafting innovation group. These four who shared the same interest in this project formed a social network of expertise and began to fashion the digital tools that would be seen to provide the solution they were looking for: in this sense the path they took and tools they developed reflected a shared conceptual frame that was 'not accidental, but constitutive.' (Clarke 1998; Bowker & Star 1999: 36). What made the process more intriguing is the relationships and interactions between the actors themselves and the actants (materials), and the transitions – the boundary crossings – that a problem so identified enabled. Boundary crossings can require both getting in (e.g. gaining access to the problems) and getting out (e.g. forging an alternative solutions different from the original one). These boundaries are interpreted or constructed through the problem-solving process of software design. In the process, actors move across boundaries and shift their identities as outsiders or insiders to the core innovation group.

### **5.5 Innovation elements**

Mundane programming practices entail rich socio-technical meanings that are key to understand the essence of software innovation. In this section, I will examine a number of social and technical elements to explain how a social network for solving problems is created, maintained and developed through the interactions between actors and actants.

#### **5.5.1 Teamwork and communication**

Programming or writing codes can be done at an individual level, as one of the features empowered by software technologies. However, team-work is crucial for

producing software of good quality. Not only does software engineering require a division of labour (programming, testing, debugging, maintaining etc.), but it also requires engineers to interact with colleagues. Innovation needs to be mobilised, so isolation is a major barrier to it. In other words, software that is produced should be something that is seen by others as useful, rather than being merely (on its own terms, as it were) 'usable'. In this regard, a comment from Gosper's account describing Stallman's individual style of work is instructive: 'I can see something Stallman wrote, and I might decide it was bad (probably not, but somebody could convince me it was bad), and I would still say, "But wait a minute—Stallman doesn't have anybody to argue with all night over there. He's working alone! It's incredible anyone could do this alone!"' (Levy, S. 1984 quotes Bill Gosper; Williams 2002 ch. 7: 7). The importance of teamwork is again strengthened in a saying in the FLOSS community that 'Given many eyes, no bugs cannot be fixed.' (Raymond 2000). It is however difficult to separate the individual work and the team work as both practices are woven together and have a reciprocal effect on the software innovation process.

This view of teamwork in open source innovation endures today. A majority of my interviewees claimed that their ideas and problem areas were pursued within a team while the rest said that they have formal or informal contact with programmers or users in different sectors/institutions in a number of different ways to develop software. Peter, a programmer at an OSS firm based at Germany, described his experience of working in a team as follows:

When I was a student, I did a few programs on my own. [But] today we always work in a team... I think the problems are becoming more difficult, especially for commercial profits we are doing and it's much easier to develop something in a team because different people have different ideas how to optimise things and the combination of all the ideas is what makes something really good and you can't do that alone.

(PC060201)

Peter's observation in favour of teamwork is a response to the complexity and

heterogeneity faced in the process of crafting solutions. While he refers to the pressure of making a profit, working in a team for Peter, and for many programmers too, is seen as the most feasible way for solving problems, especially in terms of securing diverse sources of ideas and solutions. When a new member of the team is recruited (formally or informally), s/he seems to share some common interests of the group but s/he also brings some differences into the group. It appears that the more actors are involved in the team, the better innovation is mobilised, though less easy it becomes to maintain a consensus. Negotiations about problems become a crucial aspect of teamwork.

The practice of teamwork can be both face-to-face or at a distance, though the former seems to be strongly favoured. Bo, a Danish programmer working for SuSE, one of the top Linux distribution companies, told me:

I think it's always better to sit next to another developer when we work together. But it's definitely more effective. But I don't think you have to it that way. There are a lot of good reasons for letting people work at home and just meet from time to time. And if they can work on their own there is no problem doing it. But do once a while you simply have to meet, I think. You don't have to, but it helps program SuSE, it helps to regain focus, and be more effective again when you go back home. E-mail and Internet Relay Chat (IRC), especially IRC is pretty good for working because mostly it gives you what you used to sit next to... you cannot point at the screen and say what is this, but you can almost do it. You can say can you do it there and then explain it to me. So more or less. But... I don't know, it's a difficult question to answer it's good idea to work at home or not. I think it's a good idea if people want to.

(BO060201)

Working together with colleagues face to face was the normal routine in the 1970s when EMACS was invented. Though the first e-mail message was generated in 1972, e-mail did not operate at a global level until 1983 when TCP/IP was invented to afford a global speech. Given this material limitation, the way Steele,

Stallman and their colleagues communicated in the 70s was unlike what it would be today. At that time, the most practical manner of sharing knowledge and improving peer's works was sitting down at a programmer's terminal, looking into his/her machine and opening up a his/her work to make comments and modifications directly to the machine. That was also how Stallman realised what Steele was doing in his sorting out the confusion associated with the diversity of macros/commands—he recalls, as noted above, passing by his desk by chance and watching what he was doing. Such physical contact in a proximal space (of the office or lab) was the main site of interaction between programmers. When the members in the EMACS team wanted to share work, they either saved programmes to the discs (still then rather huge in size) and swapped them, or they simply sat down in front of the other's machines to do hands-on discussion. The personal relationships in the office became one of the main factors influencing the development of the software.

Here is an example of this, involving Steele's cooperation with Stallman to improve EMACS' print function, which was originally designed by him with a keystroke-triggered feature that reformatted EMACS' source code so that it was both more readable and took up less space. As Steele recalled, 'We sat down one morning. I was at the keyboard, and he was at my elbow. He was perfectly willing to let me type, but he was also telling me what to type.' (Williams 2002, ch.6: 8). As Williams, who reported this, also notes, , 'Steele was used to marathon coding sessions' which was different from Stallman's "intense coding style" that 'forced Steele to block out all external stimuli and focus his entire mental energies on the task at hand', though eventually after 10 hours they managed to write the print source code within 100 lines (ibid. p 8). Such accounts suggest that when two persons work together, one has to adjust to another's working style or that they negotiated with each other. Agreements had to be reached to process the collaborative work, though this might become quite arduous: as Steele said "It was a great experience, very intense, [but one] I never wanted to do it again in my life." (ibid.). The partnership between Steele and Stallman did not however carry on, in part because they had different working styles. Apart from that, Steele's leaving to work for a commercial company was another reason. Steele's departure posed a risk and brought some uncertainty to the EMACS project. Originally EMACS was basically derived from Steele's idea of charting a constellation of macro

commands, with help from fellow colleagues at the MIT AI Lab.

Stallman's way of managing this uncertainty was to open things up—instead of downsizing the social group to reduce the uncertainty and risk. He released the EMACS source code with a condition of use that requested feedback about any modification to the source code, while also allowing its redistribution at the same time. In so doing, Stallman actually redefined and broadened the boundary of the developing team and made the EMACS innovation more accessible. In issuing this social contract, on the one hand Stallman drew users' attention to the extensibility of EMACS, and on the other hand fulfilled his belief in the freedom of information. The condition he put on source code distribution therefore acted equally to engage users with a practical attitude as well as to promote his philosophy and to sustain the culture that he was used to living within the MIT AI Lab (its practice/pattern/habit). He attracted new users by saying that,

Extensibility makes EMACS more flexible than any other editor.

Users are not limited by the decisions made by the MACS implementers. What we decide is not worthwhile to add, the user can provide for himself. He can just as easily provide his own alternative to a feature if he does not like the way it works in the standard system.

(Stallman 1998: 1)

Stallman helped shape the innovation environment in expressing his welcome for open contribution<sup>xxxiii</sup>.

A coherent set of new and redefined functions can be bound into a library so that the user can load them together conveniently. Libraries enable users to publish and share their extensions, which then become effectively part of the basic system. By this route, many people can contribute to the development of the system, for the most part without interfering with each other. This has led the EMACS system to become more powerful than any previous editor.

(ibid.)

User customisation helps in another, subtler way, by making the whole user community into a breeding and testing ground for new ideas. Users think of small changes, try them, and give them to other users—if an idea becomes popular, it can be incorporated into the core system. When we poll users on suggested changes, they can respond on the basis of actual experience rather than thought experiments.

(ibid.)

To help the user make effective use of the copious supply of features, EMACS provides powerful and complete interactive self-documentation facilities with which the user can find out what is available.

(ibid.)

The above articulations meant that more material and social resources were brought into play and the community of practice fostered. Originally Stallman and colleagues merely dealt with the four copies of macro commands that Steele collected. In bringing in more innovation sources, Stallman expanded the social network of EMACS innovation and was able to redistribute the power of the artefacts in the innovation system. As a self-proclaimed hacker, he sought to foster a philosophy of open innovation. As he said:

[E]ven though there was no organized political thought relating the way we shared software to the design of Emacs, I'm convinced that there was a connection between them, an unconscious connection perhaps. I think that it's the nature of the way we lived at the AI Lab that led to Emacs and made it what it was.

(Stallman 2002: 1)

Stallman's statement points to a key aspect of the social construction of EMACS software: the culture of sharing knowledge embedded in the programming practice in the 70s. Most programmes created were based on this daily practice. If 'culture' is defined as a way of living, which is invisible and embedded in our daily lives, sharing and exchanging programmes is such a programming culture embedded in everyday programming. Stallman adopted this position and tried to sustain it against the wave of commercialising software. The weight on 'the hacker culture', which constantly appears in Stallman's writings and speeches, serves to explain the way he designed software in an unconventional way.

To summarise this section, EMACS was successful because it was able to meet the requirements of most users by being flexible (allowing users to define their own control keys). This feature of flexibility reflects EMACS's affordance and enables more actors to move into the innovation process. Whereas TECMAC and TMACS (the first version of EMACS editors for TECO) appeared to be a solution as the real-time display editor for TECO, new problems had emerged. It took programmers a considerable time to understand each other's definitions of commands before they could bring new order to the programme. The temporary equilibrium reached in the TECO innovation system wobbled again. Steele came up with the idea of a standard set of commands to solve the problem. He, Stallman and the others who shared the same view started to craft such a programme. For the sake of durable efficiency, Stallman reckoned the best way to avoid the derivation of new confusions was to have new-defined commands reported back to him. Hence he wrote the terms of use for EMACS to request feedback of new modifications. This social contract was an informal rule, on the one hand drawing on technical efficiency to sustain the function of EMACS, and on the other hand building up a social network to maintain the 'hacker culture', the daily practice of sharing knowledge that Stallman and other programmers were used to. These social and technical practices and expectations worked together to foster innovation within EMACS. Nevertheless, while the social norm linked to innovation gained greater weight given Stallman's later act, some users were reluctant to conform to the obligations. This is one of the reasons why Stallman's GNU Emacs is labelled as a *moral* product regardless of its technical utility (see the discussion on 'holy war' below). People who did not share Stallman's vision went on creating other editor programmes. Furthermore, new versions of EMACS

were created through yet other problem-solving process (e.g. EINE, ZWEI, XEmacs etc. are derived from the need of porting EMACS with other programming languages).

### **5.5.2 Shared interests and trust**

As seen in the story of EMACS, engaging actors in a network is the key to effective innovation in that the range of expertise in the network will affect a group's abilities to solve problems. Since everyone is an expert with regard to some things and a novice with regard to others, a problem/question in an open environment will be answered sooner or later. It is worth noting, however, the actors' involvement in a network is not randomly assembled, but determined by shared interests. Given the proposition that innovation starts with classifying problems, presentations of problems that deliver degrees of problem definition (well-defined problem or ill-defined problem; see Reitman 1964; Simon 1973; Borgman 2003) become crucial for engaging actors in innovation networks. Presenting a well-defined problem will maximise the probabilities of getting useful information. A problem presented also signifies the problem addresser's level of expertise. An actor with higher expertise often phrases her/his questions in technical terms while a lay user often explains her/his problem loosely in everyday, non-technical language.

Hence, the re-presenting of problems, the ways in which they are framed, and redefined, appears to convey expert or a lay user identity and status within the network. This is particularly true in a virtual field. Without knowing each other in person, the identities of actors are represented by and configured through the questions or answers they provide. As a result, an intellectual stratification emerges. Mailing lists or newsgroups for example are typically hierarchical in this regard. In order to maximise the chance of finding peers sharing common interests, people post questions/problems in places where they expect they will be most likely to find their level of expertise. Certain mailing lists or newsgroups thus are specialised and reproduce specific levels of expertise. From my review of the mailing lists of local Linux user groups (on which I report later in the thesis) it would appear that their expertise is normally considered as directed to immediate

local interests. However, as observed, even experienced software developers cannot always answer the questions asked in local user group forums. Some questions are very specific to certain types of hardware and systems, and are not easily answered unless one has engaged with these specific topics before. This suggests that local discussion groups normally regarded as simply low-level technical discussions can be innovative (e.g. in regard to addressing hardware or system configurations); this will be discussed in the next Chapter.

In the following, I explore various threads on different mailing lists to illustrate how a problem was presented in a communicative medium to engage actors with shared interests, and how respondents react and take further actions to craft solutions.

Technologies are crucial for engaging actors in innovation networks. Conspicuously, newly developed ICTs are deeply embedded in the mundane practices of programming, peculiarly in the FLOSS innovation system (e.g. e-mail, mailing lists, newsgroups, repositories, wiki). These tools/techniques are applied to facilitate communications and collaborations (e.g. peer-review) between programmers. The virtual fields providing spaces for discussions on collective topics may be analysed as sites, which offer access to innovation by spanning the boundaries between different social networks. They represent a process of social innovation in a way as Liff and Steward (2003) puts it that "This heterotopian character offers a distinctive new learning context for users through the juxtaposition of practices which traditionally have been pursued at different sites." (Liff & Steward 2003: 331). Stallman's GNU project, as many other FLOSS projects, precisely took advantage of ICTs to target peers to join his innovation network. After Stallman left the MIT AI Lab and planned to write an Unix-like operating system for non-commercial distribution, a range of ICTs facilitated his individual action. These artefacts enabled him to maintain communication with other programmers and even helped to secure some innovation resources when he worked alone. On 27 September 1983, Stallman posted the message below onto the Usenet newsgroup net.unix-wizards in order to invite people who shared the same interest or parallel knowledge to join the discussion. Stallman posted the

following message onto

Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for GNU's Not Unix), and give it away free to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed.

(Williams 2002, ch. 7: 1)

In this message, Stallman revealed his defined problem and the proposition for possible solutions—a complete Unix-compatible software system. Because he had lost institutional support (financial and intellectual) from the MIT AI Lab, he was more likely to find peers interested in joining the social network of developing a new operating system and to engage their attention by posting messages onto the Unix newsgroup, where specific users/programmers with the same interest inhabit. Without knowing who was going to get in touch, the message posted entailed uncertainty and risk in Stallman's project. While Stallman posted this message, he created a social network of crafting a new operating system. If Stallman was able to invite many programmers to join this project, the network would grow and the project would take off. Here, making decisions about which mailing list or newsgroup a message should be posted to in order to attract as many actors as possible, is another form of classification shaping the innovation process. Stallman reckoned the Unix group was the one where he would be most likely to find his target peers. This tendency of looking for peers who share the same interests echoes my previous argument that a shared interest among peers is crucial for the continuation of collaboration. The common interests engage actors to work together, share knowledge and exchange information. The teamwork gets more complex with higher peer participations. However, if the management style stays in a democratic/open way, the boundary of the team will remain soft. This type of innovation is more accessible because open debate within the team is more likely to produce multiple topics to attract actors. The construction of a shared interest or a common topic is resonant with what Law (1986) and Latour (1987) termed *interestment*, through which actors are enrolled to mobilise innovation networks. The working environment at MIT AI Lab in the 70s was similar to this way. Most of the FLOSS projects that rely on virtual collaboration also meet the criteria of

Law and Latourian *interessment*. In contrast, a closed or centralised direction of a project would reinforce innovation boundaries and restrict accessibility to the innovation process. In so doing, a project can be kept under control to eliminate risks and uncertainties generated by multivocality. Following this route, a project will approach closure eventually. Proprietary software is mostly managed in this way.

It is worth noting, however, that open innovation is not universally celebrated in the whole FLOSS social world. It depends on the extent of the openness. Yes—the FLOSS innovation endorses the phenomenon of wiring people and indiscriminating multi-dialogues between diverse actors across social worlds. Nonetheless, because of their business concerns, OSS firms often have to close up their innovation networks. Office teamwork provides more determined solutions, whereas discussions and cooperation in virtual fields, which are less rigid and more diverse, may help in other ways. Developers at OSS firms would like to leverage this assorted practice to get the best out of it, but the majority of FLOSS community projects originate via virtual communications, taking up, packaging and distributing software solutions. In these FLOSS projects, technologies (notably source codes) and project administrations can be so open that there is almost no barrier to access the core of innovation networks. But on the other hand, sometimes an innovation initiative may disappear because the boundaries for an innovation are drawn too softly to be tangible: there is no clear project with which peers might be enrolled. In some cases, projects disappear because the corpus of the idea is too fragile (i.e. the initial problems are seen as ill-defined). In other cases, projects split off into individual sub-networks because boundaries of the original networks are too elastic to concentrate on any central themes (everything is diverse).

In short, FLOSS enables malleable boundaries of social groups sharing interest in similar problems, which makes the innovation more accessible than proprietary software. In this ambience, any type of project is able to have its own voice, and if someone sharing the same interest hears the voice, s/he will join the project. In this regard, FLOSS innovation creates a heterogeneous field with diverse actors and actants. However, this malleable character of FLOSS projects also implies higher

risks and uncertainties in software innovation than proprietary software. When OSS firms arise, managing risks and uncertainties becomes essential for their success. And the management reflects on the practices of their developers and companies' policy (e.g. licences). More on this type of hybrid innovation will be discussed in 6.2.

### **5.5.3 Material culture: Interactions between Human and Technical Artefacts**

Hitherto, I have investigated how a software innovation network is created and developed in terms of classifications of problems, identifications of solutions and common interests. Actors and actants are brought together, interact and negotiate with each other to solve problems. Boundaries of networks, which determine access to innovation systems, vary in different contexts. A successful innovation, as discussed, is the one that manages to bring in as many actors and actants as possible, to extend and mobilise the network as well as handle the uncertainties and risks emerging during the process. To extend this analysis of the dynamics of the software innovation processes, the following discussion will focus on programmers' mundane practices to explore how key innovation sources (both tangible and intangible) are employed. This account is in the tradition of material culture analysis, and is thereby 'as much concerned with how subjects are constituted within material worlds as with how they understand and employ objects (Miller 1987), a perspective analogous to the writings of Latour (e.g. 1993 [b], 1996) on science studies and technology' (Miller and Slater 2000: 3). This account has therefore to be multifaceted and not reduced to one dominant or homogeneous notion of either "hacker culture" or "business strategy". It is anticipated that software innovation processes in different contexts can be understood in this way without falling into the dichotomy of moral/immoral, efficient/inefficient that the typical arguments about FLOSS and proprietary software usually have. Having said that, 'the complexity and multiplicity of these affinities are precisely what strongly impel us to take as our point of departure the way in which a communicative technology is encountered from, and rooted in, a particular place.' (ibid.: 4).

As shown earlier in this section, programming begins with defining a problem (or a request, as written in many programming textbooks) and then working on it (finding solutions). In programming, it often plunges into a series of abstract cognitive activities. As an interviewee programmer put it: “I think about how I solve the problem I have in mind, develop algorithms, modularise a piece of software that I am going to build, think about how I thought that before I actually start to code anything” (PC0602). Whereas this process appears to be nonfigurative, it actually is based on a series of material practices using machines (PCs, laptops), manipulating software tools (editors, compilers, interpreters, browsers, chat programmes, mail programmes), searching and exchanging information (a Google search, an archives search, wiki, blog), contacting people (via e-mail, IRC, ICQ), reading and writing (viewing literatures, making notes on To-do lists, writing documentations), and programming (planning, typing commands, testing). There are various contingent factors influencing a programmer decision which programming language will be used, which bug (problem) to be patched (solved), which programme to be packaged and distributed, which channel on an IRC to be connected to discuss issues with peers. In making these decisions, actors give voice to an innovation ‘repertoire’. A few examples below demonstrate how artefacts are not merely neutral; they are ascribed to situated socio-cultural meanings in the innovation process.

The first example is the Linux ‘kernel’ programme. There was a variety of C compilers available at the time when Torvalds was to implement a C compiler for use on his machine to gain access to a library of C programmes in order to push his development of a clone UNIX kernel programme forward. But he chose to adopt GCC, which was written by Stallman. In choosing GCC, Torvalds indicated he had a potential interest in the concept that GCC embraced, and this might link to his later adoption of GPL for the Linux kernel programme. Such a choice of a material tool helps build a programmer’s repertoire.

Another example is what is recounted in my fieldwork as a ‘holy war’. A holy war happens when different users make claims on which tools are better than other parallel ones. In demonstrating which tools are more powerful, these arguments actually reflect users’ values and ideologies, rather than being purely technically-

oriented. For instance, in the case of EMACS, some hackers find EMACS too heavyweight and baroque for their taste, and expand the name as "Escape Meta Alt Control Shift" to spoof its heavy reliance on keystrokes decorated with 'bucky bits'. Other spoof expansions include "Eight Megabytes And Constantly Swapping", "Eventually "malloc()'s All Computer Storage", and "Emacs Makes A Computer Slow" (Jargon File).

As a number of sociologists suggest, to understand the dynamics of the innovation process, it is necessary to "revise the status ascribed to humans, non-humans and their environments, and more fundamentally to rethink the dynamics of their interrelations by considering the technical system, not in terms of a 'simple' interface or 'pure' tool of communication, but as a mediator of human activity in its biological, cognitive and social dimensions." (Garbay 2003: 2). In this regard, it is methodologically important to ground software innovation within the rhetoric and metaphors found in the interaction between human and artefacts. This is particularly important as software innovation is not a linear process. Many dialogues, information transmissions, conversations and so on take place in this process. Decision-making also reflects distinct contexts, depending on actors involved, the tools available, and milieus given.

### **5.5.4 TODO list**

A programme/software being used is often not the final product; it is a usable tool but still contains many bugs. A programme/software is continuously developed (if the author is willing to keep it in shape). A different version of a programme/software denotes another innovation episode, with new problems found and solutions provided. Errors are not seen as failures but prompts to the search for new solutions. Looking into a programmer's 'to-do list' records one will find that innovation is built on various continuous alterations, small (e.g. a typo) or big (e.g. adding a new function), made to the existing work. An OSS programmer describes his mundane innovation practices:

I start with problem and then create a concept how to solve the problem. But as I write something down how I want to solve it, I have

note of certain aspects what do I think they are important to the solution, which is a very informal rotation of these things, and then I put it into code which I think is similar as program.

(PC060207)

A ‘TODO’ list thus reserves valuable information with regard to innovation. It tracks innovation details, which are often not shown in documentation but very crucial for understanding software innovation processes. These details are either treated as trivial or taken-for-granted, and therefore, are omitted in documentation. It features what the developers would like to add and what improvements they would like to make. TODO lists also act as a channel for outsiders to get involved in innovation networks. While documentation “allows communication to be reproducible, and therefore increases the probability of being followed by other communications” (Lanzara & Morner 2003: 14), the to-do list enables the innovation left behind to be picked up again. A prospective helper willing to contribute to a project always visits its to-do list first to see what can s/he do. Unlike documentation that stabilises technical knowledge, to-do lists mobilise innovation by activating and connecting members in the innovation system with the identified problems. As Eric, a Debian developer describes his opinion on TODO-lists,

If I spend sometime solving a problem written on my TODO-list, somehow I feel loosing the appeal. The items left on my TODO-list, though more difficult, it however preserves ideas. And this idea of “preservation” is quite similar to other activities such as archiving, storing, or sorting, so that the information will be there, easy to find.

(EZ120204)

In this regard, TODO lists reflect programmers’ problem-solving mindset, “which is quite stimulating, and having set a problem that calls for finding a new one instead of solving the one that has been found.” (EZ120204). A smart TODO-list, in Eric’s view, would be “a bridge between problem setting and problem solving that would continuously expand the problem descriptions, more and more, until you eventually are building and coding the solutions themselves”

(EZ120204). Given this account, many FLOSS projects with extremely long TODO-lists addressing all perceived problems actually contain abundant resources for innovation. Based on the same account, Ben, another Debian developer, points out that a bug tracking system (BTS) resembles a publicly accessible TODO-list.

In some ways, a good interactive bug report in a BTS where you have a developer participating with a bug reporter, and (in ideal situations) even with other users/developers who are subscribed to the package in question. You can get solutions -- or partial solutions -- posted in the bug report and it can, in some situations, look a little like the tool your describing. I think a really good example is bugs that are difficult to reproduce on one's own system. Porting issues and locale issues can fall into this category. When you keep that process moving, in this case, through a rich "conversation" that sort of merges the identify the problem with fixing it, it's harder to fall into the "bug logged, bug ignored" mode.

(MH130204)

Ben's view sheds light on the socio-technical function of the routinised bug-reporting practice in the FLOSS innovation process. Each bug is a problem, which provides opportunities or risks for human decision making or problem solving. Spotting a problem represents the opening of an innovation event.

### **5.5.5 Machines and Compatibility**

Machines (hardware) play a key role in mundane programming practice. The tight connection between hardware and software lies in the fact that software can only work on a machine (this is also the reason why in the 1950s computer companies supplied system software as part of the price of a computer, and customers developed their own applications programmes (Ceruzzi 2003: 9)). The compatibility of hardware and software is an essential issue in software engineering. If one device does not work well with existing systems (e.g. one laptop cannot read the data in a USB portable diskette drive), the device will be less likely to be in use, or the user has to configure the current system to make it

compatible (but sometimes because of the driver programmes written by the makers of the hardware it cannot be configured as it is closed source).

The machines in use also denote accustomed devices for users. For example, users of Apple Macintosh machines have different habits and preferences from other users of PCs with the Windows System built in. They also label themselves with different identities because of the machines they use. For instance, users of Macintosh usually see themselves as more unconventional and **smarter** than Windows users<sup>xxxiv</sup>. Moreover, decisions on which systems/software to be used sometimes depend on which machines are used (e.g. whether the machines supports this type of system or vice versa. If the system is supported, one will choose to use this type of system).

In the early days of Stallman's stay in MIT AI Lab, PDP-10 was the machine in use originally running the ITS (Incompatible Timesharing System). However, the Lab decided to upgrade the machine to a KL-10 one, and the system had to be modified as TOPS-20 (Twenex) too. Stallman regretted the loss of the PDP-10 machine when it was replaced (Williams 2002 ch.7: 7). However, apart from the sentimental feeling he had towards the machine he used, a more practical consideration was that he had to shift his interest to different systems (including peripheral tools and programming languages) and alter his habits. This shift denoted a new learning process of new systems and machines, which put Stallman in a novice position (though he was definitely not an innocent user). Owing to the retirement of the PDP-typed machine (e.g. PDP-11, PDP-10) and ITS, when Stallman decided to write an operating system for free distribution to stave off the emerging commercial interest, he had to create a version of UNIX, which he knew little about, instead of copying his preferred ITS used on the PDP-10 machine. It was tactically more practical to have another UNIX system than to have another ITS as UNIX had the advantage of being used on more than just one or two machines from a single manufacturer (Ceruzzi 2003: 339). Therefore, machines (hardware) influence a programmer's decision about which type of software to be produced critically.

### 5.5.6 Programming languages

As noted above, the FLOSS social world embraces multivocality. This attribution also characterises programming languages. Languages symbolise and enable the compartmentalisation of knowledge and the plurality of expression. In the employment of certain types of materials, actors decide which social group of people s/he would like to interact with. If one is familiar with C++ programming, s/he will more likely work with other C++ programmers working on projects written in C++. Because C++ is a highly conceptual programming language, which requires proficient competence of algorithms to manipulate it, projects written in C++ therefore are less accessible for lay users and are by default highly gendered. It is evident that female C++ programmers are in the minority within software innovation systems. The C++ programming environment therefore becomes an extremely male-dominated field within the software engineering world. Programming languages thus act as another parameter establishing boundaries and shaping the programming culture, here in significantly gendered ways. They thereby reinforce the stratification of expertise. This tendency can be seen in the construction of GNU/Linux as well. “Any discussion of UNIX must include a discussion of the C programming language, which was developed in tandem with it, and in which UNIX was rewritten.” (Ceruzzi 2003: 338). Thus, meant to be a clone UNIX system, GNU/Linux has a strong connection with C as well. Anyone likely to be involved in the innovation has to be knowledgeable about C. This technical constraint of the C language in the UNIX system works to reinforce programmers' affinity with it and standardises their programming habits. For instance, many GNU/Linux users have claimed that their previous experience with Unix is the main reason for them to cling to GNU/Linux. This echoes my earlier point about the emergence of particular practice of development for software. To be familiar with the system, they have to be skilful in C language. Accordingly, C also becomes a dominant language in the GNU/Linux and any other Unix-like innovation systems. Meanwhile, people having proficient knowledge in C language also gain their professional status in the GNU/Linux innovation system. In this regard, programming languages as material objects are key features in a programmer's **innovation repertoire**, carrying both the material (applicability) and the social (reflection of programmers' habits/preferences and shape social

relations) effects. They also create a sense of ‘habitus’ in **Bourdieu’s** (1987/1990) term, and impose it on programmers.

Programmers’ experiences also shape their judgements about programming languages. Through their accumulated experiences, programmers get to know under which situation they should use which programming languages and which programming language is handier. Software programming as a problem-solving process requires constant trial-and-error practice. But an experienced programmer can tell easily whether s/he is doing the right job or not. As an experienced programmer describes, “I know whether there's anything wrong with the code when I see it at the first instance. You just don't feel right about it, the way it is presented. I don't have to test it to get to know that.” (RH060201). Moreover, to become a professional programmer, one needs to know at least one programming language in depth. This will help an experienced programmer keep more commands and programming constructs in mind, and save her/him from continually looking up manuals. For example, in C language, the programming construct “=” means *assignment* and “==” means *equals to*, whereas in Pascal language, “:=” means *assignment* and “=” means *equals to*. A FLOSS developer says, “If you write a program in one of those languages, you use these constructs so much that you get to recall them very easily.” (EZ200104). These experiences will gradually shape actors’ programming habits and preferences. Programming languages in use reflect programmers’ judgements, habits, preferences or even ideologies. Programming languages, which consist of symbolic contents, should be seen not merely as algorithm codes but also as social codes. Programming languages or software tools, materialised in versatile forms, sometimes could be indication of users’ identities. When one chooses to code with EMACS, s/he might try to show her/his sympathy with Stallman’s philosophy. It is something that one practices as his/her social identifier in mundane programming. As a result, software innovation systems are co-constructed by both the social and technical factors.

### **5.6 EMACS as a boundary object**

Hitherto, I have discussed how software is developed under a problem-solving

oriented environment where social networks based on a collective interest in problems and resolutions emerge. I argue that software innovation is activated in bringing actors into social networks, which are developed to solve the problems identified. Solutions are provided through socio-technical interactions between actors and actants.

In reviewing the innovation story of EMACS, a variety of EMACSen have been innovated/renovated by diverse actors for different purposes. The functions of EMACS have been expanded and are still expanding. In other words, the affordance of EMACS is sustained. While diverse innovation repertoires are brought into the social network to tackle a joint problem, they also complicate

the situation by defining and redefining EMACS' innovation concept over and over again. If different definitions of innovation concepts cannot be reconciled, a project diverges, as the development of many versions of EMACS show. The reasons for the divergences vary in social scope (e.g. disagreement on Stallman's social contract), technical/material scope (e.g. original EMACS did not run on other programming language than TECO, so a new version of EMACS was designed for other programming language such as EINE and ZWEI for Lisp, developed by Weinreb, a fellow colleague working on the original version of EMACS as well), or other contingent factors (e.g. experimental projects-- just for fun, perhaps). These parallel processes demonstrate the dynamic in the innovation network of EMACS. Facing the challenge of heterogeneity, authors and maintainers of EMACSen try to enhance their legitimacy and uniqueness in providing greater socio-technical functions.

In this account, EMACS, as an extensible editor that can be used flexibly, has served as a boundary object for diverse users. They interpret the role of EMACS in different ways according to their situated practices. A number of common practices of EMACS can be summarised. Some take EMACS as a pure tool for editing texts, programming, gaming, web browsing. Others contribute to the FLOSS community by reporting bugs of EMACS while using it, and still others residing in the core of EMACS innovation system take EMACS as an artefact or even art work. Because EMACS denotes so many different meanings, the diverse

interpretations and manipulations of EMACS can prevent it becoming a stereotyped editor, even though its material effects (as affordances) do give it a specific technical character. This is in contrast to other proprietary software products. For example, Microsoft Office Word software has enrolled a huge number of users for processing their Word documents. When mentioning “*Word*” people take *Microsoft Office Word* for granted. Accordingly, *Microsoft Office Word* is used not more than for typing and word processing. But for EMACS, its high affordance enables many roles to be played in mundane computing practices. Once EMACS is mentioned, people would like to know which version of EMACS is indicated (e.g. GNU Emacs, XEmacs—each indicates different usage habits, particular interests in programming languages, or even political views), and for what purpose it is used. The potentiality of EMACS for different functions is resilient. It is also this trait that keeps open the innovation of EMACS. Unlike Microsoft *Word*, users can customise and configure EMACS to meet their requirements. In other words, in constructing their own EMACSen, different social actors (institutional, organizational, or individual) create different constituencies “differentially situated within actors’ multiple everyday activities, rather than arriving as a self-identical technology with definitive effects upon traditional lifeworlds” (Hand 2003: 330).

Since EMACSen are used in many different ways and denotes various socio-cultural meanings, diverse projects have symbolised users’ habits and preferences (socially and technically). In adopting specific tools and participating in specific projects, users are attached to the artefacts. These artefacts grow to be norms to demarcate boundaries. For example, the users of GNU Emacs see themselves different from those of XEmacs, while the broad range of users of Emacs distinguish themselves from other users of other editor programmes, such as *vi*. Holy wars happen often because these users want to strengthen their boundaries against each other to confirm their identities. It would be interesting to see how these projects are symbolised as norms, how they are interpreted and used. That is, in the course of explaining the heterogeneous FLOSS social world, one can study the socio-technical meanings given to various projects to understand “FLOSS”. This actor-centred view may provide a distinctive research result from the prevailing structure-centred or essentialist approaches found in other FLOSS studies.

Life in EMACS will continue to change, but the crucial presumption will be whether the boundary of the innovation network can be maintained to sustain the innovation. For instance, Gosling did not continue to share his codes, rather, he sold his EMACS version to a commercial company. His version of EMACS therefore did not continue to grow. This is not to say that selling software to a commercial company will kill the product. There are other reasons for the elimination of a software product and sometimes the involvement of commercial companies does provide other sources for sustaining or developing a product. But in the case of Gosling EMACS: 1) He thought selling the product to a firm might broaden the network. But the transaction symbolised Gosling's failure to continue the network expansion. 2) The commercial product was not successful. The firm failed to engage actors' interest in it. The commercialised software forms a firmer boundary than FLOSS community projects to exclude outsiders of the developing team from the innovation. In that case, problems will not be triggered that easily. If problems are the initiatives of innovation, a less diverse character in the team will not help the innovation at that point. When GNU Emacs was just invented and still in an unstable stage, it did not put users off, instead, the existing problems invited peers to tackle them. GNU Emacs was able to engage actors in its social network of innovation. The network expanded and a variety of functions were developed. These functions become cornerstones to attract more actors/users as in a snowball effect. Unlike some proprietary software firms who try to lock users in by using proprietary document formats (and critics say they do this in order to dominate the market), EMACSen engage users by presenting greater shared interests in socio-cultural or technical aspects. The story of EMACS sheds some light on the FLOSS innovation, though its commercial dimensions should be taken into account in order to understand the situation completely.

### **5.7 Conclusion: The heterogeneous FLOSS social world**

[FLOSS] is a multiplayer game that enables them to visualise this, and play-learn like they play-learn rent/mortgage etc with Monopoly. This is a human catalyst-rich mind exchange. The human catalysts are key. So multiplayer games is one catalysis method.

## Chapter 5

---

(Bala110304, mailing list minciu\_sodas\_en@yahoogroups.com)

Linking back to Chapter 4, in which I discuss the heterogeneity in the hacker social world. I propose that the hacker social world is not a closed community with firm boundary; instead, it is a community of practice where mutual benefits are produced through engaging with shared practices and shared repertoires. Members negotiate the meaning of the shared practices and boundary objects. The boundary and the boundary object/practice become the catalyst/medium between members for their interactions. Consequently, I offer some exemplars to show the affordance of technologies that allow more actors to participate in the social world and stimulate more interactions.

In this chapter, I argue the FLOSS innovation exhibits a similar course. While the FLOSS community continues to grow, diverse actors (e.g. developers, firms, end-users, organisations, governments etc. just to name a few) are brought into play. Meanwhile, a variety of apparatuses and inscriptions (e.g. technical ones such as software and hardware tools, socio-economical ones such as licences, educational ones such as certificates, and socio-cultural ones such as on/off line discussion forums) are developed and employed to maintain the practice. The complex composition of the FLOSS community entails a heterogeneous field where innovation is socio-technically constructed. Practices and norms in the FLOSS community are articulated and interpreted differently in support of individual demands (social, economic, political, technical) of the actors. Such a heterogeneous world resembles an ecological system that contains diversity while resources, both tangible and intangible, are commonly shared amongst actors. This is reflected in a number of new methodologies of software development such as the 'Extreme Programming' or the "Agile Development"<sup>xxxv</sup> that start to pay attention to the values of individuals and interactions over processes and tools as well as customer collaboration over contract negotiation. The "tardy, wasteful and bureaucratic" software development practices, as said in the manifesto of Agile Computing, have become an issue in software engineering.

This chapter contributes to our understanding of the formation of knowledge in the Internet era, where information and knowledge flow fluidly and rapidly. The

EMACS case denotes various key factors involved in forming cosmopolitan knowledge: how actors network together (e.g. shared interests), how they interact with one another (e.g. problem-solving process), and how local epistemologies and tacit knowledge are translated into cosmopolitan expertise in an in/tangible form (e.g. materiality of hardware or software). I believe this empirical enquiry will provide us with a means of retaining the holistic and meaningful characteristics of real-life events. Methodologically speaking, the contextual thickness makes a case study appropriate for both "how" and "why" research questions because answering these questions deals with operational links needing to be traced over time. The detailed investigation of FLOSS phenomenon with attention to its context by using multiple sources of evidence and various methods of data collection helps to examine the innovation process by which new FLOSS technologies are created, arguing that this is ongoing and involves diverse groups who give the technology different meanings. This perspective also reflects an ongoing thinking in STS that technologies, no matter their designs, uses or applications, are not independent from social factors. Given the history of EMACS, one can see how hacker ethics emerged, developed, and followed in the innovation process. Based on the hacker ethics, EMACS, or a wide range of FLOSS, are not only a technological revolution, but also a social movement that operates largely in terms of symbol and meaning, both at the level of everyday life and at that of institutional operation. In the next chapter, I explore this process in greater detail.

## **Chapter 6 Hybrid Innovation: The Dynamics of Collaboration Between the Public and the Private in the FLOSS Innovation System**

### **6.0 Introduction**

The chapter further explores how a technology develops its affordance in institutionalising an emerging range of collective practices in a variety of ways that codify and regulate them (e.g. licences, market distributors, standards etc.) yet still allows actors to meet diverse needs and interests, particularly those that are of a commercial and non-commercial nature. I want to describe a hybrid innovation pattern in regard to the development of FLOSS to illustrate the collaboration between those engaged in the social world of hacking and those located in the commercial world.

Following on the socio-historical analysis of EMACS in chapter 5 that examined the development of the social world of FLOSS as it expands and develops over time, this chapter moves on to explore the practices of those developers at OSS firms (mainly SMEs providing Linux-related products and services) to demonstrate the material practices of software innovation and the mutual help between the wider community and the firms. Whereas the more recent commercialisation<sup>xxxvi</sup> of OSS plays an important role in giving open source innovation greater momentum, a strong tendency towards problem-resolution is found among software developers across social boundaries. Their material practices are employed together with the OSP(s) (open source practices) to solve problems. Relationships and interactions between the actants and the actors are embedded in these material practices. Actants and actors together form a seamless web in the FLOSS innovation system. While socio-technical networks are formed to solve problems, the boundary of each social group is still well-drawn. Though these demarcations exist, boundaries in the FLOSS social world are however softer than those in other innovation systems, and under certain circumstances they can blur, but as I go on to argue, such blurring and lack of focus of boundary translates

into a lack of focus on innovation problems. This in turn, I suggest, creates problems of its own: the socio-technical heterogeneity works against the practical need to generate solutions that are needed, especially, as will be seen, in a commercial setting.

In empirical terms, this chapter draws on my visits to the Linux conferences, a series of interviews with respondents, and virtual fieldwork via email and the web in order to build an account of the FLOSS social world, and its link to the wider hacker social world. In light of the empirical data, it will be shown that the FLOSS innovation pattern is not markedly different from conventional software engineering in rendering proprietary software through writing codes. The uniqueness of FLOSS innovation, however, lies in its hybridity, combining the resources and practices from both the community and the commercial sector. In the next Chapter, Chapter 7, I also examine the ways in which the extensive and increasingly codified open source community associated with a more strongly institutionalised FLOSS is accompanied by, and indeed, effectively helps reproduce and is dependent on a localised, non-codified world of innovation practice. Unlike previous research highlighting the commercial-community hybrid in the FLOSS development (e.g. Mockus *et al.* 2002; Bonarccorsi & Ricci 2003), I want to argue too that even when the OSP(s) gets institutionalised this does not necessarily create a stabilised innovation process. Instead, while the processes associated with what I call hybrid innovation, this diversity and heterogeneity provides renewed sources of innovation in the field, and the mobility and fluidity of actors and actants (e.g. information, tools) across the community and the commercial sector works to foster new forms of software.

### **6.1 Beyond Game Theory**

Unlike the conventional view that attaches importance to the private sector for developing technological innovation, which is then later absorbed by the public domain, the development of FLOSS, in contrast, appears to be a community-led innovation that then enrolls the socio-economic interests of the private sector in its innovation network. Their strategic collaboration has been examined from some relatively normative or instrumental perspectives such as game theory as

developed in political economy. Here, I challenge this account and emphasise 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 actors' interests, assumes that actors are rational interest-maximisers. Accordingly, collaboration between the private sector and the community in developing open source software is based on this premise. However, I would like to argue that this approach is of limited value, particularly because of its basic assumption that all actors are rational interest-maximisers. This makes it difficult to detect the heterogeneity and contingency of the innovation process. Moreover, game theory also tends to see the innovation process as a linear cause-and-effect dynamic: in contrast, I have argued in the thesis that this process is shaped by actors' diverse perceptions and acts of interpretation, negotiation, disputation and witnessing of the problems and solutions they confront as these vary according to their socio-technical backgrounds and positions. As the history of the IT industry shows, the evolution of its knowledge base is catalysed by the actions of a diverse set of agents, who are driven by a diverse set of motives, resulting in activities that are both creative and destructive. "The heart of this process of creative destruction is the epistemic cycle of uncertainty, imagination and innovation." (Jackson *et al.* 2002: 329). Consequently, this highly fluid, flexible and contingent innovation pattern I suggest, draws our attention to the socio-technical heterogeneity found in FLOSS culture and practice. Boundary objects such as source codes, programmes and projects are actively crafted through a process of 'mutual enrolment' of actors. "[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 heterogeneity of practices that both embody and embed the interactive and temporal stabilisation of a range of cultural elements in the innovation process will be the central focus of the following section.

## 6.2 A Community of Open Source Practices (OSPs)

As discussed in chapter 5, in the FLOSS innovation system, the socio-technical interactions and relationships between actors are anchored in a range of everyday practices. These include talking with other developers or users of an IRC, sending messages onto mailing lists, writing one's 'TODO' lists, choosing licences for one's work, and so on. The FLOSS social world, constructed and shaped by a set of material mechanisms and daily interactions, epitomises our late modern society, which is 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, inasmuch as it sustains the heterogeneity and preserves diverse innovation resources. Boundaries, however, do not appear to stop members from travelling between groups or communicating with each other. In the FLOSS social world, boundaries enable rather than prevent connection between 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 localised identities. 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. This explains why the FLOSS innovation system is both dynamic and relatively unstable.

One particular reading of this might be to argue that the FLOSS innovation system is a good example of a system that “includes issues of social justices, multiple interpretations, and adjudication of conflict across social boundaries.” (Star, Bowker and Neumann 2003: 242). As I discussed in Chapter 5, the innovation of the EMACS editor programme not merely epitomised the value of iterative *technical innovation* (i.e. the editor was customised and configured to meet various needs and purposes), more importantly it represented a *social innovation* – a social movement initiated by Stallman in the late 1970s built on the advent of the GNU GPL, regulating the liberal re-distribution and modifications of software products and their derived works as well as formalising social networking amongst developers and users. The former has facilitated the transmission of

information without locking software source codes in the black box, while the latter secures the innovation resources (mainly the skills and ideas) for software projects.

It might be argued too that 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 is 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 accelerate the speed of ongoing construction. These open source practices<sup>xxxvii</sup> (OSPs), most commonly found among FLOSS developers, also correspond with other practices on the Internet and related ICTs, such as sharing information via E-mails or by text on mobile phones, contributing knowledge to wiki sites, making weblogs to document experiences, and so on. These behaviours might suggest that computer-mediated communication, data-archiving and resource sharing make the construction of knowledge more efficient and more effective.

In stipulating how software is derived from earlier works and enabling its further licensing, actors in the FLOSS community of practice are able to enrol shared interests and skills to help resolve a problem or complete an existing project. Although deviation from the project might take place, the socio-legal constriction reduces this possibility and thereby the risk and uncertainty that could characterise an open source system. Being able to maintain consistent and traceable contributions to and interests in EMACSen is one of the reasons for its currency 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 informally managed and even expanded. In this regard, apart from serving as a normative constraint that helps to support the FLOSS, over time the GNU GPL also offers a formal/legal basis to maintain the mechanism and operation of the OSPs in the broad community of practice.

Consequently, GPL, as a sets of rules that regulates the distribution of knowledge and the access to it, “tends to stabilize behaviours and dispositions that are specific to that rule-set, namely the actors’ willingness to reciprocate”

(Iannacci 2003: 21). In other words, GPL, also known as 'copyleft', "induces and stabilizes an *institutional environment* for conversations and unbounded communications. It embodies assumptions about knowledge as a public good, its generalized accessibility and usability, the value of geographically dispersed skills, and finally about how knowledge should be created and used in an open society and in a user-driven innovation process." (Lanzara & Morner 2003: 22).

There has been a growing recognition of OSPs in the wider computer world, where they are regarded as of help to software innovation (see the UN report 2003 UN Press Release TAD/1967). However, there are arguments about the extent to which the source codes and information should be fully open. Debates about these topics swing around the materialisation of OSPs (i.e. *qua* licences) in software production. While the OSPs appear to engage diverse actors in the FLOSS social world, the existence of licences, and the degree of commercialisation that these allow, challenge notions of 'free' software. In the case of licensing, some actors contend that the GPL rule is too strict to motivate potential adopters, and others question the negative influence on innovation of the political issues raised by the GPL. For these reasons, various types of licence have been proposed with different terms and conditions to reflect different views. Individuals or organisations can make clear their respective position on 'openness' 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 seem not to want to get too much involved in such 'holy wars' or political issues.

Whereas the OSPs originally emerge from a good will of a community-based gift culture, the idea of commercialising open source software is derived from economic self-interest. Given these two conflicting incentives, 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. Even so, while the tensions seem to be quite marked, collaboration does arise, and so it is important to explore how and why actors decide to collaborate. It is likely that this will be different from market-based forms of social coordination built on prices and economic self-interest. In taking social and cultural factors into account, we

need to focus on problem-identification and the social network this creates as problems are solved or artefacts created to enable this. This social network is also enabled by the existence of publications that play an important role in reproducing the FLOSS social world, such as the Linux-Magazine. The editors of Linux have declared that, “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 editor’s theme message).

### **6.3 Working practices in OSS firms**

Within the context of falling hardware prices and more reliable, compact, and standardised computer systems (Ceruzzi 2003), FLOSS has itself joined in the trend of commercialising software during the 1990s. As observed, the commercialisation of FLOSS institutionalises OSPs and widens the ‘market’ for FLOSS. In this process, firms (mainly SMEs) that support OSS development (for brevity, I will term this type of firm an OSS firm) operate as a combination of conventional companies adopting commercial business strategies yet sustaining OSPs. My interviews with OSS firms made clear the importance of virtual communication through the FLOSS community as the major channel through which developers at OSS companies communicate and exchange innovation resources (e.g. tools, ideas) within and outside the companies themselves. With a large number of peripheral members online, virtual communities appear to generate a digital market place and a learning community that provides enterprises with a road map for business. Hence, allowing developers to engage in virtual communities helps such firms to expand markets. There does appear, however, to be a cost that must be met in doing this, since the engagement with the wider FLOSS social world leads to a higher information flow than would be found within conventional ICT firms. The routine work of any company must also be attended to – meetings, paperwork, debugging systems, and so on. Though these unavoidable routines may be seen as burdensome, working in an OSS company appears to be regarded as more informal and with a higher degree of freedom than would be found in the conventional commercial sector. Brian, who works at a leading OSS firm observed,

When I do the routine work I don't always have to be at my desk. I get to decide when I get to do the things and also I have a lot of influences on what I do because we never get told 'you have to do this'. It's always we have these things that need to be done, who can do them? Then people can just volunteer. I mean, we always have plenty of things to do, so it's just the question of picking the one that is interesting to you, which is more or less the same thing you do when you work in your free time. So there's another freedom both in one I work and when I work home. But still then choices are of course made by my employer. I am not that just sponsor to sit and do whatever I want. There are some but not me.

(BO060203)

There are then constraints on what those working within an OSS firm can do and where their priorities should lie, despite their subscription to and links with the broad OSS community. The concern about profits, especially in smaller, cash-starved firms, becomes a main factor influencing a company's decision-making towards innovation. Most interviewees mentioned time pressures, the influence of investors and clients and available financial resources (e.g. a bank loan) as crucial factors shaping their innovation strategy. Their activities can no longer be 'just for fun', as often described by community members. Unlike working in an informal software context where shared interests provide the main motivation 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 more bounded social group, working on specific projects, with specific colleagues. Even so, I found that many work contents appeared to be extensions of previous (informal) community projects parallel to the firms' current ambitions. One developer is often hired through their links with other developers because they had worked on the same OSS problem in the wider FLOSS community and knew each other through that. Joining a firm for developers, therefore, signifies a shift towards a more formalised/institutionalised working partnership. The previous trust and tacit understanding between the developers may be enhanced as a result. As many interviewees working in firms told me, if they have problems at hand, it is their immediate colleagues whom they will go to for help rather than looking elsewhere,

such as posting questions on newsgroups or discussion lists. Though the exchange of information through instant messengers (e.g. IRC or ICQ) is fast and acts as an effective way of getting quick answers to questions without the need for multiple back-and-forth exchanges of e-mail, the problem is likely to be framed in such a way as to make most sense within the commercial context of the OSS firm. This will mean, for example, that security interests have to take precedence over more creative software activities (e.g. see “Programmers told to put security over creativity” on April 1 2004 C|NET News.com). At the same time, software developers within OSS firms see colleagues as a readily available and expert resource through which solutions to development problems can be found. Klaus, a desktop developer for a leading OSS company, comments on this within his own company:

Because of the people on the internal mailing list are so brilliant, 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 setting up such an internal network of developers, the expertise in the firm is centralised. The expertise of developers is concentrated in a bounded group and their interaction and relationship is strengthened formally, and in so doing trust between colleagues enhanced and problems dealt with more efficiently. With such a strong development team (expert-oriented), a firm is also likely to be in a better position to convince its customers of the quality of its products and services.

It was also suggested by my OSS respondents 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 habits radically and tend to stay with existing interfaces, a key element to make a successful software package for users is to make them “feel at home”. Expressing concern about the lack of new applications for end-users/customers, Klaus comments on 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 very very differently, then you have to find something that really [is] 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 [a] user level graphical tool, or it is [a] command line tool, doesn’t really matter. People have to feel some point at home. Which is why 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)

A number of respondents commented on the two-edged nature of FLOSS software, that, on the one hand enables flexibility, yet on the other how this very characteristic makes its stabilisation and homogenisation within a *single package* more problematic, especially thereby for customers. In regard to flexibility Klaus observed

The biggest strength of Linux is that you have all the flexibility. You have the good thing about free software and, it’s a system that can

work for almost anyone. GNU Linux is an open system that no one can decide for you which direction that's going.

(BO0602)

And others made similar comments about this technical affordance, noting,

...you can configure almost anything just as you want it, you spend your time on finding how it work, and something that for example in Windows it's almost impossible because you don't get all the information you need when you would need it to configure in another different way.

(PC060204)

Such flexibility though can create other difficulties:

...[But a] bad part is that there are not enough good applications. The core operating system is extremely good, extremely reliable, and it's definitely better than almost any other system out there when you are talking about general perfect operating system. But the applications are not to the point that Windows applications are. And that's a very big issue. That's the issue we here in SuSE when we are talking to customers. 'Well, can I get much software exchange functionality.' And then you said 'yes, you can.' 'Can I do this, this and this?' and we said, 'yes, if you take that package you can do it. And if you take that package you can do something else.' And the problem is there are too many different packages that can do some of all of it but not every package that can do everything.

(BO0602)

And as Klaus went on to say:

If someone decides on the trends, directions, then someone else will go in another direction. Which is also a problem in Linux, because

this means that you have a... sometimes an unseen amount of flexibility. And flexibility is hard, is complicated. So I think the biggest problem in Linux right now is that it is still a bit complicated for a person who can only write Word documents to access the system. But I don't think it will get that much easier to set up a Linux system, a bit easier.

(BO0602)

So the affordance of FLOSS seems to both enable its broad and multiple development across different fronts, yet thereby makes its stabilisation more difficult.

#### **6.4 Linux Conferences: A Bridge between the Corporation and the Community**

Given the dominant rule in software engineering of understanding users' requirements, communicating with clients and knowing what they want is vital. If clients can understand the concept of open source, it is believed by OSS firms that they would appreciate the product more and have a better service. As James, a CEO in a FLOSS SME argued in describing 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, such an open-minded and clever person. [Unlike] CEOs of small companies [who] 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 more perfect.

(JP060204)

Where to find such an ideal customer? Personal connections are clearly important to many respondents: as Paul, a co-founder and CTO of an OSS SME observed:

[Knowing people] is extremely important, especially when we are small company. It's very important that we know people in another companies who can give us projects or who can just give us contact to more important person in another company. That's essential because we small company it's very difficult to find entirely new customers or to whom we never have contact before. That's almost impossible.

(PC060205)

Linux-related conferences or virtual platforms also appear to be a valuable venue my respondents used to make contact. Particularly at conference sites, developers or firms sought to identify 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. the free conference at the Linux Tag) with a less commercial/business atmosphere turns out to be even more useful for developers or firms to build up good relationships. In this environment, participants established informal personal contacts with people, among whom some will become customers and the others will become colleagues. These conferences appear then to provide a platform for diverse actors to meet and interact in person. Respondents report that 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 then become important vehicles for establishing social networks in the FLOSS innovation system. Attending conferences serves as an informal and alternative way of seeking out prospective collaborators (i.e. employees or clients).

Meeting at conference sites makes *physical* the virtual communication of various newsgroups, mailing lists and IRC channels. Since some of the participants reported that they have known each other or worked on joint projects virtually, the trust between them was reinforced when they met face to face. OSS conferences and meetings thus secure opportunities 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 also act like a boundary object that brings diverse interests together under the broad banner of OSS development.

## 6.5 Pragmatic Collaborations in Practice

As illustrated in many SME interviewees' observations below, the commercial sector sees its links to the FLOSS community as of broad value:

I think the company should go closer to the Linux community. We would like to keep some information confidential for business, but we should not forget the open source ideology. Because of open source, so there are we.

(SO060202)

The community is actually necessary. It is. Because that's the form to share interests and to communicate about to get information passed along, [in a] very broad sense.

(WP060202)

This interdependent relationship can be illustrated in a number of ways: 'bug-reports/patches', open networks as sources of new ideas and problem solutions, and OSS as a medium through which status and identity can be built.

Because software is a form of language that is never 'finished' or 'perfect' and often alters when applied to different hardware systems or infrastructure, its

constant reworking is crucial to keep it in a feasible shape. Bug-reports, feature-requests and related technologies (e.g. bug tracking system (BTS)) have been developed as practical solutions in response to this innate software problem. Since source codes are freely distributed in the FLOSS social world and its innovation networks are more accessible than proprietary ones, actors can download newly-developed programmes/software, try out, spot some bugs and report them back to authors or maintainers, or even write patches for them. These patches are then incorporated into new versions of software, if they are applicable. This bug-reporting mechanism has evolved through a mutual-help culture in the FLOSS community. A common practice is that authors make an announcement of the release of her/his *experimental* projects and invite downloading and feedbacks (i.e. bug-reports or patches). When asked whether he would like to use an unstable web browser *Galeon*, which hangs often when opening up flash links, Eric, a Debian developer explained to me that he uses this programme to report bugs. In so doing, he can help improve this programme and encourage innovation in the browser technology. It is estimated that around three billion dollars worth of hours have gone into this constant upgrade process at the heart of the Linux operating system. And this figure does not take into account the millions of hours that have gone into the thousands of applications that run on Linux (Cancilla, 2003). Working closely with the community thus can save firms time and money in detecting bugs and writing patches (though they still have to do this as routines to ensure the quality of their own 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. As one of my firm respondents, Paul, says,

Linux community input mostly comes in the form of Linux distribution, for example, or software libraries that we can integrate into our customer software. And that respect is very important for us. These libraries are very helpful on Linux distributions.

(PC060208)

Although this mutual support is helpful for developing both firms and FLOSS, it does contain some risks to the innovation process, particularly because of the

informality of the working relationships which may mean, in the absence of any formal requirement, shared exchange comes to an end. However, given the scope and scale of the FLOSS community and network, it is possible to receive news on bug-reports etc. from diverse users addressing specific needs, or offering their interpretations of problems and solutions. Management of the contingency and heterogeneity of this process thus becomes essential for firms operating along the boundaries of the FLOSS system. Consequently, OSS firms formalise this bug-reporting and patching approach in their business practice while keeping channels to the innovation system open. It is even more standardised for firms dealing with patches through 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 maintains software products of a consistent standard. Nevertheless, this also implies that prospective contributors have to be wary of the rules if they are to offer patches. It is a process of technological standardisation. It also implies that the process of innovation is not completely open, and has too a degree of path dependency.

The interdependent relationship between firms and the community is also marked by open networking which enhances serendipity in resolving innovation problems: as one of my respondents observed,

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

(LT060205)

It is recognised that brainstorming and experimental practices in the community (mostly taking place on the Web) stimulate and cultivate the FLOSS development, and in this sense are seen as valuable sources of 'intellectual capital' drawn on by actors in the FLOSS social world. James emphasises the networking function of the community:

The social point of view on free software development, writing creative scripts and sharing them, is basic. You get many friends when you do free software. And the relation is not just Internet and sharing but there are communities of people [through which] to get friends.

(JP060204)

Through such networking, not only do the community's radical innovators with experimental ideas have a forum through which to display their talents, but so too do those with more incremental and modest ideas emerging from participants' experiences and skills in software development. Collaboration also provides opportunities for more informal friendship ties.

Apart from open networking, another function of the FLOSS community is that it can act as a medium through which software innovators can build a recognised status and identity. Just as employees working for firms are identified through their companies' names, in the community, FLOSS developers identify each other according to the projects they are working on. The projects (artefacts) symbolise one's identity and become signals in the wider community of the sort of skills one has. Some projects gain especial prominence because of the significance they have for OSS, and if one is working on one of these one gains wide recognition. If one contributes to more than one of such projects, the social status this brings may mean that the person is more likely to be hired by a firm, especially a larger OSS firm. Therefore, securing such an identity and status becomes one of the incentives for developers to participate in the community-based innovation. By giving away something which is well-made, developers will gain recognition from those who download their work. For some people, "the gift economy" is the best method of collaborating together in cyberspace (Barbrook 1998). The attraction of such a reputation game is portrayed Eric Raymond's well-known writing "The Cathedral and the Bazaar", and is figured prominently in the FLOSS developer survey of 2001 (FLOSS report 2002), and is mentioned in several FLOSS-related writings (e.g. Kelty 2001). As an IT consultant working at Informatique CDC noted the reputation game can be very important:

## Chapter 6

---

- a- build something I need for my current projects, or tech watch
- b- be "fair" and share back, to thanks the community for past help...  
make the system work... (If I dont reward helps, soon there will be no help)
- c- get support from a community
- d- (to be honest) get some "reconnaissance", i.e. just be known as competent and helpful

(AC21202)

The discourse of gift and reciprocity is given further expression through the rhetoric of ‘Corporate social responsibility’ (CSR), which is deployed particularly in OSS firms, and working with the community seems to become a key ethical norm in a firm’s strategic agenda. CSR, together with financial performance and environmental practices, have become the top three goals for big IT companies (Brennan & Johnson 2004). Working with the free software community seems to be a perfect technology management tactic that helps a firm achieve both competitive performance and social responsibility. In integrating the fulfilment of corporate social responsibilities with technology-driven strategies for keeping products competitive, this ethic provides the basis for new products, and changes operational conventions in a firm. Thus, investing in open source software or working with the community gives corporations an ethical image, which might enhance their market. In their strategic collaboration, one sees the interplay of the community-based hacker ethics and the industrial-led business ethics. Apart from other tangibly technological benefits, the community-based hacker ethics, embedded in the everyday practices of developing FLOSS as illustrated in chapter 5, entails an intangible advantage given to corporations for long-term, macro-level software production.

## 6.6 Shared Goals and Divided Goals

As seen above, the relationship between firms and the community in developing FLOSS is highly interdependent. Over time, the public and private dynamics of the FLOSS innovation system begin to shape the systems broader character: as McKelvey (2001) argues, FLOSS,

... becomes more and more like other dynamic, knowledge intensive industries. In that sense, the dynamics of software development are likely to rely on parallel processes of commercialisation and science [e.g. the early basic scientific and commercial uses of genetic engineering]. They will rely on both the overall production of public knowledge as well as on the closing off of parts of knowledge production within the firm in order to capture economic value.

(McKelvey 2001: 34)

Given the flourishing OSS business model, more and more commercial enterprises want to leverage its innovation components, tangible or intangible, generated within and across the FLOSS social world. Some big companies such as IBM, HP or Novell have invested in a number of open-source activities and supported Linux operating systems with their products. The technical potentiality of FLOSS features strongly in these corporations' material, particularly in terms of system integration and usage flexibility. As an IBM advertisement declares:

With Linux, companies can get all the diverse systems within and beyond their enterprise working together seamlessly. They can boost collaboration and optimise responsiveness. And that means their business runs more reliably, more productively, and more cost-efficiently.... Linux is not owned by a single company or private enterprise; therefore, it is open and accessible to everyone. And it is constantly being enriched by thousands of programmers all over the world. As a company grows over time, Linux evolves with it. This flexibility is what is making Linux so popular.

## Chapter 6

---

(IBM UK Website, URL <http://www-5.ibm.com/e-business/uk/linux/index.html>)

Although FLOSS offers socio-technical advantages, corporations do not, however, want to be bound by the GPL, which demands, as noted earlier, that they follow the obligations of the social contract. Consequently, adopting or creating alternative OSS licences has become a business strategy for corporations enabling them to become tactically involved in the FLOSS innovation system. As Välimäki's case studies on the experiences of companies Sleepycat Software Inc., MySQL AB, and TrollTech illustrate, several open source companies employ dual licensing to achieve this, both open source and proprietary licenses for one product (Välimäki 2003). In other words, various OSS licences are created in order to meet the corporations' legal, economic and technical preferences. Many other OSS licences are designed in order to lessen the social constraint and uncertainty that the GPL brings. In focusing mainly on the practical part of OSPs, these licences are meant to allow appropriation of open source code. It is an interesting paradox that the GPL, which facilitates FLOSS development, is nevertheless seen as an obstacle in some insiders' eyes. As a programmer at an OSS firm told me,

Linux's main strength is that it is already familiar to Unix users. Its reliability is no doubt an important factor but its biggest weakness is the GPL. Although it helps to prevent forks, it is a barrier to commercial investment.

(TS200602)

A CTO at an OSS SME developing portable operating systems confirmed that their policy towards OSS licences depends on the nature of the market. He argued that the reasoning behind the firm's decisions to release new versions of software under OSS licences had two main purposes:

I think [our product] is quite difficult now from the commercial point of view to compete as it was. It makes no sense that customers must pay quite serious money to get source codes and they by no means may circulate it. They could just put examples on the web site, saying

“here is our stuff, try.” Or ‘Here is the modified version of the driver’. In fact, if they have put it, I wouldn’t worry too much about it. But the licence said that they could only send to other people when they have [a] licence. So it was constrained. I think it’s quite hard to compete, now, generally speaking, if the source code is not made available. So that’s from the commercial aspect.

(CF060803)

He continues,

Another one has to do with you’ve brought as many people as possible if you like. Well, you find that one of the reasons that those become popular even more popular than some other free software such as FreeBSD, is because universities start using it. There’s also this counter culture I suppose which might attract universities or university students to use it in the first place. But I think it’s important in terms of having as many users as possible. You can do that by intermediary for such as experts done within years licences which said, “You could have this [new product] for research, education, and news.” But you can’t develop politics with it. Perhaps that will do as well. But I think what we got [was] a decision to make certain things possible and I don’t have very good reason to regret it. I think [it was] a right thing to do.

(CF060803)

Unlike such voices from within the private sector, other members in the FLOSS social world hold different views on licences and licensing. Oliver, a software developer based in Austria, favoured a non-commercial development model:

I think [a non-commercial version] could protect FLOSS programmers from exploitation. The biggest problem I see with open source software is how to generate revenue to get a project up and going. I don't think it's going to fit under a capitalistic model. I would

love to see floss wares treated like art and culture are treated in middle and northern Europe, where projects are funded based on content, not on sales. But realistically speaking, I doubt that's going to happen any time soon.

(AT090504, oekonux mailing list)

Consequently, Oliver has chosen the Creative Common Licenses (CCLs) to release his works. The CCLs, initiated by Lawrence Lessig to publish his blogs, allow copyright holders to easily inform others that their work is freely available for copying and other uses, under specific conditions, and thus to declare some rights as reserved. Slightly unlike the GPL, the CCLs deliver the idea of freedom of information, however, also prohibit access to the information from some specific agents. Oliver explains why he has adopted CCLs:

What I'd like to do is keep my software open, so that others can share from my ideas as I have from theirs. But, I don't want my apps to be a part of Mac OS X's downloadable tools and I don't want the US military to use my stuff either. The chances of the two happening for me are rather slim, but the point is one of control. I don't want the "free" in free software to mean that anyone can do whatever they like with it. ... Let's say you are working on some high-level motion tracking stuff. This is inherently militaristic, but you have other intentions for it. You'd like to release the code (especially because it was built on other floss software), but you don't want the local police to use it to track citizens on the street, and you don't want Mac OS X using it to track the users who sit down in front of their I-pods...etc....and since you worked for months on it, you probably don't want either one to use it without paying.

(AT090504)

However, Per, another FLOSS developer, challenges the claims made for the efficiency and accountability of CLLs. He says,

[CCL] places a lot of rather problematic restrictions on the content. First, it makes it impossible to distribute as part of a larger whole for which something is paid (for example as part of a Linux distribution). Second, it is legal trouble since the definition of what is 'commercial' can be stretched to fit the view of whoever has the most expensive lawyers. For example, is putting it on a web page with ads 'commercial'?

(PM090504)

Different licences provide, in short, different levels of openness of source code, modifications for further distributions, and appliances linked to proprietary software. Adopting licences is also a means of demonstrating one's position. In Sourceforge.net, the most famous FLOSS software collaboration platform, one can see diverse FLOSS released via a number of licences. For instance, Enterprise Resource Package (ERP) is released under Mozilla Public Licence (MPL) while Gaim is released under GNU General Public Licence (GPL). For some, clinging to a specific copyleft-ed licence or OSPs is a technically strategic act to fulfil practical needs, but for the others, licences and OSPs denote subscription to a particular set of political beliefs.

No matter which licence is favoured, the idea of attracting more users is shared among both firms and the community. Various OSS licences provide a common function of enticing both social and technical interests, and encircling as many users as possible. As Raymond notes about the importance of having users:

Users are wonderful things to have, and not just because they demonstrate that you're serving a need, that you've done something right. Properly cultivated, they can become co-developers. ...[A] lot of users are hackers too. Because source code is available, they can be *effective* hackers. This can be tremendously useful for shortening debugging time. Given a bit of encouragement, your users will diagnose problems, suggest fixes, and help improve the code far more quickly than you could unaided.

(Raymond 2001: 26-27)

Another shared goal among firms and the community is more technically-oriented. The observations of Barrenechea, senior vice president of product development at Computer Associates, stress the advantage of using source code and concepts already available in the open-source world:

There are almost 1 million contributors to open source today. There is an enormous amount of intellectual capability at the grass-roots level--we want to encourage innovation and we want to be able to leverage it.

(C|NET News.com May 5 2004)

Barrenechea reports that his company has spent roughly US\$2 billion on R&D over the past three years. But Microsoft, the largest software maker, will spend about US\$6 billion in year 2004 alone on research. Given the inequity, Barrenechea says,

The only way to compete against that kind of existing technology is via a grass-roots movement. If you have got a million contributors to doing something semi-orchestrated, you can revolt, and Linux is a perfect example of that.

(ibid.)

Hence, adopting OSPs helps offset a company's internal development expenses and supplements its research and development (R&D) efforts.

Although the interdependent working relationships between OSS firms and the FLOSS community can be extremely important, as this quote suggests, the collaboration can be shot through with tensions regarding the commodification of the relationship and thereby the compromising of OSS itself.

Normally, firms receive new staff and ideas from the community, and the

members of the community get financial support from the firms, especially for big projects. This reciprocal demand-supply relationship can be seen in the development of X, but, as a respondent notes, securing this can be difficult where financial payment is involved:

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)

The commodification of OSS practices may well force firms to move away from conducting the OSPs and lock up their source code or cede influence to the proprietary software firms. For example, based on the fact 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 (C|NET News.com 27 April 2004). They claim that this combining of products from the open source and proprietary software camps will help interoperability. Even Red Hat, the leading distributor of Linux, is going hybrid, selling proprietary software (*The Economist* July 25<sup>th</sup> 2002). Richard Stallman, as I described earlier in the thesis, is someone from within the FLOSS social world who holds a sceptical view on commercialisation. He criticises the commercialisation in light of his experience in developing GNU EMACS in 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, <http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt>)

However, though proprietary software firms may be criticised by some OSS advocates for not releasing full source codes, I found that 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 necessarily mean this thing has to be open source, why should it?

(LT060207)

When asked whether this meant he did not fully 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 a software company which would have a single product.” (LT060208).

Brian, who works for a leading OSS firm questions the full-scale open source policy as well.

I don’t think [the community] is a huge deal because even in a commercial world distributing closed source software you get those ideas and suggestions anyway, instead of actual code. That might be

because the companies I worked for before have all been pretty open internally. It's not something magical.

(BO060209)

Brian's view is unlikely to find a positive response in the FLOSS community. Some interviewees mentioned that, after the projects in which they had provided patches or bug-reports were bought by or merged with big companies such as Novell, they stopped contributing to them. Their reason was that they resented providing a free service particularly when firms could have paid someone (create a job opportunity) to do those things. The lack of trust between the community and firms sets up a barrier between the two sectors. How the interdependent relationship between the community and the commercial sector is balanced will be a crucial factor for the further development of FLOSS. Usually FLOSS is tied to notions of a public good:

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 the public meets the private, conflicts, confusions and tensions arise. And that often is the source of the departure from cooperation between firms and the community. Most developers, however, choose to face the conflict 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)

This pragmatic attitude is anchored in the interest in problem-solving, the key hacking practice of detection of software vulnerability and sharing scripts discussed in earlier chapters, and as illustrated in the following quotes from my interviews:

The need to share information stems from the need to resolve problems caused by the multitude of proprietary systems and you need the code-breaking mentality to take on proprietary manufacturers.

(DY011202)

I think that sharing of the stuff is not so important to me, but I think it's good to share in software when you've written something important but er, I do it mostly because other people do it. I have profit from that. So if everybody shares the code it's good for everybody. But other than that I don't see any personal gain of sharing a software.

(PC060209)

Though firm-based innovation appears to provide a standardised and stabilised innovation practice, the community-based innovation seems to afford more dynamics than the firm-based innovation. Its trait of openness and dynamism thus are leveraged to balance the asymmetric relationship between the public and the private sectors. 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 are established. The existence of OSS SMEs seems to be a balanced solution for developers who would like to hold a hybrid identity. These OSS firms are also more likely to share the same ethics valued in the community, such as adopting GPL-like licences (Bonnarccorsi & Ricci, 2003). Alternatively, developing FLOSS

through non-profit organisation works, too (see e.g. McClelland and Silvers 2002). 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<sup>xxxviii</sup>. However, in response to the claim often made that community-based innovation cares less for the needs of specific end-users, the Debian distribution of GNU/Linux has started a variety of customised distribution, named Custom Debian Distribution (CDD). 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 formal Debian Social Contract<sup>xxxix</sup> of having a non-commercial distribution of GNU/Linux operating system.

## 6.7 The Hybrid Identity of FLOSS 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 their strategic and cross-boundary engagement may be different from that in the conventional proprietary software sector, developers actually occupy a dual position in their daily practices, which could be said to give them a hybrid identity in the OSS social world. Their practices at some point have to meet the standard of the business sector for making profits and fulfilling the requirements of their customers, and at the other times their contributions made to the community projects have to meet certain normative, social and technical requirements. Having this hybrid identity in fact gives these developers flexibility to play around. They can acquire resources from both the commercial sector (e.g. money) and the community (e.g. friendship or technical support). They can use the community as a ground to test out their new idea and undertake some experiments, but later on release their work formally onto 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, told me that he committed his spare time to writing scripts for detecting software vulnerability.

The scripts he was writing would be applied to finding out security holes of the systems he develops for clients. One script did not work out fully as he had expected, but after releasing it on the Web, a few bug reports directed him to modify the script to make it better. The script can now detect security holes more efficiently. If he had not written this script to try out his idea, he would not have found out “if certain things are possible or not” (LT060210). The experiment he undertook, through sharing with the community, benefits from the feedback from other members of the community, and later on turns out to be useful for developing commercial software. This experimental manner works well along the (soft) boundaries of the community and private firms, if no other pressures (e.g. financial pressure) apply. Experimenting in close proximity to software products available via OS brings its own rewards: as one of my respondents said:

It’s just experience, something you cannot get from working with Windows or Microsoft NT servers. The most important thing is I think knowledge that comes from personal experience with other products. So you can see what you can do better than others.

(PC060206)

The private sector and the public sector (i.e. the community), however, do represent two different ways of producing FLOSS. OSS firms have to give primacy to the market, rather than the assorted ideologies celebrated in the FLOSS social world. James, a CTO in an OSS firm developing ERP software says,

Fifty percent or a certain amount of people in the free software community have a very strong political point of view. There is a group in the free software community who I know are really anarchists. Another group would be more like in favour of the economic regulation of the market economy, that’s the other group. There is another group like radical and liberal, who wants like pure, absolutely liberal economy. The political aspect exists in the community. But that aspect does not exist of course on the customer side. Not as much as maybe. My customer has absolutely no political bias.

(JP060204)

Consequently, this market-oriented and supposedly political-neutral attitude also reflects on software design and production. As Brian, an employee in a leading OSS company argues,

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 explained his feelings 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 own company is to be economically independent. Having a company provides some barrier to his being exploited by larger ICT firms, and this also prevents his creativity being choked off through such firms' market control.

[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)

The hybrid identity coerces the developers to take a less politically-biased view and to be closer to entrepreneurship. Consequently, when they move across boundaries (i.e. when they move from the community to the business or the other way round), they also interpret practices (e.g. a bug-report) and ethics (hacker ethics) differently. The dynamics in the hybrid innovation system thus complicate the traditional distinction between public and private innovation. In other words, the boundary of what is a public innovation and what is a private innovation is constantly defined and redefined within a FLOSS innovation system.

As Ricci (2003) points out, there are heterogeneous business models in the FLOSS market. She verifies the common discourse that FLOSS is a fundamentally new and attractive paradigm, and argues that, "it opens a large spectrum of organisational and business models" (p. 29). The community and corporations are simply two of the many stakeholders in this heterogeneous social world. Given the assorted episodes described above, we have seen ongoing negotiations between actors in their complex relationships and identities. "The actors negotiate their own identities and interest as well as the existence, nature and volume of

overflows.” (Callon 1998: 264). Their negotiations might bring positive or negative externality to the FLOSS development. As Callon goes on, “the market must be constantly reformed and built up from scratch: it never ceases to emerge or re-emerge in the course of long and stormy negotiations” (ibid.: 266).

## 6.8 The Specificity of FLOSS Innovation

Before concluding this chapter, there is one question emerging out of the discussion so far: is the phenomenon of hybrid innovation (across closed and open/commercial and user-based systems) illustrated above specific to software technologies? If not, has the hybridity of OSS software innovation had any characteristics different from others? Looking at other technologies, parallel stories exist about how the industry gets involved and exploits potentially profitable products invented by individuals or communities as users, for example in the innovation of wheelchairs (Woods and Watson 2003). Wheelchairs, representing mobility and freedom, also emerge from the participatory design and tinkering of users, who may suffer from the disability, are carers, or who are involved in the broad disability movement. However, when big companies started to invest in and buy these community-inspired products, community members and activists were inevitably absorbed into the industrial agenda. This "community-based" wheelchair innovation has not come to an end, however, and is still alive and well despite the increased oligopolisation of the market by two major firms (ibid.). Regardless of the common features found within both wheelchair and software technologies, I would like to argue that FLOSS innovation, given the nature of the *im/materiality* in and the affordance of software at the level of the user, differs from wheelchairs and other community-based innovations. It is less likely that the FLOSS innovation is compromised by the disruptive affect of commodification. Software, characterised as intangible information goods allowing its elements (e.g. source code, programme or software) to be stored and transmitted digitally, carries higher affordance than tangible goods such as wheelchairs. To learn to live with ICT-related products and services that have largely been implemented and deeply embedded in our daily lives, users have not only learned to install and operate software, they also learn to configure and design software for their own needs. Software knowledge thus has gradually been

deciphered and passed on to users, particularly lay-users, in the wider society. As a result of the intangibility and malleability, once source code is released, software development will not be disrupted easily even if appropriation is evident within the system. Additionally, as I have argued above, once FLOSS developers start collaboration with business, this does not necessarily mean that they have cut off their ties to the community. On the contrary, their practice and contacts are still closely related to the community. In fact, the community and OSS corporations as two of the diverse agencies in the FLOSS social world, seem to be quite permeable socio-technical actors. That said, the research does not suggest that the involvement of the private sector is negative. In other words, the FLOSS innovation network encircled with a soft boundary seems thereby to foster wider innovation patterns in the ICT technological system compared with conventional systems, in part because of cross-boundary activities, sharing and recruitment. In this regard, the FLOSS-led innovation system is perhaps emblematic of an emerging knowledge-based society where, suggest some scholars, the value-based assets such as knowledge and digital information sources should be treated as public goods that allow consumers to gain free access to them (Shapiro & Varian 1998; Shy 1998). Much of the digital economy and its information and data often arrive without clear labels as to ownership or to legal right, such that the conventional approach to IP protection and thereby the market is challenged. Accordingly, FLOSS innovation not merely signifies a technological innovation, but more importantly it stands for a wider socio-economic innovation.

Given the dynamic and cross-boundary activities occurring in this hybrid innovation, there is no clear-cut, black-and-white dichotomy between the public and the private, or supplier and user. The cross-boundary mobility and fluidity within the FLOSS social world denote a highly dynamic innovation system. Knowledge, information and innovation resources travel along with the identity shifts that those involve experience and create. The intricacies made up of these individual, organisational as well as institutional configurations complicate the relationship between the public and the private. FLOSS innovation is through the co-fabrication of both sectors, rather than being led by either side.

## 6.9 Conclusion

In personal conversations with technical people, I call myself a hacker. But when I'm talking to journalists, I just say "programmer" or something like that.

Linus Torvalds

(<http://home.tvd.be/cr26864/Hackers.html>)

Although existing FLOSS-related studies have explored the public/private nexus, many of them fail to critically examine the larger contours of the debate around FLOSS. This feature calls for a more detailed analysis, both empirical and theoretical, of the techno-organisational change related to FLOSS development. In light of my fieldwork and interview data, I explain the social, economic, and technological motivations of firms and OSS developers from an ecological and holistic perspective. FLOSS is considered as both a technological as well as a social catalyst that challenges many of the conventions of innovation, such as licensing and cooperation models. In analysing the dialogue between diverse actors and their hybrid identities, I have argued that cooperation and coordination in this hybrid innovation system are done through continuous communication and negotiation. This hybrid model allows actors to acquire resources both from the community and the commercial sector. While the community offers space for experimental projects and informal communications, the private sector stabilises and standardises the development of these community projects by incorporating them, alone or together with proprietary software, and putting them into markets. However, this innovation model may also conceal higher risks and uncertainties due to the conflicts and disagreements between actors. For instance, in their cross-boundary role, morality and ethics often become a source of dispute, which in turn is expressed in the practice of adopting some types of licences rather than others.

In the FLOSS social world, when hybrid cultures collide, most actors chose to take a pragmatic point of view to solve the conflicts and tensions between the

community and corporations, as seen in section 6.6. Instead of treating these discourses simply as “a critical opposition between the gift economy, which emphasizes qualitative relations of reciprocity between humans and which tends towards the personalization of things, and the commodity economy, which objectifies things as property” (Vandenberghe 2002: 8), I suggest that these diverse and sometimes contradictory articulations require an ecological analysis, to situate the techno-organisational change within broader contexts, without reducing it to a simple expression of the FLOSS development.

Considering my hypothesis that the FLOSS social world collides with the hacker social world to some extent, the identity *qua* hacker in the mundane FLOSS social world becomes a notion that allows diverse articulations and expressions. While some developers tended to consider themselves as pure programmers, others regard themselves as hackers, but only in a normative sense (and only being recognised after being reminded by the interviewer; see chapter 4). As Brian says,

I don't like to word 'hacker', because the word has a bad opinion about it. And I am not one of those every time someone says hacker and really mean a cracker, then you say 'no, no, no...'. I don't care. It's just a word describing the hacker culture. And if you take the right meaning of the hacker, and yes, I am a hacker. But I would never use that word to a journalist who wouldn't get it. I would never do that because there is no reason to be religious about that stupid word. It's just a word. ... If I'd like to use a word to describe myself, I would say developer. I like writing software. And I like helping other people to do it, too, which is my motivation for going into management. Then it wouldn't be me writing the software but I would still be developing it by helping people. I am definitely a developer.

(BO0602)

Brian's words confirm the essential features of OSS practice – in terms of developing and sharing software - and reflect the mainstream view prevailing in the FLOSS social world. FLOSS also provides a haven for a range of ICT innovators not baly hackers who can have a say in the innovation system. Such a

system is where *Us* and *Them* can live together and cooperate, where *Others* have the right and space to express their views. Moreover, the heterogeneous networks and serendipity found in the social world enables local and tacit knowledge to be preserved and borrowed as innovation resources. Elements of the FLOSS innovation, no matter how radical or incremental, formal or informal, communal or privileged, global or local, all have a role to perform. This not only ensures that FLOSS development be both socially and technically innovative, but also makes its innovation system highly dynamic and complex. In the next chapter, I will focus my analysis on the importance of local and tacit knowledge in the FLOSS innovation system and their socio-technical impact.

## **Chapter 7 Deciphering Expert Knowledge in Software Engineering: *Glocalising* FLOSS Innovation**

### **7.0 Introduction**

Throughout this thesis I have argued that “hacker”, instead of referring to a fixed identity ascribed to or inscribed in specific actors, is a notion interpreted by *diverse* actors engaging collectively in *specific* computing activities. Through my fieldwork, I suggested that the boundaries of hacker groups are fluid and adjustable, and lack any rigidity. To avoid an essentialist view towards hackers, I analysed the socio-material relationships between actors and actants, humans and technical artefacts in the construction of knowledge. In order to understand the practical behaviour among hackers, I observed a range of mundane, but quite specific computing activities among actors residing in a broad hacker social world, such as programming/coding, sharing and searching information on-/off- line, and reporting bugs and writing patches (challenging each other yet providing mutual support). These activities have significant socio-technical meanings for our understanding of the mutual construction of hacking in society and its place in the software innovation system.

Hacker culture and other enthusiast cultures alike are integral to the lives of technologies and claim technological expertise outside of claims to any professional identity. They complicate traditional notions of knowledge and work. The role these amateurs play in the broader technological landscape is attracting growing academic and practical interest. An increasing academic attention is paid to these enthusiast or ‘amateur’ (Ellis and Waterton 2005), cultures that tend to exist at the fringes of mainstream technological practices, and their members that explicitly define their activities in opposition to traditional understandings of productive work. I want to argue that amateur (technology) cultures are not static but serve as dynamic factors in technological innovation processes. Through my observation of the hacker social world, amateur practices may be institutionalised,

and actors (individuals and organisations) can be transformed and grouped as hybrid organisations with both formal and informal engagement with the wider society. The institutional dimensions of the FLOSS social world and its convergence with the hacker social world illustrates this process.

Looking into hackers' claim-makings and performances, their shared practices and interrelationships show a process of negotiation and collective invention, which reflects the full complexity of mutual engagement in a heterogeneous hacking environment. In this milieu, diverse actors and actants (materials/artefacts) reside and interact through technological means.. The findings described in the earlier fieldwork demonstrate that software technology is socio-technically constructed and all parties affected contribute something to the innovation process. Negotiations and compromise around practices or materials is a common feature, as exemplified by the opposition of RIAA and users of file-swapping software illustrated in chapter 4, or the hybrid innovation model composed of the OSS corporations and the FLOSS community illustrated in chapter 6. To fully capture the complicated relationships and interactions between actors and artefacts in a dynamic enterprise with a plurality of referents, I analysed the ways in which FLOSS innovation emerges through the defining of a problem, engaging actors' intellectual and social interests in tackling the problem, going through a socio-technical innovation process of crafting solutions to the problem, and eventually reaching a spin-off status or redefining the original problem and going through another innovation sequence. As shown in my investigation of FLOSS innovation presented in chapter 5, mutual interactions between actors and artefacts are useful narratives/discourses/texts for the analysis of the FLOSS innovation system. I argue that the FLOSS social world represents an epistemologically multiple arena yet one that is concentrated on the common practice of releasing software source codes shared amongst diverse actors. Holistically speaking, the consumption, deployment, employment and production of FLOSS artefacts/products/cultures/values are embodied in the common open-source practices (OSPs). The OSPs have upheld a global innovation network through exchanging material resources across boundaries (particularly intellectual, social and technical ones). Actors and actants are networked in this heterogeneous social world where communication occurs at a global level particularly via the Internet. Mechanisms and artefacts that emerge to maintain the OSPs (e.g. documentations,

licences or repositories) help to stabilise technical knowledge and make the global transfer of knowledge possible. Altogether these factors enact an open environment for knowledge making and sharing (Lee & Cole 2003; Lanzara & Morner 2003). Hence, converging to or adopting the OSPs creates a sense of socio-technical inclusion for members in the FLOSS social world. In this regard, policies about the deployment of FLOSS are addressed in corporate or governmental strategies and white papers. However, it is worth noting that OSPs are akin to what Fujimura has described elsewhere as “transformative elements in ways that are not predetermined or predictable by cultural location”, being “both producers of society and culture and products of culture and society” (Fujimura 2000: 83).

In this chapter, I focus on the local and tacit knowledge whereby different interests and definitions of problems are articulated to form and to shape the process of software innovation. Whereas FLOSS innovation has been emerging as a global phenomenon, the knowledge network on which it depends is built on a variety of local practices and concerns and tacit intelligences anchored in the widely adopted OSPs. In the deployment of such practices, the FLOSS social world and its knowledge network expand and enrol more actors. In an analogy with (and perhaps exemplification of) today’s “knowledge society” (Knorr-Cetina 1999), the FLOSS social world represents a compelling example of how local knowledge and socio-technical experiences are mobilised in the global knowledge network. Local actors translate their requirements into software projects. Each project denotes an innovation. In each innovation process, actors get mutual help through ongoing discussions and reflections of evaluations (e.g. bug-reports or feature-requests) from peers. Everyday knowledge demonstrated in threads on mailing lists of Linux User Groups (LUGs), through global ICTs, creates valuable digital commons (Lovink 2003) and is shared across indefinite time/space distances. After being incorporated into the global framework, local epistemologies can be learnt by other groups/actors across geo-cultural borders and applied in another context. In other words, the productions of the local and the global knowledge closely connect with each other and shape their formation reciprocally. In this account, local specialists contribute to ontological and structural change in mundane software programming. Individual socio-technical experiences are valued insofar as they can be employed for constructing new skills

and materials. The daily dialogues serve as important narratives for our understandings of software innovation, particularly in the FLOSS context. They also have implications on how the lay-expert divide can be bridged and assimilated.

This view on “glocalisation” is crucial because the boundaries of ‘normality’ of knowledge have been re-created and challenged through textually mediated discourses. It allows us to document innovation stories that are situated in everyday narratives, which differ in cultural, geographical, and socio-technical aspects. In this chapter, one sees ongoing conversations between users and developers, and between the private sector and the public sector. In introducing contextualised local stories, the chapter shows that FLOSS innovation involves not only a cognitive production (how to identify problems and generate ideas/solutions, how to anticipate the unpredictable, from developers’ side), but also a socio-cultural one. It is a hybrid innovation not only because of the collective participations of the public and the private, but also inasmuch as “technological development is often a matter of insiders, but will be exposed to outsiders” as shown in Rip’s writing (Rip 2003: 5).

In this chapter, locality in the FLOSS social world is explored in terms of local performances of accumulating and producing knowledge in response to a global software problem concerning *usability*. The local performance, found in the local amateur groups, Linux User Groups (LUGs), serves as an ideal niche to observe how users translate their interests and perceptions in a form of asking and answering questions, and create collective learning environments on- and off-line. I take LUGs as an example to illustrate how the mutual help of local Linux users forms an alternative knowledge network, which connects with the global knowledge network, and facilitates a wider community-based innovation. York LUG (YLUG) where my fieldwork was done will be drawn on to examine how local practices translate the body of expert knowledge into their everyday life by tinkering with software in forms of downloading, installing and trying out different programmes, and configuring their own computer systems. As one member of the group noted, “learning computer stuff tends to be more of a hands-on thing than a theoretical exercise” (YLUG0203), to be able to manipulate software capably, LUGs

members' local experiences, both social and technical, are privileged as the most valuable. I analyse the languages and information flows (e.g. sharing information, proposing everyday problems and tacit solutions in response to hardware compatibility etc.), in both virtual and real manners, within the local group to see how the body of expert knowledge is translated into the local context, how the members enact their knowledge into practical action, and how the local expertise contributes back 'up' to the global knowledge corpus. In analysing their languages and interactions, one can also understand how expertise is presented and represented in a glocalised context, and subsequently shapes and reshapes the identity of the knowledge holders. One will see that a community-based knowledge network does not necessarily suggest the disappearance of claims to intellectual superiority. The knowledge, though more accessible than conventional science-based knowledge, still requires a certain level of expertise to perceive and utilise it.

To put the arguments in a nutshell, this chapter discusses how diverse cultures shape the deployment and employment of FLOSS, and how local socio-cultural capitals are mobilised to facilitate collective actions in community-based innovation systems. I analyse how local actors manage to maintain a useful innovation method in terms of software engineering and turn that into the common practices (i.e. the OSPs). To raise qualitative questions of meanings and values in the FLOSS social world, I delve into the local stories to analyse different articulations of socio-technical structures and relationships. This analysis contributes to the understanding of the operation, organisation and structures of 'knowledge societies' in terms of epistemic cultures (Knorr-Cetina 1999). I also analyse how social capitals are accumulated and applied to different contexts to develop software to meet diverse users' ends. To accommodate the differences of diverse actors, there has to be a communication channel that allows frequent conversations. These diverse discourses carrying deep meanings and values drawing on people's experiences become multiple sources in the FLOSS innovation. Overall, the local characteristics in the FLOSS social world mirror what Beck (1994) termed "reflexive modernity" in the "knowledge society". The notions of lay expertise or user-led expertise denote a judges-as-the-juries situation that expertise is a medium for opening up rather than closing down questions. One example supporting this is about patches: patches written in the free software

community are under continuous falsification through peer review, just like a wheel driving further innovation, while those written in proprietary environments are much less mobile, less open to change. The other example is about documentation and source codes. Although documentation and released source codes stabilise the computing knowledge, these artefacts in the community-based FLOSS innovation serve as cookbooks that instruct users (in a broad sense) to learn to manipulate scientific/engineering expertise. Users are able to apply the recipes, improvise their own dishes (e.g. configure their own systems), and publish their new recipes. Hence, the asymmetry of the domination of scientific knowledge in the society is challenged and the power of knowledge is redistributed given different community-based solutions emerging from diverse actors' interests and knowledge in the FLOSS innovation system. Mutual-help and community-based socio-technical (-instrumental) support challenge the conventional professional and the industry-led expertise. Expert knowledge is contested by lay knowledge, and the boundary between the two is redrawn.

### **7.1 Localisation within Globalisation**

There is an increasing attention, from both the academia and the industrial sectors, on localisation issues in the fields of engineering and software development alongside the long-standing analysis of globalisation. The drive to develop goods, which match with the different needs and circumstances in differing geo-cultural contexts – or, as some would put it: in different industrial cultures – has been growing. Nowadays Ford's idea of one mass produced car model for the whole world may sound strange, but until a few years ago the concept of a world car was alive and well a subject of serious effort. In software development, there are two dynamics at work today: one is about Universal Design (UD), and the other is about localisation. The former tries to maximise the affordance of the technology by providing open systems built on open standards to meet users' requirements universally and create interoperability, whereas the latter concentrates on specific user needs and develops a customised distribution for a specific group of users. While the idea of UD, proposed within the computer industry since the early 80s, to create a technical community that could overcome barriers of technical incompatibility and so facilitate global collaboration, suggests

a globalised innovation system, the idea of localised distributions however accentuates the importance of producing adapted, rather than standard, products to meet unexpected, contingent challenges. Both streams of developments are well-presented in the FLOSS innovation system, where active collaboration takes place in a virtual environment.

As noted by many scholars researching Internet culture, new forms of electronic communication, as part of the general globalisation process, are giving rise to 'virtual neighbourhoods' or 'communities' (e.g. Wellman, 1999; Rheingold, 2000; Hampton & Wellman, 2003; Jordan *et al.*, 2003; Hargittai 2003). Contemporary life has been driven by the speed-up of technological innovation, which is accelerated by the breaking-down of physical boundaries, both spatial and temporal (Miles, 2000). The virtual geography demonstrates 'the way in which identification and participation are increasingly deterritorialised' (Baumann 1998; Robertson 2001: 466). As a result of globalisation, software production is no longer bounded by the geographical market place. Anchored in intense communication through the Internet, FLOSS innovation is also often pictured as a global phenomenon with large-scale resource mobilisation (Lanzara & Morner 2003). Hence, FLOSS innovation resembles one of the global Internet cultures where social capitals (e.g. the key roles of trust, shared values, and community) are accumulated and disseminated via virtual means through a form of collective learning. As discussed in chapter 5, the materialised OSPs, embodied in mundane practices and artefacts such as databases (e.g. CVS, Freshmeat.net or Sourceforge) or licences, are diffused worldwide, deepened, widened and put into action by computer users. In this regard, FLOSS seemingly has generated what Castell (1996) terms 'the power of knowledge flows' in 'a network society' (p. 469). This feature however does not suggest that the FLOSS phenomenon creates a homogenous culture residing in the 'information society', (cf. Webster 1995). Rather, it is exactly its character as empowering that allows local improvisations. The introduction and operation of new media technologies does not necessarily then direct society to a singular development, but quite paradoxically, through the enlargement of the public sphere and widening of the social network, the situation tends to be balkanised (Bauman 1999).

In addition, we might argue that the hacker culture embodied and represented in the FLOSS social world situates and contextualises its politics in different socio-cultural localities. This suggests that FLOSS might allow actors to empower themselves to mitigate what Giddens (1991) terms the ‘disembedding mechanisms’ typical of globalisation. Unlike commercial software packaged in a standard way for local users (e.g. the MacDonaldisation of Windows-Intel platforms that have been making the software available in a number of languages but all in a single operation environment), the FLOSS innovation system welcomes local articulations, discourses, interpretations, representations and solutions of problems found at *both* global *and* local levels. In so doing, local innovation, in fact, reflects global software problems such as usability, accessibility and an urgent need for open standards supporting interoperability. As we will see below, actors come up with different solutions to conquer the problems about usability, accessibility and digital divide, all of which are under the umbrella of a broader software problem of diverse requirements (user needs) Furthermore, to enable the local solutions to be compatible with the rest of the world, designs have to be based on open standards. Hence, a dialectic of the local and global emerges to signify the oppositional interplay between local involvements and globalising tendencies (Giddens 1991: 242). On the one hand, global practices serve as exogenous forces shaping local strategies, but on the other hand local cultures have seasoned the global imagination with a local flavour in the global-local knowledge framework (Rip 2002, 2003). In other words, local innovation can be transformed into an established expertise on a *glocal* ground. In this regard, it is essential to see how FLOSS innovation is mobilised by local articulations and expressions of problems, particularly those unsettled ones that have become global nuisances. With social relationships heavily embedded in local contexts, local stories/events serve as important discourses to understand the cultural dynamics in FLOSS innovation, as in any other socio-technical fields. These locally-rooted routinised norms and practices, constructed on a global basis by means of Internet technologies, become sanctioned social controls that shape users’ behaviours and produce socio-technical identities under the frame of glocalisation. In addressing the issue of glocalisation, this chapter contributes to the discussion of various problems that attend to simple distinctions between the global and the local, homogenisation and heterogenisation, the universal and the particular (Robertson 1992, 1994). The localities stressed here also suggest the diversity celebrated under the social worlds

framework. While the localities demonstrate “the technological self” (Foucault & Martin 1988) and develop local confidences in deploying and employing FLOSS, the shared OSPs blur the boundaries and link various locals with a globalised FLOSS social world. Hence, the local and the global deeply interlink together in a FLOSS software process.

It is worth noting that the concepts of ‘global’, ‘globalism’ and ‘globalisation’ mentioned here, are not referred to in an economic sense, though it is evident that commercial distributions of open source software have risen to a global/inter/transnational level, as in other techno-economic sectors (and the large Far East market is the hub). Instead, the global-local issues discussed in this chapter are oriented to cultural terms, rather than in primarily economic terms, as it is widely understood in business studies. In presenting the local episodes of the deployment and implementation of FLOSS, the prospective cultural shift is weighed higher than the actual change of the purchase power of digital information or goods. Culture, in many scholars’ views, is “embedded in social structure, in the sense that all social structures convey cultural meanings, rather than being somehow a distinct and separate phenomenon.” (Crane 1994: 4; Sewell 1992). Moreover, not only is the FLOSS innovation process embedded in local episodes, but so are the mundane engagements with the technology that embody users’ approaches to ‘the problem’. In other words, the discursive frameworks engaging with the technology are both embedded and embodied. Nonetheless, such an objection to economic reductionism, does not deny the socio-economic consequences of globalisation. Indeed, global phenomenon represented by topics such as economic neoliberalism, de/regulation, privatisation, marketisation and the crystallisation of what many call a global economy (or global capitalism) is an important topic in its own right (Robertson 2001). However, for my methodological purpose, it is more appropriate to study the globalisation of FLOSS innovation in its socio-cultural context. It is also worth clarifying that a ‘global’ perspective, rather than an ‘international’ one, is adopted in the present text. While ‘international’ suggests geo-cultural relations between nations or nationalities, ‘global’ seems to be a more inclusive concept. As Robertson (1992; 1994) explains, “[‘Global’] does not involve the assumption that ‘international relations or communications cover all that is to be known about the world as a whole” (Robertson 1994: 34).

## 7.2 A Community-based Innovation: An Alternative Software Process

Most software methods are highly idealised and grossly inadequate for analysing and modelling the articulation of real world arrangements. They fail to take informal interactions (heterogeneity and contingency) in the innovation process into account. This gap is partly derived from the limited exchange between developers and users, which has been a prolonged problem in software innovation systems. To integrate both perspectives from users and developers, conventional software processes require some modifications to enrol diverse actors and encourage dialogues. As discussed, a “problem-solving” mindset is strongly grounded in software engineering. When the boundary of the innovation system is volatile, this attitude can be mobilised to inspire innovation under a heterogeneous environment, as discussed in chapter 5. However, this prospect is not well illustrated in most classic software processes (some call it ‘systems development’), such as the well-known ‘Waterfall Method’ (see chapter 2) (Royce 1970; Sommerville 2001). Most of the time, users’ requirements are pictured solely based on the stereotypes in developers’ minds without hearing the voices of the actual users. In this regard, a problem is articulated in a monolithic way and few solutions can be incorporated into the development. In other words, software development is treated as a linear formula, rather than a fluid process. The prospective innovation energy that might be generated through the co-construction of knowledge and skills drawing on the social and material resources available for diverse actors, is not available to the innovation process. As Bannon (1998) argues, ‘These approaches tend to leave out far too much of what is involved in the nature of work, specifically the inherent capabilities of the human actors and more particularly the communities of practice around such work.’ (p. 53). If software development could incorporate the experiences of the ‘workers’ themselves, or here, “end-users”, through active participation in the design process, it is more likely that a usable as well as useful information system is produced that integrates both developers’ and users’ perspectives.

York Linux User Group (YLUG) is just such a social network that epitomises a community-based innovation system. Innovation within YLUG is developed

through collective learning, and everyday practices and tacit skills are shared, learned and evolved. Within such a community of practice(s), learning and action are situated, and that work is accomplished via artefacts, in conjunction with others. The ‘model’ (used in a loose way here) exemplifies an alternative software process that bridges the gap between designers and users. This concept seems to be similar to the ‘activity-oriented’ approach (Sachs, 1995) or the ‘participative design’ approach (Kyng 1995) showing how to ground design in a deep understanding of the practical contingencies of work practice. However, the model, based on a cultural phenomenon, instead of positioning designers and users in two different knowledge frames, brings them into a common platform where they encounter and interact directly. This platform, mainly in a digital form, serves to build up a community where designers and users share the same interests and act collectively. Engaging in a community of practice(s) has become a unique cultural experience for both users and designers. The peripheral participation in the FLOSS community makes the software innovation process transparent and permits diverse innovators to engage in it. Creativity and culture thus emerge in this community, both at the global and the local levels.

### **7.2.1 Demography of a hybrid Organisation: the Virtual and the Real**

YLUG is an informal organisation without restriction of membership. It is geographically based around the University of York, but is open to everyone, and the membership includes a good proportion of non-university people. There are about 90<sup>xl</sup> group members from different sectors (government, industry, academic institutions) living in York and the surrounding areas. Members gather because they share the same interests (mostly technical) in Linux. This group of enthusiasts engages with a wide range of levels of expertise and experience. Though the group meeting is held once every two weeks during the term time, the operation of the group depends heavily on the Internet ( <http://www.york.lug.org.uk/intro.shtml> ). Hence, there is a strong tendency that the group acts as a virtual organisation. The YLUG mailing list works actively and topics are vigorous: hardware information, product reviews, security news, trick or tips exchanges and so on. Related research by Silvonen (2002) on the Finnish LUGs (FLUGs), shows there are mainly 4 types of discussions going on FLUG’s mailing list with 1800 subscribers and 241

individual threads consisting of 3248 topics. And the proportions occupying the list are: asking for help (62.4%), business about FLUG (17.6%), Linux focused (12%), and other issues (8%) (Silvonen 2002). While it may be arguable whether Silvonen's categories of these 4 types of discussions is over-simplified or not (in that the category of 'other issues' may contain dozens of other types of discussions and be omitted to make a point), his observations provide a useful characterisation of LUGs around the world. The YLUG, heavily technique-oriented, also appears to be a virtual organisation for exchanging information and sharing knowledge, where 'networking is essential' and can be referred to as a 'networked organisation' (Murray and Willmott 1997). As Silvonen notes, 'The community is the result of continuing joint activity on the Net, of conflict and negotiations by which the identity and borders are created' (op. cit.). LUGs, as virtual organisations, exemplify social and organisational, and not simply technical, innovation. As Dutton (1999a) characterises this, "the essential dimensions of this organisational form include (a) networking through the use of ICTs, (b) restructuring into a decentralised network of companies, and (c) building a team culture.' (Dutton 1999: 475). Having said that, 'the boundaries of the organisation and the units that compose them are becoming increasingly permeable.' (Dutton 1999a: 474).

Though the members mainly interact through global ICTs, it does not mean that the conventional face-to-face communications have been substituted completely. Instead, face-to-face communications still play an important role for the group to build cohesion and trust (ibid.: 475). LUGs are constructed virtually and physically, as also found in Silvonen's case study. He says, 'FLUG hosts on-line and real-life activists as well as radical and moderate hackers. The meaning of 'Linux' is being constructed in continuous debates between these groups.' (Silvonen 2002). Face-to-face meetings, as other Linux-related conferences mentioned previously, entail profound social functions that virtual communication cannot achieve. As Silvonen remarks, 'The Linux community is a complicated network of different communities. Although the Internet is the essential tool for these, they are also anchored in local social activities.' (ibid.). For example, in terms of knowledge transmission, oral conversations sometimes explain better than written discourses, either in a digital form or a paper form (perhaps why we still need lectures in schools even though there are a plenty of textbooks and WWW

resources). One can see this from the short message reproduced below. After Zoe gave a talk on LaTeX at the regular YLUG meeting, her slides were uploaded to the Net. A few hours later, a message responded:

OK. Zoe's talk makes for really good reading. Wish I could have been there. Couple Questions...

(YLUG011303)

This message implies that though a reader could understand the talk clearly with the material given on the Internet, and the network did give him a chance interacting with the speaker, he still wished to be at the live presentation. Among the YLUG members, the knowledge about LaTeX is built up through diverse forms: going to the talk, reading the material on the Internet, emailing and discussing with people on the mailing list. Engaging in binary communications (i.e. on- and off- lines) allows information to flow more fluidly within and across the group.

### 7.2.2 Hacking Experience and Tacit Knowledge

As mentioned earlier, the most common topic on LUG mailing lists is asking for help. Though FLOSS helps open the black box of software technology, not everybody can understand what is inside the black box. Getting onto [www.kernel.org](http://www.kernel.org) and downloading the source code of the Linux kernel programme, will hardly make sense for someone with little knowledge of the basic programming language. Some suggest then that the FLOSS is too expert-centred to be user-friendly, and only competent programmers can use them. Moreover, even if customised or packaged FLOSS has been developed progressively for the past years, there are fewer manuals or guidelines available compared to those for MS Windows and its peripheral systems. FLOSS documentations, mainly based on contributions from users in the community, are not complete either. At this point, LUGs serve as a channel for users to gain help. It is worth noting that help via the community is not a one-way process, but a form of mutual help. A person giving the answer to a posted question may not only be a helper but also a learner. Unlike mainstream expert-led problem-solving approaches, the practices of LUGs often

illustrate hands-on trial-and-error tinkering methods. As recent community studies show, a barrier-free, accessible community environment is intended to offer the members opportunities to engage in interaction with others who share the same interests (Evers 2001). Through the Internet, they can present themselves, express thoughts and ideas and engage in social interaction on an equal basis with wider Internet users. Furthermore, the shared practices of trying these methods ground the local innovation and enhance the identity of the members and the boundary of the community. These opportunities of *glocal* communication also appear to offer ways to reduce feelings of isolation and contribute to the exploration of self-identity and increasing self-confidence in social environments.

John, an undergraduate at the Physics department, sees himself as a fast-learned novice Linux user. One day, he asked a question about SuSE networking config:

```
I have with me a fresh SuSE 9.0 installation which is failing to boot;
freezing when the init script tries to bring up eth0 (a pcmcia rtl8139).
This in itself I could try to debug, if I could just persuade suse not to
try to bring up the interface on boot. I have edited /
etc/sysconfig/network/ifcfg-eth0 to replace TARTMODE='onboot'
with STARTMODE='manual', but it's still trying to activate it at boot
time. Any suggestions as to where to prod are appreciated.
```

(YLUG141203)

In this message, John explained the problem by identifying which Linux distribution it is (i.e. SuSE) and what happened (i.e. unbootable after its installation, and freezing when the init script tries to bring up eth0 (a pcmcia rtl8139)). He also described the approach that he used to solve the problem (i.e. debug). However, the script he changed did not seem to work well and thus the problem remained. John has tried to give detailed information about the background of the problem encountered. In so doing, John embodied the question and made it comprehensible within a virtual space, so that other members did not have to sit in front of John's computer screen.

Matt, a computer science undergraduate, responded:

If it's to do with PCMCIA I would check your irq settings and such like for your PCMCIA setup. I have a HP nx9005 and the use of any PCMCIA card made it hang without alterations to the setup.

(YLUG151203)

Matt's suggestion was based on his experience. Though John's problem was not directly related to his previous experience, he made a conjecture about it. However, in response to Matt's message, John clarified that,

The card actually comes up properly during the initial installation boot, as it does with knoppix or suse live. Don't think the problem is this low level, but thanks.

(YLUG151203)

By clarifying the condition, John has negotiated with Matt over the proposed solution for the problem. In his reply, John, nevertheless, narrowed down the scale of the problem by distancing the problem from possible difficulties with peripheral hardware (i.e. card). When the problem came up, John might or might not have known its exact cause. He knew there might be more than one solution for the problem, but he was not sure what was the cause of the problem. By removing one of its possible causes, it was more likely to identify the main difficulty and find a solution for it. That is to say, Matt's suggestion, though it did not provide a feasible solution, acted as a catalyst that filtered out the noise in the problem-solving process.

Ewan, a doctoral student at the chemistry department and an experienced Linux user, asked a further question about the status of John's system when it was about to bring up eth0 of the pcmcia rtl8139: "Which is plugged in at this point or not? And what happens if you try it the other way?" (YLUG151203). By asking this question, Ewan also tried to improve the identification of the problem. Additionally, Ewan suggested that

How about either starting it up in single user mode (or booting with

knoppix) and simply removing the links to whichever init script brings the interface up. Then, once you've got a working system figure out what you need to do bring the interface up manually, then fix the start up scripts to do the same. I'd guess this is an ordering thing, maybe bringing the network up before starting the pcmcia stuff. If knoppix can get it right then it might make a useful reference point.

(YLUG151203)

Ewan guessed that this problem was to do with the order of execution of booting scripts, so he suggested to leave the SuSE installation behind, and use a more simple distribution like Knoppix to get a working system established first. He reckoned that in so doing John might be able to figure out which order was the correct one. It was a very hands-on solution. This progressive problem-solving approach (like peeling an onion layer by layer and eventually getting to the core of the onion) requires recursive trial-and-error steps to identify the problem and solve it.

In response to Ewan's first question, John said, "Tried both ways, no dice." (YLUG151203). He seemed disappointed that Ewan's question did not help to improve the situation. Then, Ewan's suggestion, though apparently not particularly novel either, somehow affirmed John's view:

>How about either starting it up in single user mode (or booting with knoppix) and simply removing  
> the links to whichever init script brings the interface up?

I suppose this is my next step. I'd just hoped to stick with the kind of configuration that YaST can control, rather than getting my hands too dirty. Oh well.

(YLUG151203)

In this narrative, one can see that John was not an innocent user asking questions or simply seeking help. He had given serious thought about how to solve

the problem on his own. When a similar approach was proposed by his peer, it confirmed his knowledge at certain level. To get endorsement from one's peers for an imaginative solution is important because it reduces the uncertainty of the solution and directs the actor to continue the experiment. John's motivation to resolve the problem was thereby enhanced. As a result, apart from asking for suggestions from the local user group, he also reached out to make association with the wider Internet community at a global level:

>If knoppix can get it right then it might make a useful reference point.

Yeah, knoppix does get it right. Interestingly, so does the SuSE netinst! I installed the system over the LAN, but at one of its many stages it hung bringing eth0 up. At this point, I still had access to another VT, and could see what was going on: modprobe.old was running at 100% cpu, indefinitely. Killed this and the installer was happy again, still with a nice working network. Hoped it was just a glitch, but clearly not. I've found one reference to this issue through googling, which is an unanswered cry for help with some other PCMCIA NIC (this person hadn't pushed their way through the installation as I did).

On with the distasteful hacking of the boot scripts....

(YLUG151203)

His Google search made him realise that he was not alone with this problem. Though there was no immediate solution available on the Internet, the lingering problem inspired him to hack. After a few hours of hacking, John announced his success in solving the problem:

Resolved:

Ewan's guess about it being an ordering issue was spot on: I put an "exit" line straight after the hashbang in /etc/rc.d/network, and the

system booted. Interestingly, the next thing after the step that had been failing was a nice little "beep beep" and "Starting kernel PCMCIA". From here I could ifup eth0 with no problems. I've now rearranged the order to something a bit more sane.

Oh, and I'm going to make a private embarrassment public: it wasn't suse 9.0. It was suse 8.2, installed with a 9.0 netinst disc. I installed off the wrong LAN share, and at no point in the installation did I notice (I swear it never mentioned a version number). Oh well, 8.2 works well too, and it's not for a power user. The worrying part is that I only just noticed this last minute :-)

(YLUG151203)

Ewan's suggestion had shed some light on the solution. Though John had been approaching the problem in a similar way, he did not know that it was an ordering problem. Ewan, apparently a more experienced user, was able to reason through the problem and thus to deduce a workable solution. The whole problem-solving process relied heavily on hands-on practices. The dialogues between the members also showed that solving a software problem is not only a cognitive process involving much immaterial labour (i.e. intelligence), but also a highly materialised activity engaging with a variety of computing components, both hardware (i.e. tangible artefacts) and software (i.e. intangible artefacts). To solve a software problem, one has to be competent in handling both hardware and software in the procedure. Moreover, describing the precise nature of the problem appears to be a crucial to securing wider help from the group. In representing the problem, one has to describe the status of both the tangible hardware and the intangible software. To make the question more comprehensible to other members, one also has to be familiar with some technical jargon or terms. Because John was able to speak fluent computing language, the problem-solving process was more smooth perhaps than one encountered by a more innocent user. However, his representation of the problem also opened up another problem, inasmuch as his audience interpreted the difficulties in their own ways. Thus, in the narrative above, Matt and Ewan both make conjectures about John's problem. Matt, without asking any further questions, regarded John's problem as a matter of hardware. This suggestion while

not providing an immediate solution, still served to filter out the noise in the problem-solving process and to categorise the problem. But Ewan, acting more deliberately, clarified the situation by asking a further question and giving a practical suggestion. In order to get a full picture of the problem, Ewan discussed the issue with John in the form of enquiry and then made a less prescriptive conjecture about the situation by saying “how about this?”. For John, he had learnt how to select useful information and secured a solution to his problem through accessing both a local (e.g. help from the enthusiastic YLUG members) and a global (e.g. he did the Google search linking to the global Internet) level. Each response to his original message served as a catalyst that helped him (re-) classify the problem. Throughout the narrative above, Matt, Ewan and John had made different interpretations about the problem. In their exchange, Matt and Ewan were not merely information givers, and neither was John merely a question raiser. They interacted and learned mutually. By giving information to John, Matt and Ewan learned whether their conjectures were feasible. And by sharing his problem-solving experience, John became an information provider giving his experience away as tacit knowledge. Hence, the knowledge was socio-technically constructed and embedded in the negotiation process between Matt, Ewan and John.. While asking a relevant question or conjecture helped to clarify the situation, it also showed how the meaning of the problem was produced by and negotiated between the actors. Questions and problems appear not only to address current matters, but might perhaps be preserved as repository of innovation.

In his last message in this thread, John shared his solutions and a “private embarrassment”, as he put it, with the list. The information was valuable in that his experience, as a type of tacit knowledge, was valuable for prospective novice users. Even his private embarrassment was appreciated as a reminder to other users that paying attention to the ‘version number’ during the installation is important. Other users around the world, such as the one he came across to when doing the Google search on the problem, could also benefit and learn from his experience by conducting another Google search. Thus, John’s role in the software innovation system is rather passive. Without positioning himself on the frontline of the FLOSS innovation system, John contributed to the FLOSS innovation in an implicit way of making the discourse containing tacit knowledge available on-line. Whoever, whenever, and wherever might make a Google search on this topic

would benefit from John's local narrative. Community-building thus takes place at both the local and the global levels. And it is also clear that the Google search engine has a social-technical function facilitating FLOSS innovation. Moreover, in terms of global innovation, John's experience could be borrowed by software distributors to improve their products (e.g. marking a version number more clearly, or changing the default booting order). That is to say, users' experiences provide insights about technological designs and contribute in productive ways to the broader knowledge system. This contextualised problem-solving story thus has a glocalised affect on the FLOSS innovation.

### 7.2.3 An Open Hierarchy of Expertise

As observed, most members in the YLUG are experienced computer users. Given the fact that everyone is good at something but unevenly so, the same members might act 'sometimes as insiders, sometimes as outsiders' (Star 1995: 7). As shown in the message below, a skilful user positions her/himself as something of an innocent abroad, an outsider, in seeking help from the group:

I have a large pile of (scanned) photos from a holiday. A lot of these are panoramas which I want to join together to make one big picture. Anyone recommend any software for doing this easily? If your answer is the GIMP then please point me at some instructions since I had a frustrating time with it!

(YLUG032503)

The author of this message is actually a systems security advisor in the Computing Services department. Apparently he is very knowledgeable in computing. But his message here presents him as an 'outsider' or novice in regard to processing graphical data. A couple of replies were received. The first went:

One way to do it in the gimp:

Open one of your pictures, note the size of the image.

Open a new blank image and create it of size equal to 3x (or however many photos you want to join) the width. Allow 20% or so extra on the height as well.

Then open the rest of your original images. Copy them and paste them into the new image. Make sure you've got the ones you paste first lined up before you add others.

When you're either save, or adjust canvas size, or just drag a box around them and copy. Then create new and it'll automatically be the right size for you to paste your copied image into.

Save however you like.

(YLUG042503)

This message was from Ewan, introduced earlier, who is a doctoral student at the chemistry department who might ostensibly be regarded as having less knowledge than staff in Computer Services. However, he acted like an insider here not least because he was an experienced Linux user but also because he was skilful at this specific software application. Another message follows:

Sounds mostly right to me. You may find lining things up easier if you load each image into a different layer, then you can set each layer to be partially transparent. I hope you've got a fast machine.

(YLUG052503)

This author of this message is an employee at a tele-computation company. In his message, he acted as a more experienced insider than Ewan by saying 'Sounds mostly right to me.' And he added some tips and tricks based on his experience to endorse the thread. These personal experiences, after being shared, have been objectified and distributed among the members as a source of problem-based innovation that can be used by –others compiling family holiday photos.

Tips and tricks gained through solving everyday problems are particularly appreciated in the FLOSS community. This type of knowledge, generated in an informal network, is useful and creative for mundane software practice. Unlike the mainstream innovation systems with a strong inclination to codified and expert-led knowledge, the community-based innovation system suggests an open hierarchy of knowledge production. Regardless of the social roles of the actors (e.g. lecturer or students), whoever owns knowledge that works more feasibly appears to enjoy a higher status in the user group. The meaning of “expert” thus is contestable and need not accrete to one person over time. The status and the identity of the members that comprise the local pool of expertise are flexible and fluid. Without a rigid hierarchy, the members can interact more freely without being constrained by hierarchical social rules. This social heterogeneity is itself reproduced through the more open knowledge exchange process, and in turn feeds back on this process to renew and extend the scope of problem solutions.

In addition, collective practices such as exchanging information and sharing experience also help to integrate members in the YLUG. In a group enjoying a high information flow, members appear to feel a stronger sense of belonging and greater communication is encouraged. In other words, there is a positive synergy between the velocity of information flow and the degree of group consolidation. Unlike a formal knowledge network that can be constraining, this informal network sustains the innovative dynamics of the community and encourages risk-taking and experimentation in the group (Shearman 1997). As Sharp (1997) argues elsewhere, “technology is often complex, multi-dimensional, expensive to implement and specific to a particular firm ... a large part of it is tacit knowledge (i.e. passed on by word of mouth and not written down) and derives from trial, error and learning, rather than from the systematic application of science.” (Sharp 1997: 93).

Whilst YLUG acts as a carrier of information and knowledge, it is the trust between the members that sustains the synergistic ambience and upholds the whole community. The dynamics of the construction of the trust can be observed through reading the digital rhetoric, which embodies the members’ tacit knowledge and everyday experiences. Given the narrative above, I want to suggest that there are

two dimensions of trust, what I call “trust of persona” and “trust of information”. These two types of trust are endorsed respectively by the off- and on- line activities. Face-to-face communications can foster the trust and cohesion in the group. By meeting each other in real life, the members deepen their understandings of each other’s personal backgrounds: where are they from and what they do. At this point, face-to-face meetings help the members build trust and cooperation in the group in a way of empathising their shared interests and collective practices. I see this as ‘the trust of persona’ insomuch as the trust is built on the understandings of members’ roles and characters. The second type, “the trust of information”, facilitates collaboration to carry on smoothly in virtual space. In the category of ‘the trust of information’, the level of trust and the intensity of cooperation vary according to the quality of information. If the quality of information given is more accountable, the person who provides the information is regarded as more knowledgeable. Since her/his information is trustworthy, s/he gets credit for sourcing the information and consequently s/he enjoys a higher socio-technical status in the group. The trust of information thus symbolises an indicator of expertise in the YLUG. The trust of persona and the trust of information appear to work this way among YLUG members: whilst the higher trust of information defines a more accountable source of information, meeting in person helps improve personal relationships and determines whether the source is sociable enough for further collaboration. The reciprocation and expectation between the members shape the trust and cooperation, and vice versa.

In light of the YLUG episodes, FLOSS, as a technology socially produced, “needs to be understood within a multi-level, multi-actor historical context, with the state as one among a range of actors constitutive of the international/transnational innovation system” (Talalay *et al.* 1997: 8).

### **7.2.4 Local Expertise and Global Innovation**

As discussed briefly above, the YLUG activities, though situated within the local group, have a global impact by way of the heavy usage of the global ICTs and ongoing contact with the worldwide FLOSS social world. The following thread on the YLUG list gives a more thorough illustration of this glocal

phenomenon. Through the narrative below, one will see that the local innovation materialises in strong associations with intangible artefacts (i.e. code and programme). The actors who are more likely to provide valid information normally have a superior competence at speaking fluent computing languages to communicate with and mediate between human and non-human actors. Additionally, the innovation process is embedded in their everyday hands-on computing practices. This attitude concretises a strong risk-taking approach to experiments and improvisations.

This thread was initiated by Gary, a skilful enthusiast in the group, when asking a question dealing with the problem of shell portability. As Gary is quite knowledgeable, the followers of this thread had to share about the same level of expertise as his to be able to provide mutual help. Thus, the barrier of this thread is somewhat stronger. Many of the discussants, including Gary, develop software for global use. In engaging the local actors in the discussion of his problem at hand, Gary's action linked the local with the global, the exchanges situated locally with many 'crouching tigers and hidden dragons' at the YLUG, ultimately contributed to the global software innovation system. Gary's original post went:

Heya, Guys

Question: Is the following shell script portable across all flavors of bourne? Notably including SCO, IRIX, Solaris, Linux[duh], HPUX, AIX, BSD ?

```
#!/bin/sh
for f in "$@"
do
    echo \"$f\"
done
```

Obviously, It's a question about "\$@". I have access to all these systems, but there's a difference between it empirically functioning in the environment here at work and there being an official line that it WILL work everywhere.

Gary (-;

(YLUG070404)

Gary had been aware of the core of the problem which was “\$@” when posting this message, and his script actually worked with his local platforms as well. But portability, dealing with multiple platforms around the world, has to be checked carefully whether it is actually compatible with other systems. As Gary said, “there's a difference between it empirically functioning in the environment here at work and there being an official line that it WILL work everywhere.” Gary chose to ask his local fellows about this. Will, another member on the list, who did not usually respond, answered Gary’s message. Here it went:

AFAICT in IRIX, Solaris, HPUX and AIX /bin/sh is the Bourne shell and has a common ancestry to the code originally written by Steve Bourne. According in 3.4 of Mr Bourne's shell tutorial (for example: [steve-parker.org/sh/bourne.html](http://steve-parker.org/sh/bourne.html)) the \$@ is as being part of the Bourne shell. So my guess is, unless some drastic "refactoring" has happened, you are safe.

Under Linux /bin/sh is \*normally\* bash. However, I have been bitten by somebody who preferred tcsh set this as a symlink. I have also seen a static bash shell with all possible options switched off which can break stuff. However, excluding the criminally insane, again this will be fine.

Dunno about the forty seven varieties of BSD. IIRC the Bourne shell was removed for copyright reasons from the original BSD code dump and am too lazy to dig out my O'Reilly BSD CD to check this.

In summary, you will be fine. But YMMV,

(YLUG070404)

Will’s message confirmed Gary’s piece by providing sensible explanations from the textbook/tutorial (formal knowledge) and his experience (tacit knowledge).

Though he was not able to provide more information about the BSD systems (because BSD has got too many divisions and Will was not familiar with the systems), he did provide a source to check it out which is the O'Reilly BSD CD.

Gary was happy with Will's reply. In response to Will's first answer drawing from the Bourne's tutorial, he said "Makes sense. That's the answer I was hoping for". Gary confirmed that Will had made the right interpretation of and the right answer to his question. As to Will's self-experience about being cracked, Gary said "Frankly, I consider a box with /bin/sh as tcsh to be broken-by-design, and I couldn't give a monkeys if the box works or not." And in the end, Gary understood that Will was actually pointing him to a reference, for which he replied "Cool. Thanks." to Will. In this dialogue, the communication between Gary and Will was very smooth because their knowledge was at about the same level and each was able to understand or make further conjectures about what the problem was.

David, a self-employed IT contractor providing open source software support, followed up the thread.

Can't see who can give any "official" line on whether something will run the same across all those unix variants, but Portable Shell Programming by Bruce Binn is a pretty good reference (bit pricey though). Linux has generally caused me the most problems as I've seen /bin/sh linked to; /bin/bash, /bin/pdksh, /bin/ash, /bin/tcsh, /bin/zsh.

On HP-UX 11i and Tru64 /bin/sh is no longer the Bourne Shell but the POSIX shell.

Can't see how you'd come a cropper using something like \$@. Maybe looking at some "configure" scripts might help you find out some of the things you want to test for and then generate a script based on that?

(YLUG070404)

David, as Will, pointed Gary towards several sources of information. However, he had a different interpretation of the problem querying, I “can’t see who...” and “can’t see why...”.

Roger, another experienced Linux user and a Debian developer, followed up:

It's used in Autoconf-generated configure scripts, so if that's your idea of portability, I guess it's OK.

If you want to write really, really, portable stuff, it might be worth using M4SH. Probably found in /usr/share/autoconf/m4sugar/m4sh.m4[f] At the expense of preprocessing your scripts with m4, it might be quite useful--it's used to provide shell portability for Autoconf, but there's no reason why you couldn't use it. (Aside: m4sugar.m4[f] might also be useful--but this stuff makes badly-written Perl look comprehensible!)

Part of m4sh:

```
# Be Bourne compatible
if test -n "${ZSH_VERSION+set}" && (emulate sh) >/dev/null 2>&1;
then
    emulate sh
    NULLCMD=:
    # Zsh 3.x and 4.x performs word splitting on ${1+"$@"}, which
    # is contrary to our usage.  Disable this feature.
    alias -g ' ${1+"$@"}'=' "$@" '
elif test -n "${BASH_VERSION+set}" && (set -o posix) >/dev/null
2>&1; then
    set -o posix
fi
DUALCASE=1; export DUALCASE # for MKS sh
```

(YLUG070404)

In his first sentence, Roger argues over the definition of “portability”. Though “portability” is a professional jargon found in computing science, its meaning is not always clear or unequivocal. In light of the Gary’s script, Roger conjectured

that Gary was locating portability within “Autoconf-generated configure scripts”. However, Roger has his own notion of “really, really, portable stuff”, materialised in a specific ‘sh’ language, ‘m4sh’. From this exchange it seems clear that the whole software process is based on programmers’ differing definitions of the situation. The technology produced is determined by the actors’ readings of the problem, based on their diverse experiences and backgrounds.

Zoe, a research fellow at the computer science department and the only female computer exponent in the YLUG, provided her suggestions. As an experienced programmer who had given many lectures at YLUG meetings on topics such as LaTeX and ‘shell scripting’ Zoe pointed out some of the aspects related to non-portability, based on her own experience:

On a side note, there is a corner case where different older versions of sh shipped with different vendors will have a different behaviour - when there are no arguments, some will run that loop once with an empty string, and some will just not run the loop.

The script looks as though it's supposed to show each argument on a separate line, preceded by and followed by a double-quote character?

I'd write it as follows:

```
#!/bin/sh
for f
do
    echo \"${f}\"
done
```

As far as I know, that will have the same behaviour in all versions of sh. If you really need to know whether there are any arguments at all, in general, the following ought to work:

```
[ x = x"${1+x}" ] && exit
```

Run that with no arguments, and then with an argument of "", to see

the difference (perhaps with "#!/bin/sh -x" at the top of the script for trace output).

(YLUG080404)

In the discourse above, Zoe is making her own conjecture about Gary's software problem. This is based on a case she had heard of, which reflected the problem of different versions of sh languages. By analysing the structure and components of Gary's script, Zoe deduced that Gary might like to show each "argument" (as a computer jargon here) on a separate line, preceded by and followed by a double-quote character. However, she was not sure whether that was Gary's intention. Thus, her sentence was ended with a question mark. Zoe's judgement was also derived from Gary's scripts. Gary's script materialised his problem/question and provided a concrete platform for further discussion and debate. Accordingly, Gary's script acted as a boundary object in this thread, shaped and also being shaped by the members' thoughts. On the one hand, the discussants all anchored their claims in light of Gary's script; on the other hand, they all came up with another idea that either modified the original script or directed to references that would challenge the original script. The human and non-human interactions consequently made the negotiation process centring on the problem more dynamic. Because the specific category of programming language (i.e. shell programming) was put in the centre of the thread, one needed to own this specific type of knowledge to be able to penetrate the boundary to join the discussion. In a way, the discussants were selected (by the given context) from the members to participate in this peculiar problem-solving process. The relationships between the members also change subtly through the way they reformulate their ideas about the problems. Their practices, both individual and collective, define, mark and identify their status and identity.

From the thread above, one could see how an innovation is derived from a problem proposed by an actor, engaging other actors who shared the same interest and knowledge to provide solutions. In the problem-solving process, the members negotiated their definitions of the problem and its components, exchanged their opinion by proposing their own conjectures. The materials (i.e. the code and the scripts) in which the negotiation was embedded and which also embodied different

ideas are examined from different angles to represent the problem and solutions. The dialogues between Gary and the regional YLUG members contributed to his programming for a portable software product that had a global bearing. As such, coding is an iterative process whereby the programmer has to simulate the context within which a programme is to be used. Because the presumed context contains various uncertainties, engaging with more actors in the innovation process is a key part of the process.. Additionally, the global ICTs enabled Gary to have such a glocalised discussion with his local fellows. Knowledge was generated in a process and shared among these discussants and other users on the Internet. The knowledge process is endless. Though a temporary solution would be assimilated and applied to Gary's product, the local discourse scattered on the Internet would become informative re/sources of innovation to be picked up and used elsewhere in new ways.

Such glocalised activity is a feature defining of the FLOSS innovation system that fosters a high degree of flexibility for improvisation. Problems with hardware compatibility and bug-reports/feature-requests are common glocalised phenomena in the FLOSS development. Much of the bugfixing, integration work and special features added to the local requirements goes back to the core Linux development (e.g. the new release of Ubuntu Linux<sup>xli</sup>, a hybrid Linux distribution combining resources from both the private sector and the community, contributes back to several free software projects by testing and debugging Debian's unstable packages, enhancing GNOME<sup>xliii</sup>'s feature, and simplifying the installation of X window system in diverse hardware configurations by improving Xfree86<sup>xliiii</sup>). Additionally, hardware compatibility is one of the main challenges that software engineering faces. As one software engineering textbook notes, software engineering faces the challenge to operate as distributed systems across networks that include different types of computer with different kinds of support systems (Sommerville 2001, p. 13). Since software is a set of instructions detailing the operations to be performed by the computer, inevitably, the design of the hardware shapes the development of the software (Peláez 1988, p. 2). Assembling hardware and software together into a single a machine is highly problematic with unpredictable effects, at least at present. Linux, and other operating systems, all struggle with open systems.

However, a feature of Linux and other FLOSS operating systems is that users can find some 'kernel' patch from somewhere to fix a problem. Such a patch is the result of many trial-and-error activities similar to the original author's. For example, some frustrated innovators, mostly local users, found some RAM was not compatible with the Linux system, and reported this back to a database, which is accessible to all users, or emailed the project maintainers. Someone with a technical competence saw these bug-reports and wrote a patch dealing with the problem of these bad RAM. The innovation, the 'badram patch', has not, therefore, come from one designer's idea, but started with various users' trial-and-error innovative activities. Without users noticing the 'segmentation fault with rpm install', without local users randomly guessing the probable problem of 'RAM', without reporting that a certain brand of ram is badram, the patch and the badram page would not exist. The patch, therefore, is a socio-technical artefact. The dialogues above just show how effective the local innovation can be, and how the local knowledge base is formed. This type of innovation based on a problem-solving process through social learning can be linked back to the discussion of the TODO-list and bug-reporting in chapter 5. As Ben, a Debian developer comments,

In some ways, a good interactive bug report is in a BTS [(bug tracking system)] where you have a developer participating with a bug reporter, and (in ideal situations) even with other users/developers who are subscribed to the package in question. You can get solutions -- or partial solutions -- posted in the bug report and it can, in some situations, look a little like the tool your describing.

I think a really good example is bugs that are difficult to reproduce on one's own system. Porting issues and locale issues can fall into this category. When you keep that process moving, in this case, through a rich "conversation" that sort of merges the identify the problem with fixing it, it's harder to fall into the "bug logged, bug ignored" mode.

(MH130204)

His words point out the weight of the local problems in solving software problems at a global level. A joke based on a true story shared by Ben illustrates

this glocalised link vividly:

If you'd like funny example I can give you one from the last several releases of Template Toolkit. There was some strange locale issue where in Australia, a date manipulation test passed and gave the correct day of week (Monday) but in other places it was Tuesday. It took more than 2 full releases and a great long on-list debug session for a while to get this squashed but because the bug was difficult for the author to reproduce, he was dependent on this interaction with folks who drove on the other side of the road to try to figure it out.

There was one point when the primary author said something like: "To get test 25 to work, do the following things." A follow-up said added another suggestion: "move in with Andy, it works in his house."

(MH130204)

### **7.2.5 Social Facilitation: Mobilising the Community**

As many scholars have noticed, "social networks are indeed essential elements in the generation of technological innovation, and they are the backbone of the social organisation of any innovative locality" (Castells and Hall 1994: 234). YLUG, based at the university, is able to build its social network with the local staffs and students, and extend it by enrolling regional enthusiasts. Though most activities take place during the term time, the virtual communication allows the members to continue their discussions during the vacation, overcoming time and space restrictions.

As noted above, shared interests and the shared knowledge are the main factors engaging the members in the group and sustaining their interaction. To leverage such local creativity and turn it into innovation power, it is essential to understand the shared interests among the members. In light of my observation, technical issues of Linux or generic FLOSS are of more interest than social-political ones to the YLUG members. The content generated on the YLUG mailing list, therefore, is

centred on technical issues. Additionally, to get as many involved as possible, members tend to focus on generic issues. This is illustrated below in my recording of a technical thread about the PDP-typed machines.

On a note on extremely powerful computers - I was just reading about the PDP-10 not a while back.

It's really quite fun - you can get an emulator and boot TOPS-20 or ITS. I still can't understand how people worked on those things. My father has told me stories of soviet PDP's (I think they were made in the USSR but with original DEC processors) and terminal to computer problems. Unbelievably it stood until 1995, the PDP-10, in FIAN in Moscow ( Physics institute ) when it was probably slower than a 486DX ( the VAXens still stand there today ). Anyway the emulators are worth a go if you have some hacking time - there is also a free TOPS-20 shells service that is more or less dead but for guest logins. Telnet into twenex.org, but the website itself does not work so to read about it you have to get Googles cahces a go :-)

(YLUG 110504)

Arthur, the systems security advisor at the computing service department followed up,

> Unbelievably it stood until 1995

More worrying there are still live PDP-11's running minor things like Nuclear power plants....

(YLUG120504)

Arthur's opinion prompted a lively exchange about terrorist attacks. Simon soon made a connection between Arthur's message about nuclear power plants and a potential security issue. And he also expressed his concern over the Windows system.

I'm sure that BNFL will thank you for revealing info like that ;-)

What worries me more is that things like this are invariably Windows-driven - witness the havoc caused in the US and South Korea by the Slammer worm last year ...

(YLUG120504)

Windows is often attacked in this way by Linux users, describing Microsoft as the common enemy that unites the community to fight against the monopoly empire. Whilst this claim might be partial, it shows a popular perception about Windows systems among Linux users, as seen in the later messages in this thread. Chrispin, a staff at the electronic department, joked about Simon's message.

> I'm sure that BNFL will thank you for revealing info like that ;-)

Terrorist! Get him! :-)

I remember visiting some nuclear power plant in Suffolk (Bradwell?) First one in the country. They had no Y2K problems there. In fact, they had no computers controlling the place at all. I like that.

> What worries me more is that things like this are invariably Windows-driven - witness  
> the havoc caused in the US and South Korea by the Slammer worm last year ...

Just you wait until Linux \*really\* takes off on the desktop. Then it'll be possible to make fair security comparisons. Not to knock it but it will be interesting.

(YLUG120504)

Arthur then answered Chrispin:

> Terrorist! Get him! :-)

Not a great secret. e.g.

[http://groups.google.co.uk/groups?q=pdp11+nuclear+reactors&hl=en  
&lr=&selm=4rtvd8%24ara%40flood.weeg.uiowa.edu&rnum=1](http://groups.google.co.uk/groups?q=pdp11+nuclear+reactors&hl=en&lr=&selm=4rtvd8%24ara%40flood.weeg.uiowa.edu&rnum=1)

(from 1996 but talks about PDP8s which obviously predate the PDP11)

(YLUG120504)

Paul also shared his recollections about nuclear power plants:

Sizewell A and B are the nuclear power stations in Suffolk and I live within 20 miles of site when I'm not at Uni. Bradwell is quite close by in Essex, but the reactors were shut down last year.

On the subject of Windows powered systems, has any one noticed that Nationwide cash machines have shown that they run Windows (probably Windows CE) by recently having a problem (especially the one at Goodricke on campus) where a run time error occurs and a Windows dialog box appears saying so with a button saying OK, which you can never press!

(YLUG120504)

Paul's message is contextualised in his daily experience of the cash machine, both about the nuclear power plants and the Windows systems. The YLUG social networking is then highly embedded in members' daily lives and lay knowledge becomes focus for discussion. When the topic came to Windows, more members joined the debate. Phillip confirmed the cash-machine's use of Microsoft Windows system:

I believe they're running Windows NT 4.0 actually - judging by the

widgets of the error message anyway.

(YLUG120504)

The daily banking experience certainly reminded the members of their bad experience with Windows. Hence, more messages came:

Lloyds/TSB used to run OS/2(!) - I remember walking down Piccadilly and seeing the OS/2 equivalent of the BSOD on one of the cash machines on the corner by the traffic lights; tried to take a photo of it, but the batteries in my camera had gone flat :( I've seen the genuine BSOD on the NatWest machines at the railway station in the past \*and\* on the GNER monitors.

Good to know your money is in safe hands, eh? :-) I don't think that BSOD spotting will take off as a hobby any time soon, though.

(YLUG120504)

Phillip came up with more examples about his bad experience with Windows:

The software in train ticket machines in Austria had a bug that caused the system (Win98[!!] I think) to BSOD when you just "wiped" across the touchscreen with your hand. Luckily they fixed it soon after the machines had been introduced. It wasn't very nice to be at the whim of teenagers with a tendency to vandalism as to whether you got a ticket or not...

(YLUG120504)

Another member, Russell, also showed his concern over the cash machines around the UK using Microsoft Windows systems:

The thing that amuses me is seeing cash points that certain banks use (Nationwide and Halifax and anyone else with the pointless, slow pretty picture ones rather than the classic easy to read text ones) that

have crashed because of Windows networking errors.

What is that saying about small things pleasing people?

(YLUG130504)

Acting silently most of the time I experimented to see whether it is possible to direct members to discuss more social issues about PDP and hacker ethics in a broader sense. I posted a message in light of the case study I have done about EMACS where the PDP-10 machine played an important role in the early times during which Richard Stallman was writing EMACS. However, no one has responded to this message. In other words, I failed to engage the members' interest and thus my message was not followed up. Similarly, other threads on the list that did not get any response either failed to engage the members' interests or were not able to find matching expertise to continue the discussion. For example, Vladimir did not return to the discussion, and that might be because the discussion had been developed into the direction he did not expect it might be (off his interest). However, the tendency of the discussion was of interest to most of the members and thus it created a centre of attention on the list. To mobilize the intellectual and social resources in the YLUG, one has to locate the topics and content of the discussion that are compatible with interests of the members.

Software technologies are deeply anchored in our daily lives. We all have many memories and experiences of using computers, either successfully or frustratingly. Some of these memories and experiences are individual, and some of these are collective. As Giddens (1991) says, “virtually all human experience is mediated – through socialization and in particular the acquisition of language. Language and memory are intrinsically connected, both on the level of individual recall and that of the institutionalization of collective experience.” (Giddens 1991: 23). It is this hybrid level of individual recall and that of the institutionalisation of collective experience that connect the local and the global in an Internet-based network society. Through this process of institutionalisation, these user experiences are illuminated and can be useful sources for software innovation. Sometimes these daily experiences are taken for granted and thus ignored in software process. However, paying attention to the thread above, one can find the usual concerns

over security and computer breakdown. The YLUG forms a common ground for the members to share these experiences and values. Their commitment to the technical keeps up the momentum and distributes a collective practice of the OSPs. The members' collective identity is under constant socio-technical (re-) construction through both virtual and real environments. These can be mobilised to stimulate ideas about designs, both bad and good. In this way, YLUG acts as a social facilitator that fosters further innovation in providing peer support. While others reliant on proprietary systems are constrained from innovation and locked into the technology and business models of entrenched interests, the YLUG members use technology to foster innovation in making the most of the localised community of practice. It is worth noting that all discussions were followed up within 24 hours. This also shows how resourceful a network can be where a common problem is shared.

### **7.2.6 Glocalised innovation: Local Peer Culture and Global Knowledge System**

This section provides a critical introduction to the role and cultural significance of technological innovation in redefining the boundaries of experts and lay, tracing this process through the digital and verbal rhetoric in the group. The innovation shown in the case study above is based on a specific setting of the YLUG. In other contexts, we might find other dynamics, although I would argue that the point about localised knowledge being turned into global innovation still holds. The case study also implied that software development does not follow a simple linear path, as shown in the conventional 'waterfall' model. Instead, it can 'be shaped and reshaped erratically and somewhat haphazardly as a host of different actors and events came into play' (Lea *et al.* 1999: 319). Among various organisational forms existing in the software industry, the LUGs, however, illustrate how informal organisations serve to mobilise the innovation, socially and technically. It gives us an insight that how a local innovation crafted by a community of practices can be taken up to shape a wider range of innovation, and be shaped as well. While many computer users are used to one single operating system, many members in the FLOSS community create their own culture with a shared interest in Linux, one of the FLOSS systems. But rather than this necessarily leading to convergence as to one reading or sense of Linux, the members of the FLOSS community sustain the

socio-technical openness of the system by programming/coding, reporting bugs, contributing patches, distributing packaged solutions, publishing articles, or simply using a set of software, and in so doing, make sense of ‘the system’. In so doing, they also demonstrate their self-identity through virtual performance, embodied in mundane programming. In this sense, FLOSS is a negotiable idea, not a stable set of artefacts: ‘the’ FLOSS.

As observed, the YLUG activities took place in both the real and the virtual environment. Such interactions, whether face-to-face one or virtual, involve humans and non-humans (artefacts) within a complex setting or multimodal environments. In recent years, research on the interaction between humans, artefacts and situated cognition in collaborative activities has been the subject of a growing interest in different fields and approaches. Whilst most researchers have focused on discourses within collaborative activities in formal settings such as hospitals, classrooms or laboratories, my observation of the YLUG case concentrates on episodes happened in informal settings. I argue that these mutual interactions taken place in an informal group will shape the software innovation *glocally*.

As discussed in chapter 5, the community-based innovation is a “self-unfolding” model that enrolls actors sharing the same interest in the innovation network to solve problems (Shah 2003; Merten, S. Oekonux Project<sup>xliv</sup>). This innovation based on ‘a community of practice’ or ‘a community of interest’ is parallel to what Fleck (1988) terms ‘a horizontal division of labour’ in his ‘innofusion’ framework. ‘This horizontal dimension is directly concerned with the distribution of knowledge among the communities carrying that knowledge.’ (Fleck, 1988: 12). Fleck remarks:

There would not be a perfect one-to-one correspondence, of course, between the commonalities of knowledge and the communities of peers. And so a measure of overlap results, sufficient to enable the transmission of certain elements of knowledge through extensive networks of communication and contact, despite the necessarily highly localized distribution of specialist competence. Such a

localized distribution follows of necessity because of the tacit and contingent (ie. specific to the particular context) elements in the vertical structure of knowledge, which create very formidable barriers to entry and transmission, so that once a practitioner has made an investment in a particular area, he or she becomes de facto committed.

(Fleck 1988: 13)

Fleck also notices that there is a wide range of institutions involved in technological activity (ibid.). Yet, Fleck believes that only formal relationships between institutions appear in the technological community. He says: ‘ [I]nstead of being integrated by an extensive network of informal communication and contacts, relationships between the groups are mediated primarily by formal contracts, by market exchanges, and by the physical transfer of the technology itself.’ (ibid.) This seems to me to underestimate the role of informal networks within the ICT technological innovation process. The informal communications between actors, whether conflicting, negotiating, coordinating or competing, act to shape the innovation as well. Though Fleck is aware of the existence of the informal contacts and communications, these factors to him are minor. In particular, Fleck assumes that only a hierarchical organisation, what he calls ‘a vertical division of labour’, can efficiently serve a rapid technological innovation. While Fleck calls for extensive user participation in design to ascertain whether the new possibilities that come to light are what users want, he appears to advocate the incorporation of users into a more formal structure.

In contrast to Fleck, the example above shows the importance of informal relationships in a local user group. In integrating and deploying artefacts in their activities, the YLUG members negotiate the meanings of concepts and artefacts in the problem-solving process. This also means that the artefacts themselves shape patterns of interaction and this feedback between actors and software scripts is a socially reflexive process. moreover, the informality of the YLUG network is in part dependent upon the flat, hierarchical character of the network and the display of expertise therein. I suggest that this generates a more rich and open approach to innovation that can become globally valuable as non-YLUG users access and engage with the issue in hand. In other words, the local feature of peer-production

is linked to the globalised FLOSS community. This is suggested in Benkler's observation that,

thousands of individuals make individual contributions to a body of knowledge, set up internal systems of quality control, and produce the core of our information and knowledge environment. ... Individuals produce on a non proprietary basis, and contribute their product to a knowledge 'commons' that no one is understood as 'owning,' and that anyone can, indeed is required by professional norms to, take and extend.

(Benkler 2001: 7)

As the case study has shown, activities among the local user groups can have a global impact through the WWW and the local expertise working on software for global use (e.g. portable software). Apart from that, the members can also participate in FLOSS development by reporting bugs, requesting features or providing patches. The openness of the FLOSS enables users to take part in the innovation process. The feature of social networking thus becomes an important asset for innovation. Rather than looking for formalised relations, I have suggested how informal social relations and hacking practice enables the co-construction of the cultural/material artefact of YLUG and, at a wider level, FLOSS. As Brian argues, 'The social construction of artefacts is at the same time the materialisation of a practice that enables particular kinds of agents to intervene productively in the world of things.' (Brian 1994: 193). Likewise, Suchman argues 'systems development is not the creation of discrete, intrinsically meaningful objects, but the cultural production of new forms of material practice' (Suchman 2000: 9). Additionally, the practices are also to 'inscribe agency in artefacts, give objective status to agents, and enact the boundaries of the domains in which they are empowered to act, in which the available technologies of production operate' (ibid). This argument highlights the relationship between human and non-human actants, and is parallel to Suchman's argument that 'objects are subjectified' and 'subjects objectified' (Suchman 2001: 6). FLOSS thus is not only a technological innovation, but more importantly, it is a social one too.

### **7.3 Empowering the Minority: Usability, Accessibility, and the Digital Divide**

Hitherto, I have analysed the social and technical interactions and relationships between the actors in the YLUG by observing their problem-solving exchange and practice. These interactions, taken place in an informal setting, symbolise a dynamic and idiomatic culture of knowledge-sharing and innovativeness which itself could be highly formative in the production of more codified ICT products and services. The YLUG and the other user groups alike provide a space for FLOSS users to exchange their experiences, transmit tacit knowledge and provide mutual help. Their activities not only contribute to the global software innovation system in a sense of helping out some programmers working on the software projects and products, but also in another sense of tackling a global problem of usability. FLOSS is said to empower users in making source code available. However, without adequate expertise, after opening up the invisible black box of software technologies, users are still left in a weak situation and strongly dependent on experts. At the YLUG, the members learn to read the code, manipulate the code and work with the code. Here, “the code” does not only indicate software source code, but is more generically referred to the code of knowledge. Whilst open source code is decoded, so is the formal expertise deciphered. Consequently, software technologies can be challenged, adapted, and ameliorated to satisfy diverse user needs, which is a pervasive and prolonged problem in software innovation. That said, usability can be improved with the advent of FLOSS. Nevertheless, this problem will not be solved in the short term. Whilst open source code can be exploited and employed by expert programmers for individual needs, for users, unless they have the expertise as well, the knowledge does not transfer to them straight away after source code is released. That said, the empowerment of technology is not a natural process. It requires skills and experiences dealing with and translating detailed information into various languages which facilitate wider accessibility to the core of the knowledge system.

In this context, the YLUG and other Linux user groups might serve as a catalyst (and template) to facilitate the knowledge-deciphering and -transferring process. Consequently, software knowledge could be transferred and relocated in users’

daily life, and users can be empowered concretely. The global problem of usability, hence, might be tackled through grassroots social networking.

### **7.3.1 The Blind's Self-Prescription – the Innovation Story of Brltty**

Another global feature of FLOSS is linked with the problem of software accessibility, which is another type of usability problem in a broad sense. To distinguish from the concept of usability referred to generic problems, accessibility addresses problems concerning disabled users. A positive example of this type of innovation is Brltty. Brltty is a free software project as well as a product developed for vision-impaired users. The innovation story of Brltty exemplifies the accessibility problem and shows how a community-based innovation, emerging from a marginalised problem and engaging the socio-technical experiences of users both with and without impaired vision, can be produced. Marginalized user needs are not well addressed in mainstream software designs. In the Brltty context, users, particularly those handicapped ones often ignored, are empowered in the innovation process. Brltty, as a product for the visually impaired, symbolises a material expression of the current concern with accessibility within software design to satisfy a group of people on the margins of visual space. Brltty and other innovation alike not merely benefit currently disabled people, but also eliminate the fear of getting old in an aging society by providing tools to overcome potential vision-impaired problems.

Additionally, the story also brings up the issue of inequality in the course of the globalisation of software production. When software engineering is placed in a globalised frame, the codified and institutionalised knowledge for developing software reproduces unequal structures of power, wealth, income and social status. The unequal distribution of these elements is articulated in local contexts, and connected with problems of usability, accessibility and the digital divide. To quote Bauman, “[R]ather than homogenizing the human condition, the technological annulment of temporal/spatial distances tends to polarize it.” (Bauman 1998: 18). Under such a situation, "particular social groups find themselves persistently denied the same degree of access to social rewards and resources as other groups" (Cohen & Kennedy 2000: 99). The Brltty story, and the implementation of FLOSS

in developing/undeveloped countries (see below) is a response to the forms of inequality found in software innovation systems. In an analogy to Steve Brown's (1996) studies, in the Brlty case, people with disabilities have forged a group identity to translate their interests and needs into an artefact for self-help. Based on a common history of oppression and a common bond of resilience, their behaviours denote a kind of resistance to powerful mainstream engineering, and also reflect the fact that the relative significance of different kinds of inequality can change.

However, as with the specificities of the YLUG case, the particular life experiences embodied in the design of Brlty can actually contribute to a wider software design and innovation. For instance, the function of Brlty is to transform the content of a virtual console into Braille code and transfer it to Braille displays. The 'Daemon interface' program for Braille displays provides a new basis for interface software development. Furthermore, the body of knowledge in the field of speech-synthesis software ('talking software') is expanding in response to the ongoing development of Brlty in that speech-synthesis technologies are important for the blind user groups as sightless people are not able to see anything on the screen.

### **7.3.2 Bridging the Digital Divide: Implementing Software in Local Contexts**

Beyond the specific issue of developing software that meets disadvantaged groups needs, it is worth noting how FLOSS is addressing wider patterns of inequality through the work of NGOs in developing/undeveloped countries. Here, the notion of the 'digital divide' becomes a matter of concern at a global level.

The digital divide refers to the gap between those who have access to the latest information technologies and those who do not and the economic and social handicap that the latter experience (Compaine, 2001). In software design, structured inequalities operate along the main axes of gender, race/ethnicity and class. Each of these in turn generates its own structure of unequal practices giving rise to institutionalised sexism, racism or class

divisions/conflict. “Gender, race and class also crosscut each other in various complex ways, sometimes reinforcing and at other times weakening the impact of existing inequalities.” (Cohen & Kennedy, 2000: 100). For instance, Webster’s research (1996) employing feminist approaches to study computer system designs addresses the issue of a male-dominated system design field, which continuously excludes female users’ needs, requirements, interests and values in the innovation process. She argues that “human factors may be bolted on to existing methods of systems design, local and contingent knowledge of work and information handling processes held by users in an amorphous sense may now even be incorporated into the systems design process, but this does not create an awareness of the way in which skills and knowledge are defined in gender-divided terms.” (p. 150). In a similar way, I argue that users’ experiences in developing or undeveloped countries are often ignored in conventional software designs manufactured in developed countries. Although localisation of an information infrastructure is a key issue emerging in current system development, profit-oriented products and services, such as Microsoft’s local language program (LLP), do not really meet local needs. Rather, this type of multi-languages software package, a software suite fabricated universally for countries around the world, signify the phenomenon that I term the “MacDonaldisation of Windows-Intel platforms”, which in fact alienates users and their local contexts<sup>xiv</sup>.

In contrast, in putting FLOSS into practice, it is anticipated that not only software knowledge but also the pattern of social networking can be grounded in local interests to help overcome the digital divide. In recent years, an increasing number of governments have endeavoured to either convert the public administration infrastructure from Windows to Linux or to adopt FLOSS for similar tasks (e.g. Munich in Germany or Zaragoza in Spain) (c.f. “Linux in Spain” on LWN.net; C|Net News.com August 29, 2001; Open Source Observatory 2003).

There have been a number of practical considerations that need to be taken into account when implementing FLOSS in countries or organisations devoid of intellectual or financial resources: economically, how to keep software costs down; in educational terms how to improve human resources; and politically, how to ensure proprietary software does not take too large a market share while at the

same time promoting digital autonomy. These concerns have brought local governmental and non-governmental organisations (NGOs) as well as FLOSS activists together to tackle these issues.. Because knowledge transfer is as crucial as infrastructure implementation, hands-on training made available to local users is essential in the execution. Projects such as the E-Riders<sup>xlvi</sup> and Low Income Networking and Communications<sup>xlvii</sup>, or events such as the Summer Source Camp<sup>xlviii</sup> and Africa Source<sup>xlix</sup>, all illustrate the transfer of knowledge and technology across cultural borders. These examples also show how the implementation of FLOSS shape the lives and identities of local users as well as software developers around the globe. Additionally, there is considerable evidence that we are seeing much greater implementation of Linux-based infrastructure in the local educational, NGOs and governmental organisations in developing countries or regions (e.g. *Washington Post*, November 3, 2002,). The advents of embedded technologies such as the “Simputer”, a Linux handheld applied in India, are believed to enable affordable, sustainable village development in places without phones and power, giving more and more people a voice in their future (Cherlin, 2002). Wireless technologies also help to bring the Internet to developing countries or regions to facilitate networking at both local and global levels. Krag, a Danish expert in wireless technologies whom I met at the 2003 summer source camp in Croatia, describes wireless technologies as low-cost and decentralised. Here is a quote from his talk at the O'reilly 2004 emerging technology conference<sup>1</sup> about the advantage of wireless technologies:

Billions of people in the world have never been online. The Internet as a technology is an elitist tool, reserved for the few and unreachable by the many. This is a problem not likely to be solved by the commercial interests of existing telecommunications companies and existing ideas about expensive, centralized infrastructure. But low-cost, decentralized wireless technologies could have an important role to play, in this respect. Their low price point and decentralized nature, and the openness of the standards, mean that these technologies are incredibly adaptable to new situations and new uses.

(Krag, 2004)

Krag and his colleagues have been working in the undeveloped/developing world, building up and promoting wireless technologies (mainly 802.11b standard, also known as WiFi) for local use. They bring the Internet and intranet connectivity to those parts of the world not included in the plans of the commercial telecommunications companies. They teach and give hands-on training to the local about how to use ICTs, and at the same time build wireless networks in the countries they visit. In so doing, they hope to “not only raise awareness and heighten skillsets, but also gain the experience necessary to build a central repository of documentation and tools, targeted specifically at the developing world.” (ibid.). Krag’s words show that working with local users in poorer countries has both immediate value in training and accessibility, as well as longer term benefit in helping to establish a more global software platform that has local utility. That said, working in undeveloped/developing areas in fact is not a one-way process but one that involves mutual-help and mutual-learning. Sometimes, extra functions are introduced to the original products or facilities to meet local users’ needs or habits.

Krag argues that the innovation that can be seen to derive from addressing the digital divide will require the deployment of hacker ethics. Here, hacking not merely denotes a purely technical act of reverse engineering of proprietary programmes, but the socio-cultural expression of hacker ethics that offers a political challenge to dominant ICT systems. In Foucault’s term, hacking challenges the legitimatised knowledge and signifies one’s resistance to subjection (Foucault 1978b; Wood 1985; Allen 1993). In considering hacking as a metaphor of subjectivity, tracing hacking practices and discourses thus contributes to our understandings of modern societies where knowledge and power are connected to each other (Foucault 1973, 1975, 1978a, 1982; Weeks 1989; Wood 1985; Allen 1993).

### **7.3.3 Democratising Software Innovation Process—Whose Democracy? Utopia or Dystopia?**

The concept of the digital divide has always been a contested issue. Social scientists argue about whether the digital divide is a social problem that can be

confronted individually or a more structural one rooted in wider problems of inequalities. As Atanu Dey, an Indian economist, argues,

It is not the digital divide that is preventing the poor from benefiting from ICT. It is the fact that they are poor that is preventing them from benefiting from ICT. Not just benefitting from the use of ICT, the poor also are not benefiting from the advances in medical technology, in cosmetic surgery, in plasma tv technology, ad nauseum. It is not the digital divide, stupid, it is an income divide, it is a wealth divide, it is an opportunity divide.

(Atanu Dey <http://deeshaa.org/>)

If the digital divide is a social problem that can be dealt with separately, when solutions are crafted to solve this problem, nevertheless, they are mostly identified from the point of view of the developed countries. Even though NGOs and FLOSS activists are keen to articulate local needs, it appears that in practice the framing of problems still favours a developed country perspective. They believe that information and knowledge should have primacy, and team building and social networking the most effective means through which to foster social capital and innovation itself.

However, in some countries access to and freedom of information is not the priority; but rather the need to improve local peoples' wages and purchasing power in the local/global market. Implementation involves both local and global politics. As Nahrada, a devoted FLOSS advocate based in Vienna, says reflexively in a message posted to both [minciu\\_sodas\\_en@yahoogroups.com](mailto:minciu_sodas_en@yahoogroups.com) and [globalvillages@yahoogroups.com](mailto:globalvillages@yahoogroups.com) mailing lists, "People that 'we help' are not just 'local people in a local community', they are future global players, too, sharing a common task - At least this is what we should expect them to be and what we should empower them to be." (dated 19 September 04). Whereas this perspective of glocalisation is worth of bringing into practice, the main problem then is how to address the digital divide in parallel to social inequalities. In the local implementation process of FLOSS, one might have to pay more attention to the question of "who provides what for whom?".

In the case of wireless technologies, it is true that wireless technologies are relatively cheap and have the technical advantages of simplicity, openness, decentralization and autonomy. However, there is a tendency among FLOSS workers to see in this some form of technocratic fix and thereby to ‘often take the technological configuration of the new media as a ‘given’ or prefigured system that needs to become more widely diffused to citizens’ (Mansell 2002: 408; Mansell & Steinmueller 2000; MacKenzie 1996). However, as Mansell (2002) points out, arguments such as Lessig’s (2000, 2001) research ‘does not examine the rhetorical forms that help to sustain the configurations of the new media that are favoured by an influential minority of technology developers and producers’ (ibid). It is also likely that the implementation of FLOSS could weaken the opposition between the rich and the poor because the poor cannot perceive the digital divide as an urgent problem.

To deconstruct the myth of empowerment and participatory democracy (digital independence), I argue that a rethink of the digital divide is necessary. Compared with other ICT based on proprietary software, a FLOSS-based implementation does provide a better approach to bridging the digital divide from many aspects, particularly in terms of mapping and incorporating local users’ requirements and lowering their ICT purchase costs (Wheeler 2004). However, the degree of empowerment may be counteracted if local users’ interests are not fully included in the implementation agenda. So-called “Digital independence” and “Participatory democracy” are often perhaps cases of a FLOSS hype, as they are in most ICT advocacy contexts. Having said that, it is not my intention to deny the advantages of FLOSS and its socio-technical impacts. But I would like to point out some blind points remaining in its worldwide implementation and deployment. There has been a tendency in the development of ICT to facilitate human activities through the provision of a more ubiquitous and efficient infrastructure and interface to meet users’ needs in various areas. Whilst this vision of a global digital culture sounds promising, the emphasis on the presumed benefits of ICTs also leads to all manner of delusions and falsehoods. When a newly invented ICT product is praised for its influential potential to empower consumers, we need to ask whether a product is usable and useful to users, and how this varies in different cultural and social

contexts. This focus on the infrastructure-building has misled both the public and the private sectors to believe in technocratic fixes and a set of dominant and monolithic values, such as ‘efficiency’, ‘modernity’ and ‘improvement’ (in the context of the ‘state-of-the-art’ – we see this in the UK at present in the attempt to provide a new ‘information spine’ for the NHS and a linked telehealth care system that will handle chronic illness remotely (see eg Wanless, 2002). Users are expected to embrace these developments. As a result we see an emerging socio-technical hierarchy is simply reproduced to categorise users: to be connected/wired or to be alienated from the digital world; to be mobile or to be outmoded; to be electronic or to be antediluvian. Alternative meanings of ‘innovation’ and the limitations of ICT itself receive little or no attention in this dominant socio-technical context.

Whilst the priorities and deployment of FLOSS opens up the issue of the digital divide from being simply about level of access to information facilities (hardware) to information contents (software), this extended scope remains the preserve of a small group of digital elite, as Mansell (2002) argues. In this regard, I share the opinion of Thomas & Wyatt (2000) that access is not the only important issue for understanding inequality. Instead, it is the assumption that the diffusion of Internet connectivity is seen as necessarily expanding and beneficial that should be questioned. Sociologically, it is more essential to understand the meaning of ‘access’ to the Internet and how this is interpreted and appreciated differently by different social groups across geo-cultural borders. When FLOSS engages with the local level, it may well be the case that "[d]isadvantaged groups are also exposed to forms of discrimination as well as to ideologies, culturally dominant values and learning roles that induce them to accept their ‘proper’ social place" (Cohen & Kennedy 2000: 99). Many cases of local or lay operations, such as the recent hands-on workshop on FLOSS targeting women technicians from Southern Africa (<http://www.apc.org/english/news/index.shtml?x=25034>), exemplify the multiple forms of resistance that are mobilised against different bodies of organised power when socio-technical change takes place (ie. a movement of subjective

reconversion) (Foucault 1978; Wood 1985; Allen 1993). The strategies and tactics adopted and deployed in local settings represent socio-political ideologies situated in a specific space and time. But these resistances to taken-for-granted realities and dominant forms of social organisation together with desire for new technologies and socio-technical statuses simply point up the paradoxes found within the *glocalised* FLOSS social world. Whilst it is believed that the development of FLOSS epitomises a participatory democracy, a bottom-up innovation grounded in our daily lives, one may find that the world will not become such a utopia, especially though the incipient technological determinism that some FLOSS activists promise, liberating people to form a global egalitarian community. Nor on the other hand will it be a dystopia producing armies of disembodied, lonely individuals to form a digital communism resembling Huxley's *Brave New World*. Yet, the plurality of FLOSS innovation and its expression in real and virtual environments does I reckon have some broad social and political value. That is why I reckon the implementation of FLOSS is a "syntopia". (Katz & Rice 2002). It provides users more choices. It aims to bring a cultural shift into being, rather than directly confronting economic inequalities. Compared to the mainstream, internet based systems that tend to reduce diversity, FLOSS opens and retains cultural variation on which choices might be built. Sustaining such diversity is important at a more general level, that is beyond the specific domain of FLOSS. As Hand (2003) argues,

The Net appears as the key individualized technology of self transformation. In other words, our received images of the Net appear to have become framed within the grammar of 'third way' sociopolitics: the grammar of empowerment, interactivity, participation and choice. However, these apparent continuities should not lead us to think in terms of a homogenization of contemporary culture. We continually stress the contingency of these arrangements. As we have seen, the dominant discourses of the Net have at their core a will to reduce, compress, and simplify the multiple, fractal and conflictual dynamics of the Net in everyday life. We must avoid simply repeating this form of description ourselves in a different form.

(Hand 2003: 331)

## 7.4 Conclusion

One of the salient characteristics of knowledge societies is the considerable relative extension of the conditions and capacities of actors that allow for a reconfiguration of social action. The process of globalization is completely in accord with the significance of modernization as an extension of capabilities to take action, and therefore as nothing more than the territorial extension of opportunities for action beyond the borders of the nation-state.

(Stehr 2001: 18)

In this chapter, I strengthen the usefulness and significance of everyday experiences and soft skills for innovation mainly in light of the contingent and ideographic narratives found in the YLUG. I analysed the digital language and exchange on the YLUG mailing list to understand how everyday meanings and practices reproduce, shape and depend on FLOSS. My observations on the YLUG network are similar to those of Silvonen's (2002) study on FLUG. He characterises Linux User Groups (LUGs) as an essential part of the complicated network comprising different communities in FLOSS, existing virtually but also anchored in local social activities. Their activities are grounded on a series of problem-solving peer-supported collaborative links. I share the same view as Silvonen that LUGs offer an arena for local articulations of 'Linux' and behave as 'translators' between developers and lead users as they negotiate over software problems. In this regard, I interpret these narratives as portraits of the relationship between the process of representation and the production of digital epistemologies about the right sort of questions to ask of software problems. Consequently, the traditional view of knowledge codified in an expert-oriented intellectual heritage can be challenged in terms of reflexivity, presentation, and meaning.

The aspects of experience and social networking are particularly valued within the FLOSS social world. As seen in the documents (e.g. magazines, books and conference talks), products reviews based on users' experiences or tips/tricks

based on users' daily experiments are highly regarded within FLOSS. This is also why new technologies such as blogging and wiki are welcomed. The value of experience can be seen in the book *Building Wireless Community Networks* (Flickenger 2003). The book covers the author's real-life experience with the Sebastopol Community Network (NoCAT), a multi-tiered network that provides wireless access for O'Reilly employees and free Web browsing to anyone in the area who has a Wi-Fi card in his or her computer. He describes his experience in using 802.11b, selecting the appropriate equipment, finding antenna sites, and coping with the general problems of outdoor networking. It is said that the book has helped thousands of people engage in community networking activities. At the time, it was impossible to predict how quickly and thoroughly WiFi would penetrate the marketplace. Today, with WiFi-enabled computers almost as common as the Ethernet, it makes even more sense to take the next step and network one's community using nothing but freely available radio spectrum. This is a clear case of socio-technical innovation (i.e. the concept of social networking and the technical wireless technologies) based upon a person's experience and needs.

Users' experiences and problem-solving processes are gaining increasing attention in computer science as well as in industry. As a research project about user frustration in the use of information and computing technology, conducted by Bessiere *et al.* (2002), shows,

The frustration generated by these problems can be personally disturbing and socially disruptive. Psychological and social perspectives on frustration may clarify the relationships among variables such as personality types, cultural factors, goal attainment, workplace anger, and computer anxiety. These perspectives may also help designers, managers, and users understand the range of responses to frustration, which could lead to effective interventions such as redesign of software, improved training, better online help, user discipline, and even resetting of national research priorities.

(Bessiere *et al.* 2002)

Despite the diversity within the FLOSS social world, the origin of FLOSS was initially rooted in the hacker culture in the 1970s. The traditions of the hacker culture, which embodies a self-performed and self-expressed engagement with the software innovation system are still apparent as the YLUG and other cases discussed above. In the previous chapter, I argued that FLOSS innovation expressed a hybrid process combining the formal and the informal. Unlike mainstream software innovation taking place in a formal setting (e.g. the office, laboratories or classrooms etc.), open source practice is institutionalised in industrial and governmental organisations but still maintains a strong informal collaboration with the community. In this chapter, I have explored the narratives of the local Linux user group and conceptualise these as tacit knowledge made public, typically hidden within conventional programming. I also suggested that the informal interactions and relationships between the YLUG members shape and are also being shaped by the global FLOSS knowledge system, yet at the same time that they serve as a source of innovation that contributes to the FLOSS innovation itself. The YLUG as a community of OSPs, is drawn on by local and global to provide new knowledge and solutions to emerging problems that the ongoing evolution of software language(s) creates.

I also argued that the FLOSS mode of innovation in deconstructing (or reverse engineering) software, can thereby create products and services better oriented to usability and accessibility. It could therefore be applied in the field to empower users with marginalized needs or with limited access to information technologies. However, I also argued that care need to be taken over the tendency towards imposing technocratic solutions on local populations, especially in developing countries. We need to attend to the context within which information and knowledge are used, and how this context gives meaning to both: as Panofsky (2003) says:

Knowledge gains meaning, power and legitimacy from its context and must be studied as such.

(Panofsky 2003: 22)

To conclude, I argue that, in sociological terms, the development and

implementation of FLOSS cannot be de-contextualised from the diverse settings within which it is found. As a researcher, one needs to try to engage with such settings – as I did with the YLUG group – to understand the interactions between actors and actants. It is widely argued that cultural values are getting more unstable, fractured and complex than previously, leading to a greater fluidity of meaning and effects. Theorists such as Giddens and Beck have argued that we are experiencing a processes of ‘individualisation’ in which people are increasingly reflexive about their identities, values and actions. This phenomenon corresponds to my observations of the heterogeneous FLOSS social world where diverse actors interplay and negotiate through multiple re/presentations of different interpretations of hacker culture and a range of key concepts within FLOSS. To continue such empirical investigation on the relationship between FLOSS-led technology and society, future research should focus on people’s experiences and understandings of expertise in diverse cultural settings. This avenue of analysis will contribute to our understanding of the socio-technical change made possible by FLOSS, both in terms of its extent and limits.

## **Chapter 8 Conclusion: Towards a (Open) Knowledge-based Information Society**

### **8.0 The Contextualisation of Hacker Culture and FLOSS Innovation**

To critique the popular stereotype of hackers as isolated, socially marginalised, deviant and indeed criminal actors in the computer world, and to interrogate the limited work in sociology that has located hackers within a specific subcultural group, I embarked on a detailed exploration of the social world of hacking, and found considerable diversity within what I called the community of practice of hacking. Through use of ethnographic research methods, I have explored hackers' practices and their relationships with the computing world and the wider society from a socio-technical perspective. In the early stages of my journey, I soon realised that there is no single identity that can be allocated to or is descriptive of "the" hacker. Instead, hacking performs through a range of self-expressions and self-performances in cosmopolitan society. The cultural and social diversity of the hacker community comprises actors from various social-technical backgrounds who inhabit and interact through their engagement with a common platform, based on OSP. At the same time, though such diversity is evident, I have argued that we can find a constellation of im/material practices shared collectively among members in this hacker social world, particularly among those engaging in developing or using free/libre open source software (FLOSS). Whereas the boundaries of different individuals or organisations remain evident, these collective practices blur boundary lines and enable cross-boundary activities to take place. Given that, a particularly important feature of the identity of hacking relates to its radical hybridity and fluid mobility as actors engage in various on- and off-line activities across social worlds. I have argued that a defining feature of this broadly based hacker social world is a dynamic culture of knowledge-sharing and innovation which is expressed in a much more formalised way elsewhere in the production of conventional ICT products and services. Whereas the effects and role of hacking might be regarded as generative of tensions in and disruption to the information system, the development of FLOSS has, I argue, provided a way in

which such tensions may be overcome.

FLOSS allows software source code to be used, copied, studied, modified and redistributed freely. Derived from the hacker social world, the practices of sharing and reusing code have been extended as a way of enrolling new actors into the mainstream software innovation process, and of engaging the knowledge and enthusiasm of what are in effect ‘volunteer’ programmers. For instance, FLOSS projects are typically developed using open tools, such as PHP, Perl, Python, Java and Unix. Working in these environments reduces ‘lock-in’ with specific vendors or software packages. As a result, open platforms also increase the amount of supporting software and experience that is available, and reduces the cost of employing developers. In other words, FLOSS not only reduces development costs, but also demonstrates strengths including interoperability, customisation, rapid problem resolution and community support. Being an approach maintains its software infrastructure collectively, this type of development signifies a community-based innovation (Shah 2003) where both social and technical ‘experiments’ (diverse forms of innovation) are carried out. That said, FLOSS has the potential for a more extensive deciphering of software technologies, circumventing the existence of software monopolies, and making knowledge flows more fluidly across boundaries. In this innovation process, hacker ethics are articulated in actors’ everyday practices and intermingled with other ethics. Hacking practices, which were deemed informal, localised and unconventional, have been institutionalised in the wider information society such that, as we saw in the last chapter, they now address what might be called ‘public knowledge politics’. This type of innovation challenges the conventional industry-led innovation pattern and the superiority of professional expertise through encouraging public participation in knowledge making and transferring and codifying local knowledge and tacit skills, and translating them into more formalised and authorised expertise.

To assist our understanding of such a heterogeneous and contingent innovation system, I placed my main actors, FLOSS developers and/or users, flexibly along a spectrum of contrasting identity types (particularly, ‘outsider’, as an informal actor (e.g. hacker), and ‘insider’, as a formal actor in the innovation system), to monitor

the cross-boundary processes and clarify socio-technical practices (i.e. boundary practices) surrounding software generation. These hybrid identities vary according to different spatial, temporal, biographical and opportunistic conditions. The boundary practices and identities found among these amateur-expert hybrids in the information system are useful in exploring how FLOSS has been and is developed and shaped. FLOSS research in evolutionary economics often fails to contextualise the negotiations between actors and actants, and instead characterises FLOSS innovation as being a smooth and linear process, exaggerating both its altruistic membership or its technically efficient dynamic. While clearly the influence of a subscription to open standards and some important technical developments in software (such as the Linux kernel programme, the KDE desktop system, the Apache HTTP server, OpenOffice.org) suggest the strength of the FLOSS innovation model, we should always locate such developments in the wider environment, since this is much less codified, formalised and in these senses less constrained than the conventional innovation system. The hybrid FLOSS businesses, though they are proposed to play a part in the new knowledge economy, are also caught up in debate over the status of 'public goods' and 'intellectual property', both an matter of debate in the wider FLOSS 'social' (e.g. Bezroukov 1999). These issues intertwine and mingle together. This affirms the STS perspective that innovation is not a linear but a heterogeneous and contingent process. FLOSS should not only be seen as a cluster of software scattered in the ICT world; rather its innovation system might be seen as a particular type of ecology that runs through the ICT world in more or less evident ways, and which fosters the collaborative creation of knowledge and through which an environment for innovation and human resource development, both social and technical, is provided.

### **8.1 Summary of the Chapters**

This thesis offers a contribution towards the empirical and theoretical analysis of the socio-technical dynamics of the FLOSS innovation system. I will now review some of the core arguments that I have developed in earlier Chapters.

After the introductory chapter, chapter 2 provided a theoretical account that

supported existing arguments that link the hacker community with FLOSS development. I argued that a hacker identity is not pre-given and hacker culture is not a fixed culture or subculture. Drawing on the social world perspective, I argued that we should see how and where we might find and describe a ‘hacker social world’ and thereby to explore how diverse actors engage in collective practices but give and negotiate different meanings to their behaviour. The notion of ‘hacker’ acts as a boundary object that allows individuals to articulate their perceptions of, and perform collective practices in, the hacker social world.

Subsequently, I proposed a ‘constellation’ of collective practices to capture and configure the way hacking is embedded and embodied in the hacker social world. I argued that, these collective practices, originating from the informal milieu found within hacker culture, have been institutionalised in recent years, with regard to the deployment and employment of FLOSS. In engaging in these collective practices, actors, nevertheless, assign different meanings to and reflect different values in their hacking behaviour, particularly in regard to debates about the freedom of information and the place of intellectual property rights that preoccupy conventional information society. To avoid getting locked into debates about the ‘real’ meaning of hackers, I employed the theoretical framework of “social worlds” from social studies of science, introduced and discussed in chapter 3, to enable me to attend to the heterogeneity and contingency of the field. This inevitably required a focus on the everyday and mundane activity of those who engage in hacking, and so led to my adopting a more ethnographic (both virtual and physical) approach to my fieldwork. Through examining the multiple readings and doings of this activity, I not only developed a socio-technical understanding of software technology that is generated in a particular way (i.e. through community-based innovation), but also have articulated more fully the primary features of the FLOSS-led innovation pattern that might be useful in regard to the development of policies explicitly geared towards collaborative innovation in the field of software development.

Drawing on my empirical fieldwork, in chapter 4 I looked in more detail at the ways in which social actors negotiate and express, perform, write, and code their engagement with the hacker culture. I also provided an analysis that explained how

the hacker social world interacts with the mainstream ICT social world. In light of the complex composition and structure of the hacker social world, I argued that a sociological analysis of the socio-technical relationships between actors from different social worlds and the communication processes through which they negotiate and handle diverse software needs is necessary. I also argued that the materiality of artefacts (e.g. software applications and source code) have a degree of affordance that allows an identical technique or software tool to be used in different contexts to produce quite different results. This points to the need for the social analyst to recognise movement across boundaries and the need to avoid making or seeking clearly defined categories of actor/behaviour in trying to understand the hacker social world. Given this, I then proposed an approach to technological innovation systems that, while allowing some sense of technological determination, embraced too the notion of the socio-technical construction of technology.

In chapter 5, I looked at the innovation process of EMACS, a FLOSS product and project, to explore the interactions and relationships between actors and actants in an innovation system emerging from the social world of hacking. In the innovation process of EMACS, the practice of social networking played an important role in enrolling new actors into the networks. The innovation was thus embedded in the practices of and the intricate relationships between actors (institutional, organisational, or individual) to solve problems. In analysing these mutual engagements in developing the boundary object (i.e. EMACS), one sees how different readings of hacker ethics are articulated, interpreted, and performed. In looking into the nature of negotiations over diverse readings of problems, and cooperation based on the socio-technical interests commonly shared among actors to solve problems, the explanatory framework suggests that the socio-technical potentiality of technologies can be developed in a highly open and heterogeneous environment, where various sources of innovation are brought about to mobilise the ongoing development of FLOSS itself. The FLOSS story reinforces my argument that there is no self-defining, linear cause-and-effect process of innovation. Technological innovation is a contingent and heterogeneous process particularly in terms of highly individualised and plastic socio-technologies such as software. Such an approach to analysing the genealogy of problems, both social and technical, not only contributes to our understanding of where innovation

comes from, but also helps to understand the identity of and interactions between actors in the innovation process. In looking at the process of how a problem is identified, addressed and tackled through networking and negotiations, FLOSS-led innovation I suggested does not necessarily conform to the 'official' picture of open-source development.

Once the classic development of EMACS had been elucidated, I went on to describe how the FLOSS innovation process is complicated further through its involvement with the private sector. The business model of Linux has intrigued other (non-OSS) social actors in the ICT industry and led many to embed OSPs in their innovation routines, which in turn will shape the innovation process. Chapter 6 explored how FLOSS develops its affordance in institutionalising an emerging range of collective practices in a variety of ways that codify and regulate them (e.g. licences, market distributors, standards etc.) yet still allow actors to meet diverse needs and interests, particularly those that are of a commercial and non-commercial nature. Despite the cultural divergence of the FLOSS social world described above, there are dominant 'grammars' of FLOSS development emerging (i.e. OSPs), which effectively shape the broader software innovation system. The collective engagements in the OSPs bring diverse actors into play on a common ground to develop FLOSS. In referring to the community of OSPs, I described a hybrid innovation pattern to illustrate the collaboration between those engaged in the social world of hacking, mainly in the public domain, and those located in the profit-oriented commercial sector. In my analysis, I developed some new ideas about the cooperation between OSS companies and the FLOSS community, both acting globally and locally. The community of OSPs allows a broader range of social interests to have some input into software development. Business is there to make sure that the "input" in software development is ruled by trade and profit (with the involvement of governments in supporting or regulating the development of FLOSS and related issues such as IPs), whereas the free software community brings the broadest range of social interests. Through such inclusivity and transparency a greater number of social actors contribute towards the future impact and utility of ICT software. The public-private collaboration that involves the crossing of political and economic borders is emblematic of the mobility of FLOSS developers and their work, and points to the hybrid identity of developers,

bounded and shaped by hacker ethics and business ethics jointly. Because *the boundary of expertise has been redrawn and reshaped through this process, the innovation pattern is more dynamic, yet also more unstable and so difficult to predict. Managing risk and uncertainty within the innovation process thus becomes another issue in this type of innovation model.*

The FLOSS innovation paradigm articulates a community-based innovation model assembled with common interests and built upon collective practices, embracing the heterogeneity and contingency in the social world. While FLOSS gains ever-greater attention from different economic and social interests and is rapidly adopted by private as well as public organisations worldwide, this does not necessarily suggest the homogenization of FLOSS culture anchored in a common grammar of open-source practices. Unlike most official documents that emphasise the low-cost, non-proprietary and module-like characters of FLOSS, I paid particular attention to the formation and evolution of actual socio-technical networks engaging in OS issues. This was exemplified in the case study of YLUG in chapter 7 that explored a mundane context of FLOSS innovation embedded in everyday peer practices and dialogues. In so doing, I sought to investigate the specificities of FLOSS innovation ‘on the ground’. My enquiry challenged the dominant view that software development requires specialist and technocratic resources, and described instead a quite different set of practices that subsumed the designer/user distinction as well as that between lay and expert. The narratives of YLUG that I described, detail the process through which localised knowledge displays a form of expertise, but which does not thereby depend upon formal credentialism to secure recognition of authority. Practices in YLUG enable the codification of local knowledge and its translation into more formalised and sophisticated understanding of problem sequences and their solutions, without this necessarily consolidating (and homogenising) as a form of ‘professional expertise’ and protocol. To demonstrate the values of local tinkering, soft skills and multicultural communications in FLOSS innovation, I investigated how locally defined software problems and locally crafted solutions towards the problems are codified and translated ‘upwards’ or laterally within FLOSS innovation through ongoing hands-on practices and debate that circulate freely and widely. Thereby, the tacit knowledge and tinkering skills that are anchored in everyday hacking and

FLOSS practices are mobilised and gain value in the wider FLOSS innovation system and have influenced the ICT innovation system at a *glocal* level. But in doing so, they do not thereby become black-boxed, stable and homogenous platforms for globalised users, as is the case in conventional proprietary software.

The open feature of FLOSS development suggests the possibility of decentralised and low-cost development that promises more space for configuring, experimenting, and improvising software products and services. But meanwhile the artefacts carrying information flows also shape actors' practices and perceptions. The collaborative community built on social networking indicates a virtual learning environment allowing information to travel and to be codified as various forms of knowledge. Such a community-based innovation model not only entails a new paradigm for technological sustainability in software design, but also secures social sustainability in the network. I argued that FLOSS-based systems act as more appropriate apparatus "that can help to encourage the majority of citizens to acquire the capabilities or new media literacies necessary for a democratic dialogue" (Mansell 2002: 419; Sen 1999). Being more liberal and open, FLOSS bridges the so-called "digital divide" more effectively than conventional top-down thinking that in a technocratic and linear way endeavours to remove barriers to wider adoption of the new media without taking cultural differences and local needs into account. However, I also argued that while the FLOSS implementation seems to be able to address "digital entitlements" (Mansell 2002) more properly in bringing digital independence to the local people, how local problems are understood should remain problematic, a matter for reflection and debate, especially when problems are addressed by NGOs and FLOSS workers. Instead of improving living conditions of the poor directly and bringing them improved economic purchasing power, most FLOSS implementations seek to bring about gradual cultural change in access to networking at the local and the global levels. While FLOSS helps developing world regional communities by improving ICT capacity and empowering local users, the relationship between the ICT expertise and the local cultures should be examined more carefully in order not to lose sight of the need for economic resource allocation to poorer communities.

In showing the play and diversity of discourses in various FLOSS-related conferences (chapter 4), in the development of FLOSS projects such as EMACS (chapter 5), in a hybrid innovation system where the private sector (business) and the public sector (the community) co-operate yet also collide (chapter 6), or in amateur-led user groups such as YLUG (chapter 7), the configurations of the FLOSS social world and its expression in the form of everyday hacking practices allow us to reject the “representation of artefacts as mere tools or autonomous tyrants”, and argue ‘instead that technological, conceptual, and moral changes are webbed together in everyday practices’ (Davison 2004). That said, in looking into the nature of negotiation (e.g. diverse readings of problems and doings of solutions) and cooperation (e.g. engaging social interests and network together to solve problems), this thesis offers an analysis of the FLOSS innovation, explaining how the ambivalence of hacker culture and FLOSS technologies are “differentially managed through [their] in-scription, de-scription, and re-scription by different groups within specific contexts” (Hand 2003: 330). Human actors and non-human actants are co-constructed through the practice and materiality on which each depends. That said, actors and artefacts both shape and are shaped by the multiple dynamics of both local and global contexts. Such dynamics lead to the institutionalisation of OSS relationships and the creation as well as breaking down of boundaries over time. The FLOSS innovation system based on social networking and knowledge sharing denotes a new pattern of community-based, peer-designed ICT innovation. While the problem-based driver of this form of ICT innovation may suggest an incremental rather than dramatically changing innovation path, the social innovation that it depends on might be regarded as itself quite ‘radical’.

## 8.2 Research Implications

Within this thesis, the overall argument has principally concerned how the meaning of FLOSS is managed and negotiated in and through an innovation process that is quite distinct from that found elsewhere how cross-boundary activities are carried out, how identities of and relationships between hybrid actors are managed, and how we might conceptualise and understand emerging arrangements of FLOSS, its modes of innovation and forms of knowledge and

identity in light of these events within and across social worlds. I argued that exploring the heterogeneity of discourse is central to understanding FLOSS in socio-technical terms. In a similar analogy to other recent studies of the Internet culture, the FLOSS innovation system encircles various locales where socio-technical problems are *glocally* defined/redefined. It can be understood to express a culturally diverse arrangement of technical artefacts, models of socio-technical orders, and discourses and practices of the self, operative within different institutional and organisational sites (ibid.). This represents a significantly different concept to the work found in innovation studies. The popularisation of ICT in the society is neither a property nor an effect of innovation, but locally defined and articulated.

Whilst contingent, informal and tacit knowledge have acquired serious academic attention from both the perspectives of evolutionary economics and social shaping of technology (cf. Metcalfe 1998; Faulkner & Senker 1995; Winter 1987; Dosi 1988; Fleck & Tierney 1991), these conceptualisations of knowledge used in innovation focus on how knowledge can be transferred and distributed between firms (evolutionary economics), or amongst individuals and organisations (social shaping of technology), in formal and hierarchical environments. Whereas my studies on FLOSS resonate with these earlier works on the value of contingent and tacit knowledge, I argue that the knowledge locally held in hands of multiple information carriers (individuals and organisations) not merely is distributed and flows between formal institutions and organisations (e.g. firms, governments, public/private research centres), more importantly, it travels fluidly amongst individuals and institutions through both formal and informal channels. The intermingling between the formal and informal factors makes socio-technical interaction more complex, dynamic and precarious than had been considered thus far. This feature echoes what Callon & Rabeharisoa (2003) have written about, namely, “research in the wild”, referring to the process through which lay people develop their understanding and knowledge of a shared problem/issue, especially through the web, to challenge established views on an issue or problem. New forms of techno-science-society interactions, notably a hacker-typed amateurism and expertise described in the thesis, in which lay knowledge is valued in the co-fabrication of techno knowledge, contributes to innovation systems in a broad sense. Though the level of competence differs, the celebrated “amateurs as

experts” (Waterton 2003) concept widely appears in various areas such as the environment, health and medical technologies (e.g. Law and Mol 2002). In favour of participatory mechanisms, policy makers also attempt to broaden participation and inclusion in policy-making networks. The phenomenon of enlarging citizen engagement in technological innovation processes addresses the issue of the declining role of science in the rationalisation and legitimisation of public actions (Ezrahi 1990). There is however a need for further inquiry into the processes of the enrolment of new actors in innovation and policy-making networks, their progress, benefits and/or problems. Furthermore, forms and modes of citizen mobilization and expression, including (new) social movements, new forms of participation, consumer activism, patient/user groups, and indigenous peoples should all be included to build a collaborative learning community. Such a mutual-learning environment, virtual or real, provides some means of what Giddens (1991) terms “reskilling” in a postmodern society:

‘Reskilling’ – the reacquisition of knowledge and skills – whether in respect of intimacies of personal life or wider social involvement, is a pervasive reaction to the expropriating effects of abstract systems. It is situationally variable, and also tends to respond to specific requirements of context. Individuals are likely to reskill themselves in greater depth where consequential transitions in their lives are concerned or fateful decisions are to be made. Reskilling, however, is always partial and liable to be affected by the ‘revisable’ nature of expert knowledge and by internal dissensions between experts. Attitudes of trust, as well as more pragmatic acceptance, scepticism, rejection and withdrawal, uneasily coexist in the social space linking individual activities and expert systems. Lay attitudes towards science, technology and other esoteric forms of expertise, in the age of high modernity, tend to express the same mixed attitudes of reverence and reserve, approval and disquiet, enthusiasm and antipathy, which philosophers and social analysts (Themselves experts of sorts) express in their writings.

(Giddens 1991: 7)

Additionally, in recent years, a concern about sustainable design has emerged in

software engineering. To develop responsible products and services, community-based innovation is often regarded as a more appropriate and effective approach. A community of practice(s), notably seen in the FLOSS social world, might serve as a strategic *innovation space* engaging with diverse perspectives and experiences, providing a platform to generate high quality, innovative ideas. Deriving from hacker culture, the FLOSS-led innovation pattern facilitates peer-production in embracing heterogeneity and contingency in everyday practices, and enrolling and mobilising diverse experiences and resources for further innovation ‘in the wild’. The development of FLOSS thus appears to be a compelling case demonstrating the fact that the innovation processes found therein act as catalysts to stimulate new thinking and viewpoints, both at the local and the global levels. It serves as a prototype to manage the risk and the social sustainability of innovation in the process of knowledge creation and production. This trend also suggests that future techno-society is a knowledge-based society where citizens are empowered through mutual interaction and democratic participation (Metcalfe 1998).

### 8.3 Future Research

Some economists note that, in the evolution of the computer industry, specialisation and the division of labour helps to create a stable environment for further knowledge growth (Jackson, Mandeville & Potts, 2002). While this FLOSS phenomenon may reflect this perspective partially, creating increased, and also novel, specialist competencies, I hold a sceptical view on what some researchers on FLOSS studies suggest, viz. that the FLOSS innovation system is coherent and can be stabilised (e.g. Lakhani & von Hippel 2002). In fact, with the commercialisation of FLOSS and growing deployment and employment of FLOSS worldwide, the negotiation between the public and private sectors complicates the innovation process and makes such an innovation system constantly dynamic and complex. More research needs to be done to understand FLOSS and the communities in which the software is created and how this is appropriated, and when not, within a more commercial FLOSS environment.

Whilst everyday FLOSS activities are worth exploring to learn about the interactions between human actors (e.g. users, developers) and non-human actants

(e.g. software, source code, hardware), the ways in which production and consumption (in a broad sense, not merely in terms of economic purchase) in ICTs should be examined as well, particularly how FLOSS is shaped by identities such as gender, sexuality, race and ethnicity, and embodiment. Additionally, the roles, rights and responsibilities of software developers and/or users in both public and private domains are worth exploring. More empirical and theoretical studies from interdisciplinary perspectives are also required to understand the collective processes of knowledge production and distribution. Contextualising innovation stories are required in order to understand the substance of the FLOSS innovation more fully, particularly about the following topics:

1. Organisational structures and processes within or across the FLOSS social world. How is FLOSS created in practice? How is knowledge shared? How are bugs solved?
2. Ethics and policy implications. Given that FLOSS is being rapidly adopted, a growing concern will be related to issues like ethics and policy formation. Should governments adopt a FLOSS policy? Who are the stakeholders? How are the virtues of the 'network society' changing in response to FLOSS?
3. *Glocalised* Perspectives. Even though FLOSS seems to be a global phenomenon, most of the attentions and resources are focused in the developed part of the world. How are software developers who come from other countries like India, Brazil, etc. involved in FLOSS? Is FLOSS truly a device to decrease the digital divide? How is the FLOSS implementation process to act in different locales?

A FLOSS-oriented innovation model could be identified in the ICT field where innovation dynamics are built and emerged through social networking and knowledge sharing. Whereas I propose that such a system may epitomise a formal model in ICT innovation systems, I also highlight the role of the local and situated knowledge. Following MetCalfe (1993) who suggests that it is important to study patterns of innovation at both the macro level yet rooting this in the analysis of change at the level of micro phenomena, future FLOSS studies should embrace

both macro and micro perspectives to conceptualise and contextualise FLOSS innovation in *glocal* environments. In so doing, our understanding of ICT innovation will be in a better position to understand the articulation and iteration between everyday hacking practices, their constellation in the social world of hacking and their mobilisation and codification within the FLOSS community of practice.

# Appendix A: Interview schedule

## FACE-TO-FACE VERSION

Date and Time:

Name		Sex		Nationality	
Occupation				Email	

4. What is your current job? What are you currently working on?
5. What is your experience of using Linux?
6. Have you been involved in the software innovation process?
  - \* How do you come up ideas for new projects?
  - \* How do you write codes/program? Alone or through team work?
  - \* How do you make sure of the reliability of the program? (debug)
  - \* Do you get feedback? How do you deal with the feedback?
  - \* How do you distribute the program?
7. What in your view are Linux's strengths and weakness in terms of its technological base?
8. What in your view is the crucial elements of a successful software project?
9. What have been the sources of the idea for the projects you are working or have worked on?
10. To what extent do you maintain informal personal contacts with scientists and programmers working in commercial or government sectors? Do you have formal collaborations with them?
11. What would you do if you have a problem for which you don't know the answer? What source do you consult first: colleagues, external contacts, literature? In practice, which sources are most likely to provide or point you to the solution you need?
12. For the benefit of an outsider, what are all the different kinds of technological inputs which typically your project requires in the course of the Linux development? (How much of the knowledge or information you require is tacit knowledge, ie. not available in published form? How much of the expertise required derives from formal training in programming, as opposed to experience gained 'on the job' or learning by doing?) Is it easy for outsider to access the core innovation system?
13. Could you rank (1, 2, 3) the more important inputs in terms of the scale of contribution to your new software development efforts?
14. What are the channels through which you obtain inputs from the Linux community? Literature, personal contacts: informal networks and formal collaboration, or others? In your experience, which of these channels makes the greater contribution to software innovation flows from the Linux community? Do you obtain tacit knowledge from any these sources? How?
15. To what extent does the impact of the Linux community inputs vary according to the particular project or type of work you are engaged on? Please give examples.
16. On the basis of your personal experience, can you compare the nature and extent of your linkage with the Linux community in software technology with

that in other situations? (same company other areas of technology, other companies)

17. Would you like your linkage with the Linux community change in the future? How would that be?
18. Could you have a look at the bullet points of practices (see the note below)? Is there any point similar to your activities in doing software projects? If not, could you summarise your software activities in your words? Do you think that software innovation is fully captured by the list? Are there specific innovation activities that are peculiar to Linux that should be added (that differ from mainstream computing)?
19. Some programmers engaged in Linux development have considered themselves as computer hackers. How do you think about that? Do you regard yourself as a hacker as well?

(Note):

The list of practices shown to the interviewees:

- 1) Interest in tackling software problems and resolving them.
- 2) Writing challenging scripts to explore software vulnerabilities.
- 3) A strong interest in decryption, code-breaking and code making.
- 4) Writing creative scripts and sharing them.
- 5) Developing novel hardware and sharing the proprietary information on which it is based.

## Appendix B: Interview schedule

(E-MAIL VERSION)

Hi there,

-- Apologise if you have seen this message before from other lists --  
-- If you have helped me with this questionnaire, thank you very much --  
-- If you haven't done so, here is your second chance doing something good--

This is Yuwei Lin from the university of York, UK. I am a PhD student working in the sociology of science and technology studies. My research is about the institutionalisation of open source software practice. To collect empirical data, I have designed a short questionnaire as follows. I would really appreciate it if you could take 10 minutes answering these questions and email back to me. These data would not only help me conduct my research but also be good for the whole OSS community when being used by other researchers in future research projects. Your information would be made anonymous. Unless you request otherwise, your name will be replaced with a code consisting of two letters and the date of the interview: e.g. AA230501. Other people that were mentioned during the interview will also go unnamed. Anonymity will not, however, cover names of companies, organisations, or government bodies, unless this is specified by you.

It would be fantastic if you can try to send your answers back to me before Christmas so that I can have a Merry Christmas. Thank you very much.

Best Regards,  
Yuwei

----- Questionnaire -----

1. To begin with, can you just give a brief description of your current work? Whether this has formal link to the computer industry?
2. What is your experience of using open source software? What do you think are its strengths and weakness?
3. Are there times when you use the functionality of open source software to develop your project/program? If so, can you give an example (this can relate to writing simple scripts, application developments or even kernel programs)?
4. What sort of issues do you discuss in the informal communication with other open source practitioners? How do you think these issues have changed in recent years?
5. Could you have a look at the following list of activities:
  - 1) Interest in tackling software problems and resolving them.
  - 2) Writing challenging scripts to explore software vulnerabilities.
  - 3) A strong interest in decryption, code-breaking and code making.
  - 4) Writing creative scripts and sharing them.
  - 5) Developing novel hardware and sharing the proprietary information on which it is based.

Are these similar to your own activities when engaged in software projects? If not, could you summarise your software activities in your own words? Do you think

that software innovation is fully captured by the list? Do you think some of these are especially important for open source software compared with a proprietary system. If so, can you say why?

6. Some programmers engaged in open source software development have considered themselves as computer hackers. How do you think about that? Do you regard yourself as a hacker as well?

----- Thank you very much for your help. -----

Yuwei Lin  
Science and Technology Studies Unit (SATSU)  
Department of Sociology  
University of York  
York YO10 5DD  
UK  
Tel. +44-01904-434742  
Fax. +44-01904-433043  
<http://www-users.york.ac.uk/~y1107/>  
<http://www.york.ac.uk/org/satsu/>

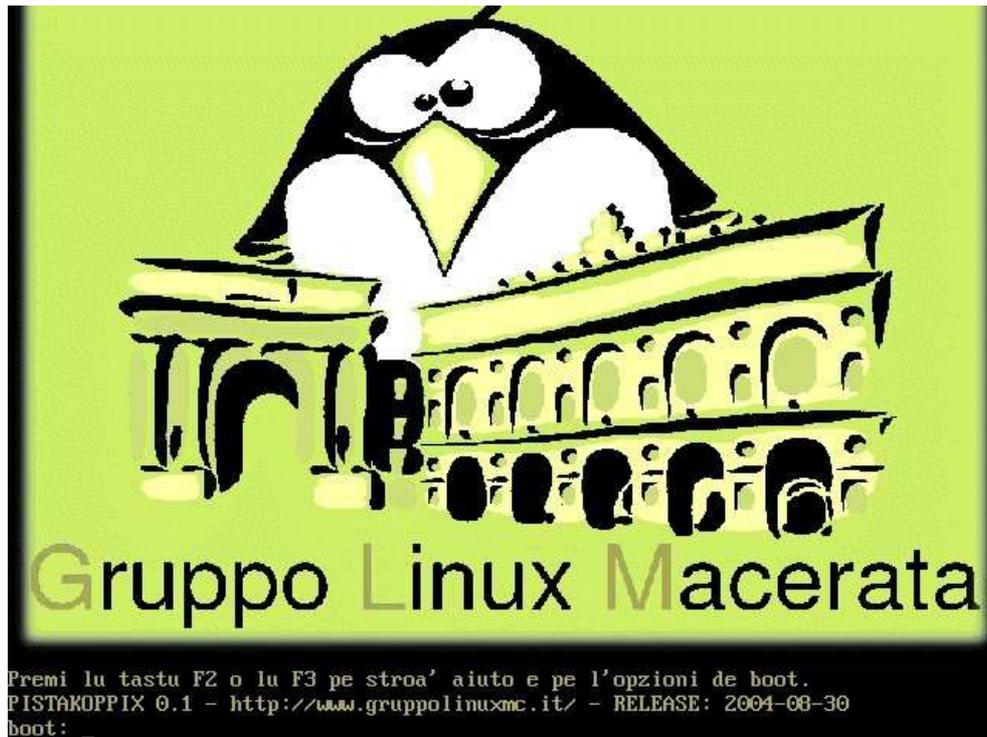
## Appendix C: List of interviewees

<b>Pilot study (Face-to-face)</b>		
Matt	MF	Undergraduate at Manchester University
<b>Hacker Conference (Face-to-face)</b>		
Rudolf	RVDB	Masters student in public relations at the University of Twente
Jaco	JK100801	Dutch ex-hacker and now an accountant, the financial officer of the conference HAL 2001
Rob	RGB	Dutch ex-hacker and now an estate agent (& pirate-cd seller)
Tom	TV	German security consultant (DeCSS expert)
<b>Hacker Conference (Semi-structured Focus group)</b>		
Thomas, Stephan, Rene, Ben	GI090801	
<b>Hacker Conference (E-mail)</b>		
Jean-Denis	JD	French network security expert
Alain	AC	French programmer
<b>Linux Tag Conference (Face-to-face)</b>		
Jean-Paul	JP	French programmer and CTO of an OSS SME
Peter	PC	German programmer
Simon	SO	German programmer and CTO of an OSS SME
Thomas	TK	German computer hardware engineer
Bo	BO	Danish programmer and a SuSE employee
Gabriele	GP	female German programmer working at the federal government
Sugar	DS	USA hacker and Debian developer
Werner	WK	German programmer and CEO of an OSS SME
Klaus	KK	German programmer
<b>LinuxTag 2002 (informal face-to-face chat)</b>		
Alan Cox	AX	UK hacker and a Redhat employee

<b>LinuxTag 2002 (E-mail)</b>		
Nils	NM	German hacker and security expert
Tony	TS	UK programmer
Florian	FB	German instructor for Red Hat Germany
Kurt	KH	Owner of a small company for networks, Java application development, training and consulting.
Matthias	MW	German programmer (student and self-employed)
<b>Summer Source Camp 2003 (Face-to-face &amp; E-mail)</b>		
Eric	EZ	Italian Debian Developer
Ben	BH	US Debian Developer
Jacob	JA	US Network Security expert
<b>Industry (Face-to-face interview)</b>		
Charles	CF	From Vita Nuova Company at York Science Park
<b>Academia (informal face-to-face chat)</b>		
Ewan	EM	YLUG member and PhD student in chemistry
John	JR	Professor at the electronic engineering department at the university of York
<b>Other Email interview</b>		
Alain	AB	Belgian Computer analyst and system-programmer in a bank
David	DY	UK management information systems programmer
Emiliano	EM	Dutch software developer in a multinational printer company
Jeroen	JV	Dutch programmer
M	MR	UK treasurer of the UK's Association For Free Software
Nicolas	NR	R&D engineer
Olivier	OB	French Research Engineer in a higher graduate school (INT/GET)
Peter	PB	Spanish Kernel device driver author
Richard	RI	UK Linux User Group leader
Roger	RL	UK programmer (YLUG member) and Debian developer
Shooby	SBA	Hungarian programmer
Simon	SBR	UK self employed software developer
Thomas	TT	student in computer science
Vincent	VG	Belgian student in computer science

## Appendix D: Customised Linux

### SCREENSHOTS OF A LOCALLY-CUSTOMISED LINUX DISTRIBUTION



*Figure 1: The initial booting*



Figure 2: Starting-up progress



Figure 3: the final desktop environment

## Bibliography

- Abbate, J. 2000. *Inventing the Internet*. Cambridge MA: MIT Press.
- Allen, J. 1993. Friends and Neighbors: Knowledge and Campaigning in London. In R. Fisher and J. Kling (Eds.), *Mobilizing the Community: Local Politics in the Era of the Global City*. London: Sage.
- AntivirusAbout.com July 30, 2002. "KaZaA, Gnucleus, Welcome to Hackville"  
URL (consulted on 7 September 2003)  
<http://antivirus.about.com/library/weekly/aa073002a.htm>
- Bannon, L. J. 1998. 'Computer supported collaborative working: Challenging perspectives on work and technology', in Robert Galliers & Walter Baets (eds.) *Information technology and organizational transformation: Innovation for the 21st century organization*. Chichester: John Wiley & Sons Ltd.
- Barbrook (1998) The Hi-Tech Gift Economy. *First Monday*. URL (consulted on 020504): [http://firstmonday.dk/issues/issue3\\_12/barbrook/index.html](http://firstmonday.dk/issues/issue3_12/barbrook/index.html)
- Baudrillard, Jean. 1988. *Jean Baudrillard: Selected Writings*. Ed. Mark Poster. Stanford: Stanford University Press.
- Bauman, Z. 1998. *Globalization: The Human Consequences*. Cambridge: Polity.
- Bauman, Z. 1999. *In Search of Politics*. Cambridge: Polity Press.
- Baym, N. K. 1995. "From Practice to Culture on Usenet". In S. L. Star (Ed.) *The Cultures of Computing*, (Sociological Review Monograph Series) London: Basil Blackwell. Page 29-52.
- Beck, U. 1994. "The Reinvention of Politics: Towards a Theory of Reflexive Modernization." in Beck, U., A. Giddens and S. Lash (eds.), *Reflexive Modernization: Politics, Tradition and Aesthetics in the Modern Social Order*, pp. 1-55. Cambridge: Polity Press.
- Becker, H. S. (1974). Photography and sociology . *Studies in the Anthropology of Visual Communication*, 1(1), 3-26 .
- Becker, H. S. (1986). *Doing things together: Selected papers*. Evanston, IL: Northwestern University Press .
- Benkler, Y. 2001. Coase's Penguin, or, Linux and the Nature of the Firm.
- Bennett, A. 1999. 'Subcultures or neo-tribes? Rethinking the relationship between youth, style and musical taste'. *Sociology*, 33: 599-617.
- Bessiere, K., Ceaparu, I., Lazar, J., Robinson, J., and Shneiderman, B. (October 2002) Understanding Computer User Frustration: Measuring and Modeling the Disruption from Poor Designs. Research Project HCIL-2002-18 , CS-TR-4409 , UMIACS-TR-2002-89 <http://www.cs.umd.edu/hcil/newcomputing/>
- Bezroukov, N. 1999. *Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism) by Nikolai*

- Bezroukov First Monday*, volume 4, number 10 (October 1999), URL: [http://firstmonday.org/issues/issue4\\_10/bezroukov/index.html](http://firstmonday.org/issues/issue4_10/bezroukov/index.html)
- Bilton T., Bonnett, K., Jones, P., Skinner, D., Stanworth, M. & Webster, A. 2002. *Introductory Sociology*, 4th edition. Hampshire: Palgrave Macmillan.
- Bloor, D. 1997. *Wittgenstein: Rules and Institutions*. London: Routledge.
- Boehm, B. 1986. "A Spiral Model of Software Development and Enhancement". ACM SIGSOFT Software Engineering Notes.
- Boehm B. 1988. "A Spiral Model of Software Development and Enhancement". *IEEE Computer*, vol.21(5): 61-72.
- Bonaccorsi A. & Ricci, C. 2003. "Why Open Source Software can Succeed?". *Research Policy* 32 (7): 1243-1258.
- Borgman, C. L. 2003. 'Designing Digital Libraries for Usability'. In Ann P. Bishop, Nancy A. Van House & Barbara P. Battenfield (Eds) *Digital Library Use: Social Practice in Design and Evaluation*. London: The MIT Press.
- Borras, M. & Zysman, J. 1997. "Wintelism and the Changing Terms of Global Competition: Prototype of the future?". BRIE Working Paper 96B (February). Berkeley, CA.: BRIE.
- Bourdieu, P. 1990. *In Other Words: Essays Towards a Reflexive Sociology* (M. Adamson, Trans.). Stanford, CA: Stanford University Press. (Original work published 1982, 1987)
- Bowker G. C. & Star, S. L. 1999. *Sorting Things Out: Classification and its Consequences*. London: The MIT Press.
- Brennan, L. & Johnson V. (2004) "Technology Management for Corporate Social Responsibility". *IEEE Technology and Society Magazine*, Spring 2004.
- Brian, D. 1994. Cultural production as "society in the making": architecture as an exemplar of the social construction of cultural artefacts, in 'The sociology of culture: emerging theoretical perspectives' edited by Diana Crane. Oxford: Blackwell.
- Brown, S. 1996. *Strategic manufacturing for competitive advantage: transforming operations from shop floor to strategy*. London: Prentice Hall.
- C|Net News.com. August 29, 2001. Governments push open source software. Retrieved May 30, 2004, from <http://news.com.com/2100-1001-272299.html?legacy=cnet>
- C|Net News.com. November 28, 2001. "The root of the problem: bad software". URL (consulted September 28, 2004) <http://news.com.com/2008-1082-276316.html?legacy=cnet>
- C|Net News.com. August 2, 2002. "Hacking their image". URL (consulted on 28 September 2004) [http://news.com.com/Hacking+their+image/2009-1001\\_3-947682.html](http://news.com.com/Hacking+their+image/2009-1001_3-947682.html)
- C|NET News.com. April 1, 2004. "Programmers told to put security over creativity". URL (consulted on 020404): <http://news.com.com/2100-1009-5183634.html>

- C|Net News.com. April 24, 2004. "Linux founder opens door to DRM". URL (consulted on 27 September 2004): [http://news.com.com/Linux+founder+opens+door+to+DRM/2100-1016\\_3-998292.html](http://news.com.com/Linux+founder+opens+door+to+DRM/2100-1016_3-998292.html)
- C|NET News.com. April 27, 2004. "Linux seller licenses Windows Media technology". URL (consulted on 020504): <http://news.com.com/2100-7344-5201352.html>
- C|NET News.com. May 5, 2004. "Computer Associates pins hopes on open source". URL (consulted on 050504): [http://news.com.com/2100-7344\\_3-5206671.html?tag=nefd.hed](http://news.com.com/2100-7344_3-5206671.html?tag=nefd.hed)
- Callon and Rabeharisoa, 2003. Research in the wild. *Technology in Society* Vol. 25: 193-204.
- Callon, M. 1986. "The Sociology of an Actor-Network: the Case of the Electric Vehicle". In M. Callon, J. Law and A. Rip (Eds.) *Mapping the Dynamics of Science and Technology: Sociology of Science in the Real World*. London, Macmillan: 19-34.
- Callon, M. 1987. Society in the Making: the Study of Technology as a Tool for Sociological Analysis. In W. E. Bijker, T. P. Hughes and T. J. Pinch (Eds.) *The Social Construction of Technical Systems: New Directions in the Sociology and History of Technology*. Cambridge, Mass. and London, MIT Press: 83-103.
- Callon, M. and Latour B. 1992. "Don't Throw the Baby Out with the Bath School! A Reply to Collins and Yearley". In A. Pickering (Ed.) *Science as Practice and Culture*. Chicago, Chicago University Press: 343-368.
- Callon, M. 1994. "Is science a public good?", manuscript written for the fifth Mullins Lecture, Virginia Polytechnic Institute 23 March 1993. *Science, Technology, & Human Values* 19 (4): 395-424.
- 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.
- Cancilla, J. 2003. Open Source Software for Windows. TechSoup.org October 30, 2003. Tech Soup. URL <http://www.techsoup.org/howto/articlepage.cfm?ArticleId=523&topicid=2>
- Castells, M. 1996. *The Information Age: Economy, Society, and Culture Volume I: The Rise of the Network Society*. Oxford: Blackwell Publishers.
- Castells, M. and Hall, P. 1994. "Technopoles: mines and foundries of the information economy". In *Technopoles of the World*. London: Routledge. Reprinted in LeGates, R T and F Stout, eds. *The City Reader* (London: Routledge, 1996), pp. 475-183.
- Ceruzzi, P. 2003. *A History of Modern Computing*. 2<sup>nd</sup> edition. The MIT press.
- Chantler, N. 1996. *Risk: The Profile of the Computer Hacker*. PhD Thesis. Curtin University of Technology, Perth, Western Australia.
- Cherlin, E. (2002). Simputer White Paper: Computers for Everyone. Retrieved May 30, 2004 [http://www.upspace.org/academy/Members/cherlin/Simputer%](http://www.upspace.org/academy/Members/cherlin/Simputer%20White%20Paper.pdf)

20White%20Paper/file\_view/.

- Clark, P. A. & Staunton, N. 1989. *Innovation in technology and organization*. London: Routledge.
- Clarke, A. E. 1991. Social worlds/arenas theory as organizational theory. In *Social organization and social processes: essays in honour of Andelm L. Strauss*, edited by D. Maines. NY: Aldine Gruyter.
- Clarke, A. & Montini, T. 1993. The many faces of RU486: Tales of situated knowledges and technological contestations. *Science, Technology, & Human Values*, Vol. 18 No. 1, p. 42-78.
- Clarke, A. E. 1997. A social worlds research adventure: the case of reproductive science. In *Grounded theory in practice*, edited by A. Strauss & J. Corbin. London: Sage publications.
- Clarke, A. E. 1998. *Disciplining Reproduction: Modernity, American Life Sciences and the 'Problems of Sex.'* Berkeley: University of California Press.
- Clough, P. & Nutbrown, C. 2002. *A student's guide to methodology: justifying Enquiry*. London: Sage.
- Cohen, R. & Kennedy, P. 2000. *Global Sociology*. London: MacMillan Press Ltd.
- Cohen, S. & Young, J. 1973. *The Manufacture of News*. London: Constable.
- Collins, H. & Pinch, T. 1993. *The golem: what everyone should know about science*. Cambridge: Cambridge University Press.
- Collins, H. & Pinch, T. 1998. *The golem at large: what you should know about technology*. Cambridge: Cambridge University Press.
- Compaine, B. M. (Ed.) 2001. *The Digital Divide: Facing a Crisis or Creating a Myth?* The MIT Press.
- ComputerWorld.com August 10, 1999. "Reporter's notebook: Hackers on holiday" URL (consulted on 28 September 2004)  
<http://www.computerworld.com/news/1999/story/0,11280,28635,00.html>
- ComputerWorld.com August 16, 1999. "Hackers on holiday network, party" URL (consulted on 28 September 2004)
- Crane, D. (ed.) 1994. *The sociology of culture: emerging theoretical perspectives*. Oxford, UK: Blackwell.
- Davison, A. 2004. "Reinhabiting technology: ends in means and the practice of place". *Technology in Society*, Vol. 26 (1), page 85-97.
- Derrida, J. 1978. "Structure, Sign and Play in the Discourse of the Human Science", in *Writing and Difference*. Chicago: Chicago University Press.
- DiBona, C. & Ockman, S. & Stone, M. (eds) 1999. *Open Sources: Voices from the Open Source Revolution*. Sebastopol, CA: O'Reilly.
- Dolby, R. G. A. 1979. "Reflections on deviant science". In R. Willis, (ed) *On the margins of science: The social construction of rejected knowledge*. Sociological Review Monograph 27. Keele: University of Keele, 1979: 9-47.

- Dosi, G. 1982. "Technological Paradigms and Technological Trajectories: A Suggested Interpretation of the Determinants and Directions of Technical Change". *Research Policy*, 11: 147-162.
- Dosi, G. 1988. "The nature of the innovation process", in Dosi, D., Freeman, C., Nelson, R., Silverberg, G., and Soete, L. (eds.), *Technical Change and Economic Theory*, London: Pinter.
- Driskell, R. B. & Lyon, L. 2002. "Are virtually communities true communities? Examining the environments and elements of community." *City and Community* 1(4):373-390.
- Durkheim, E. 1960 [1893] *The Division of Labor in Society*. Translated by George Simpson. New York: The Free Press.
- Dutton, W. H. 1999. "The Virtual Organisation: Tele-Access in Business and Industry", in G. DeSanctis and J. Fulk (eds.), *Shaping Organisation Form: Communication, Connection and Community*, p. 473-495. London: Sage.
- Dutton, H. (authors and edits) 1999a. *Society on the line: Information politics in the digital age*. Oxford: Oxford University Press.
- Edwards, P. N. 1987. "A History of Computers in Weapons Systems." In *Computers in Battle*, ed. David Bellin and Gary Chapman. 45-60. New York: Harcourt.
- Edwards, P. N. 2001. "Impact" to social process: computers in society and culture, in S. Jasanoff, G. E. Markle, J. C. Peterson & T. Pinch (Eds) *Handbook of Science and Technology Studies*. London: Sage.
- Ellis, R., and C. Waterton (2005), Caught Between the Cartographic and the Ethnographic Imagination: The Whereabouts of Amateurs, Professionals and Nature in Knowing Biodiversity, *Environment and Planning D: Society and Space*, forthcoming.
- Evers, V. 2001. *Cultural Aspects of User Interface Understanding: An Empirical Evaluation of an E-Learning website by International User Groups*. Doctoral Thesis, the Open University.
- Ezrahi, Y. 1990. *The Descent of Icarus*. Cambridge, MA: Harvard University Press.
- Faulkner W. & Senker, J. & Velho, L. 1995. *Knowledge Frontiers: Public Sector Research and Industrial Innovation in Biotechnology, Engineering Ceramics, and Parallel Computing*. Oxford: Oxford University Press.
- Feller J. & Fitzgerald B. 2001. *Understanding open source software development*. London: Addison-Wesley.
- Fielding, N. 2001. "Ethnography", in Nigel Gilbert (ed.) *Researching Social Life*, 2<sup>nd</sup> edition. London: Sage.
- Fleck, J. & Tierney, M. 1991. "The management of expertise: Knowledge, power and economics of expert labour", Edinburgh PICT Working Paper 29, Edinburgh: Research Centre for Social Science, University of Edinburgh.
- Fleck, J. 1988. 'Innofusion Or Diffusation? The Nature Of Technological Development In Robotics', Edinburgh PICT (The ESRC Programme on

- Information and Communication Technologies) working paper No. 4. University of Edinburgh.
- Fleck, J. 1992. "Configurations: crystallising contingency". Edinburgh PICT working paper. University of Edinburgh, Programme on Information and Communication Technologies
- Fleck, J. 1995. "Informal information flow and the nature of expertise in financial services". Working paper series, University of Edinburgh, Management School.
- Flickenger, R. 2003. *Building Wireless Community Networks: Implementing the Wireless Web*. 2<sup>nd</sup> Edition (first edition 2001). O'Reilly.
- FLOSS Final Report of the project "Free/Libre and Open Source Software: Survey and Study". International Institute of Infonomics, University of Maastricht, The Netherlands & Berlecon Research GmbH Berlin, Germany. URL (consulted on 28 September 2004) <http://www.infonomics.nl/FLOSS/report/>
- Foucault, M. & Martin, L. (eds) 1988. *Technologies of the Self: A Seminar with Michel Foucault*. Cambridge MA: University of Massachusetts Press.
- Foucault, M. 1973. *Birth of the clinic: An archaeology of medical perception*. London: Tavistock.
- Foucault, M. 1975. *Discipline and punish: Birth and the prison*. London: Allen Lane.
- Foucault, M. 1978a. *The history of sexuality: The will to knowledge* (Vol. 1). London: Allen Lane.
- Foucault, M. 1978b. The West and the truth of sex. In V. Beechey & J. Donald (Eds.), *Subjectivity and social relations*. London: Open University Press.
- Foucault, M. 1982. The subject and power. *Critical Inquiry*, 8, 777-795.
- Fujimura, J. H. 1986. "The Molecular Biological Bandwagon in Cancer Research: Where Social Worlds Meet." *Social Problems* 35:261-283. Reprinted as pp. 95-130 in Anselm L. Strauss and Juliet Corbin (Eds.) 1997. *Grounded Theory in Practice*. Thousand Oaks, CA: Sage.
- Fujimura, J. H. 1987. "Constructing Doable Problems in Cancer Research: Articulating Alignment." *Social Studies of Science* 17:257-93.
- Fujimura, J.H. 1991. On Methods, Ontologies, and Representation in the sociology of science: where do we stand? In *Social organization and social processes: essays in honor of Anselm L. Strauss*, edited by David Maines. NY: Aldine Gruyter.
- 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.
- Fujimura, J. H. 2000. "Transnational Genomics: Transgressing the Boundary between the "Modern/West" and the "Premodern/East", in Roddey Reid and Sharon Traweek (eds) *Doing Science + Culture*. London: Routledge.

- Fulk, J. & Desanctis, G. 1999. 'Articulation of Communication Technology and Organizational Form', in Gerardine DeSanctis & Janet Fulk (eds.) *Shaping Organization Form: Communication, connection, and community*. London: Sage.
- Gacek, C. & Lawrie, T. & Arief B. 2002. 'Interdisciplinary insights on open source', paper presented at Proceedings of the Open Source Software Development Workshop, 25<sup>th</sup>-26<sup>th</sup> February 2002, Newcastle upon Tyne, UK.
- Garbay, C. 2003. "The role of information science in interdisciplinary research: a systematic approach", translated by Marcel Lieberman. A discussion paper for the website "Rethinking Interdisciplinarity". URL (consulted on 18 January 2004): <http://www.interdisciplines.org/interdisciplinarity/papers/2/printable/paper>
- Garfinkel H. 1967. *Studies in Ethnomethodology*. Englewood Cliffs, N.J.: Prentice-Hall.
- Ghosh, R. A. 1998. Cooking pot markets: an economic model for the trade in free goods and services on the Internet. *First Monday*, 3(3). URL (consulted on 27 September 2004): [http://www.firstmonday.org/issues/issue3\\_3/ghosh/index.html](http://www.firstmonday.org/issues/issue3_3/ghosh/index.html)
- Ghosh, R. A. 2002. Workshop Report: workshop on Advancing the Research Agenda on Free/Open Source Software, 14 October 2002, Brussels, European commission. <http://www.infonomics.nl/FLOSS/report/workshopreport.htm>
- Giddens, A. 1991. *Modernity and Self-identity*. Cambridge: Polity.
- Glaser, B. and Strauss, A. 1967. *The Discovery of Grounded Theory*. Chicago: Aldine.
- Glass, A. L., K. J. Holyoak, and J. L. Santa. 1979. *Cognition*. Reading, MA: Addison-Wesley.
- Greene, L. M. 2001. *Inventorship: The Art of Innovation*. New York: John Wiley & Sons, Inc.
- Hafner, K. and Lyon, M. 1996. *Where Wizards Stay Up Late: The Origins of the Internet*. New York: Simon and Schuster.
- HAL, Hacker At Large. 2001. *Acceptable Use Policy*. Twente.
- HAL, Hacker At Large. 2001. *Script Kiddes' Guide Book*. Twente.
- Hammersley, M. and Atkinson, P. .1995. *Ethnography: Principles in Practice*. (2<sup>nd</sup> edition) London: Routledge.
- Hampton, K., & Wellman, B. (2003). Neighboring in Netville: How the Internet Supports Community, Social Support and Social Capital in a Wired Suburb. *City and Community*, 2(4), 277-311.
- Hand, M. 2003. *Understanding Internet Governance: Cultures of Technology and Self-Transformation*. Doctoral dissertation submitted to the Department of Sociology, University of York.
- Hargittai, E. 2003. *How wide a web? Inequalities in accessing information online*. Unpublished doctoral dissertation, Princeton University, Princeton, NJ.

- Henwood, F, Plumeridge, S and Stepulevage 2000. "A tale of two cultures? Gender and inequality in computer education" in S Wyatt, F Henwood, N Miller and P Senker (eds) *Technology and Inequality*, London: Routledge.
- Heydebrand, W. 1989. 'New Organizational Forms', *Work and Occupations*, 16, 323-357.
- Himanen, P. 2001. *The hacker ethic and the spirit of the information age*. London: Secker & Warburg.
- Hine, C. 2002. 'Cyberscience and Social Boundaries: the Implications of Laboratory Talk on the Internet'. *Sociological Research Online* 7(2). URL (consulted on 14 June 2004): <http://www.socresonline.org.uk/7/2/hine.html>
- Hodder, I. 1994. 'The Interpretation of Documents and Material Culture'. In Denzin, N.K. and Lincoln, Y.S. (Eds.) *Handbook of Qualitative Research*. California: Thousand Oaks.  
<http://www.computerworld.com/news/1999/story/0,11280,36724,00.html>
- Hughes, T. P. 1983. *Networks of power: electrification in western society, 1880-1930*. Baltimore and London: Johns Hopkins University Press.
- Hutchby, I. 2001. 'Technologies, texts and affordances.' *Sociology*, Vol. 35 No 2, pp. 441-456.
- Hutchby, I. 2003. 'Affordances and the analysis of technologically mediated interaction: A response to Brian Rappert.' *Sociology*, Vol. 37 No. 3, pp. 581-589.
- Iannacci, F. 2002, *The Social Epistemology of Open-Source Networks*. Working paper, Department of Information Systems, London School of Economics and Political Science, 2002, pp. 1-28.
- Jackson, M. & Mandeville, T. & Potts, J. (2002) *The evolution of the digital computation industry*. Prometheus, Vol. 20 No. 4, p. 323-336.
- Jenkins, R. 1983. *Lads, citizens and ordinary kids: working class youth lifestyles in Belfast*. London: Routledge and Kegan Paul.
- Johnson, S. K. 1971. *Idle Haven: Community Building Among the Working-Class Retired*. London.
- Jordan, K., & Hauser, J., & Foster, F. (2003). *The augmented social network: Building identity and trust into the next generation internet*. First Monday, 8 (8). Retrieved September 1, 2004, from [http://firstmonday.org/issues/issue8\\_8/jordan/index.html](http://firstmonday.org/issues/issue8_8/jordan/index.html)
- Kaptelinin V. 2002. "Making Use of Social Thinking: The Challenge of Bridging Activity Systems". In Y. Dittrich and C. Floyd and R. Klischewski (eds.) *Social Thinking - Software Practice*. Cambridge, MA: MIT Press. Page 45-68.
- Katz, J. E. & Rice, R. E. 2002. *Social Consequences of Internet Use: Access, Involvement, and Interaction*. The MIT Press.
- Kelly, P. & Kranzberg, M. & Rossini, F. & Baker N. & Tarpley, F. & Mitzner, M. 1986. 'Introducing innovation', in the book *Product Design and Technological Innovation*, edited by R. Roy and D. Wield. Milton Keynes: Open University Press.

- Kelty, C. M. 2001. Free Software/Free Science. *First Monday* 6 (12) URL: [http://firstmonday.org/issues/issue6\\_12/kelty/index.html](http://firstmonday.org/issues/issue6_12/kelty/index.html) (consulted 4 September 2004).
- Klein, A. 2002. "Ethical Hacking Techniques to Audit and Secure Web-enabled Applications". Security paper written for Sanctum, Inc. URL (consulted on 28 September 04) [http://www.sanctuminc.com/pdf/Ethical\\_Hacking\\_Techniques.pdf](http://www.sanctuminc.com/pdf/Ethical_Hacking_Techniques.pdf)
- Kling, R. 1987. Value conflicts and social choice in electronic payment systems, in 'Information technology: social issues—A reader' edited by Ruth Finnegan, Graeme Salaman and Kenneth Thompson. London: Hodder & Stoughton, in association with the Open University.
- Kling, R. and Scacchi, W. 1982. "The Web of Computing: Computing Technology as Social Organization." *Advances in Computers* 21: 3-85.
- Knorr-Cetina, K. D. 1981. *The manufacture of knowledge: an essay on the constructivist and contextual nature of science*. Oxford: Pergamon.
- Knorr-Cetina, K. 1999. *Epistemic cultures: how the sciences make knowledge*. Cambridge, MA: Harvard University Press.
- Kornblum, W. 1974. *Blue Collar Community*. London.
- Krag, T. (2004). *Wireless Networks as a Low-Cost, Decentralized Alternative for the Developing World*. A talk given at the O'reilly emerging technology conference, 9-12 February 2004, San Diego, USA. Retrieved May 30, 2004, from [http://conferences.oreillynet.com/cs/et2004/view/e\\_sess/4697](http://conferences.oreillynet.com/cs/et2004/view/e_sess/4697).
- Kuhn, T. S. 1970. *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press.
- Kyng, M. 1995. "Making Representations Work." *Communications of the ACM*. 38 (9), 46-56.
- Lakhani, K. R. & von Hippel, E. 2003. "How open source software works: "free" user-to-user assistance". *Research Policy* 32 (6): 923-943.
- Lanzara, G. F. & Morner, M. 2003. *The Knowledge Ecology of Open-Source Software Projects*. Paper presented at the ICTs in the contemporary world seminar at the LSE Department of Information Systems on 2 October 2003 at 1430.
- Latour, B. & Woolgar, S. 1979. *Laboratory Life. The Social Construction of Scientific Facts*. London and Beverly Hills: Sage; second edition published as *Laboratory Life. The Construction of Scientific Facts*. Princeton, N.J. : Princeton University Press, 1986.
- Latour, B. 1987. *Science in Action: How to Follow Scientists and Engineers Through Society*. Milton Keynes, Open University Press.
- Latour, B. 1993a. 'On Technical Mediation: The Messenger Lectures on the *Evolution of Civilisation*'. Cornell University, Institute of Economic Research: Working Papers Series.
- Latour, B. 1993b. *We Have Never Been Modern*. Hemel Hempstead: Harvester Wheatsheaf.

- Latour, B. 1996. *Aramis, or the Love of Technology*. Cambridge, MA: Harvard University Press.
- Lave, J. 1988. *Cognition in practice*. Cambridge, UK: Cambridge University Press.
- Lave, J. & Wenger, E. 1992. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press.
- Law, J. 1986. "On Power and Its Tactics: a View from the Sociology of Science." *The Sociological Review* 34: 1-38.
- Law, J. and Mol, A-M. (eds.) 2002. *Complexities*, London: Duke University Press.
- Lea, M., O'Shea, T. & Fung, P. 1999. "Constructing the Networked organisation: content and context in the development of electronic communications", in G. DeSanctis & J. Fulk (eds.), *Shaping Organisation Form: communication, connection, and community*, p.295-324. London: Sage.
- Lee, G. K. & Cole, R. E. 2003. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development" *Organization Science* Vol. 14, No. 6, November -December, pp. 633-649.
- Lessig, L. 2000. *Code and other Laws of Cyberspace*. New York: Basic Books.
- Lessig, L. 2001. *The Future of Ideas: The Fate of the Commons in a Connected World*. New York: Random House.
- Levy, S. 1984. *Hackers: Heroes of the Computer Revolution*. Garden City NY: Anchor Press/Doubleday.
- Liff, S. and Steward, F. 2003. "Shaping e-access in the cybercafe: networks, boundaries and heterotopian innovation. *New Media & Society*, vol 5(3): 313-334.
- Lovink, G. 2003. *My First Recession*. Rotterdam: V2\_Publishing/Nai publishers
- LWN.net. Linux in Spain. Retrieved September 1, 2004, from <http://lwn.net/Articles/41738/>
- Lynch, M. & Woolgar, S. 1988. Introduction: Sociological orientations to representational practice in science, in *Representation in scientific practice*, edited by Michael Luch & Steve Woolgar. Kluwer Academic Publishers.
- Lynch, M. 1993. *Scientific practice and ordinary action: Ethnomethodology and social studies of science*. Cambridge University Press.
- Lyotard, J-F. 1984. *The postmodern condition: A report on knowledge*(trans. G. Bennington and B. Massumi). Minneapolis: University of Minnesota Press.
- MacDonald, K. 2001. "Using Documents", in Nigel Gilbert (ed.) *Researching Social Life*, 2<sup>nd</sup> edition. London: Sage.
- Mackenzie, A. 2001. 'Open source software: When is a tool? What is a commodity?'. *Science as Culture* 10, 4: 541-552.
- MacKenzie, D. & Wajcman, J. (Eds) 1985. *The social shaping of technology: how the refrigerator got its hum*. Milton Keynes: Open University Press.
- MacKenzie, D. 1996. *Knowing Machines: Essays on Technical Change*.

- Cambridge, MA: The MIT Press.
- Maffesoli, M. 1989. "The Sociology of Everyday Life. Epistemological Elements". *Current Sociology* 37 (1).
- Maffesoli, M. 1996. *The Time of the Tribes: The Decline of Individualism in Mass Societies*. London: Sage
- Mansell, R. & Steinmueller, W. E. (2000) *Mobilizing the Information Society: Strategies for Growth and Opportunity*. Oxford: Oxford University Press.
- Mansell, R. 2002. "From Digital Divides to Digital Entitlements in Knowledge Societies". *Current Sociology*, 50(3): 407-426.
- Marx, K. 1867. *Capital*. Vol.1. (Reprint 1990 New York: Penguin).
- McClelland, J. & Silvers, R. 2002. How Open Source Can Open Doors for Nonprofits: Open source software a natural for nonprofits. Tech News November 01, 2002. URL [http://www.uwnyc.org/technews/v5\\_n6\\_a1.html](http://www.uwnyc.org/technews/v5_n6_a1.html) (consulted 4 September 2004).
- McGraw, G. & Felten, E. 1997. *Java security*. New York: Wiley.
- McKelvey, M. 2001. "Internet Entrepreneurship: Linux and the dynamics of open source software", CRIC Discussion Paper no. 44. Centre for Research on Innovation and Competition, The University of Manchester & UMIST.
- McLaughlin, J. & Rosen, P. & Skinner, D. & Webster, A. 1999. *Valuing technology: organisations, culture and change*. London: Routledge.
- Merton, R. K. & Zuckerman, H. A. 1968. "The Matthew Effect in Science: the Reward and Communication Systems of Science are Considered". *Science* 199, 3810, 55-63.
- Merton, R. K. & Zuckerman, H. A. 1971. "Patterns of Evaluation in Science: Institutionalization, Structure and Functions of the Referee System". *Minerva* 9, 1, 66-100.
- Merton, R. K. 1957. "Priorities in Scientific Discovery: A Chapter in the Sociology of Science". *American Sociological Review* 22, 6, 635-659.
- Merton, R. K. 1961. "Singletons and Multiples in Scientific Discovery". *Proceedings of the American Philosophical Society* 105, no. 5 (October): 470-486.
- Merton, R. K. 1963. "The Ambivalence of Scientists." *Bulletin of the Johns Hopkins Hospital* 112: 77-97.
- Merton, R. K. 1973. *The Sociology of Science: Theoretical and Empirical Investigations*. [Edited and Introduced by Norman W. Storer]. Chicago: University of Chicago Press.
- Metcalf, J. S. 1988(/1993). "Evolution and Economic Change", in (ed.) A. Silberston, *Technology and Economic Progress*. London: Macmillan. This has been reprinted in U. Witt (ed.) *Evolutionary Economics*, Edward Elgar, 1993.
- Metcalf, J.S. 1998. *Evolutionary Economics and Creative Destruction*. London: Routledge.

- Meyer, G. R. 1989. *The Social Organization of the computer underground*. Masters Thesis, Northern Illinois University.
- Miles, I. 1999. ICT innovations in services. In H. Dutton (authors and edits) *Society on the line: Information politics in the digital age*. Oxford: Oxford University Press.
- Miles, I. 2000. "Services Innovation: Coming of Age in the Knowledge-Based Economy". *International Journal of Innovation Management* 4 (4): 417-454.
- Miller, D. 1987. *Material Culture and Mass Consumption*. Oxford: Blackwell.
- Mockus, A. & Fielding, R. T. & Herbsleb, J. D. 2002. "Two case studies of open source software development: Apache and Mozilla". Avaya Labs Research Technical Reports ALR-2002-003. URL (consulted on 25 September 2004) <http://www.research.avayalabs.com/techreportY.html>
- Moody, G. 2002. *Rebel code: Linux and the open source revolution*. New York: Perseus Books Group.
- Murray F. and Willmott, H. 1997. 'Putting information technology in its place', in B. P. Bloomfield, R. Coombs, D. Knights, and D. Littler (eds.), *Information Technology and Organisations: Strategies, Networks and Integration*, p. 160-180. Oxford: Oxford University Press.
- Nohara, H. & Verdier, E. 2001. "Sources of resilience in the computer and software industries in France". *Industry and Innovation* 8 (2): 201-220.
- Nyce, J. M. & Bader, G. 2002. *On foundational categories in software development, social thinking: software practice*. Cambridge, MA: MIT Press.
- Open Source Observatory. 2003. *LOSS Deployment in Extremadura, Spain*. Retrieved September 1, 2004, from <http://europa.eu.int/ida/en/document/1637/470> (For more case studies zooming in on certain Open Source adoptions in Member States in EU please see <http://europa.eu.int/ida/en/chapter/470>)
- Orona, C. J. 1997. *Temporality and identity loss due to Alzheimer's disease*. In *Grounded theory in practice*, edited by A. Strauss & J. Corbin. London: Sage publications.
- Panofsky, A. L. 2003. "From Epistemology to the Avant-garde: Marcel Duchamp and the Sociology of Knowledge in Resonance". *Theory, Culture & Society* 20(1).
- Peláez E. 1988. 'What Shapes Software Development?', Edinburgh PICT Working Paper NO. 10. Edinburgh: Research Centre for Social Sciences, University of Edinburgh.
- Perens, B. 1999. The open source definition. In C. DiBona, S. Ockman, and M. Stone (eds) *Open Sources: Voices from the Open Source Revolution*. Cambridge, MA: O'Reilly.
- Pickering, A. 1992. "From science as knowledge to science as practice", in A. Pickering (ed) *Science as practice and culture*. Chicago: The university of Chicago Press.

- Pinch T. & Wijker, W. E. 1989. "The Social Construction of Facts and Artifacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other", in W. E. Bijker, T. P. Hughes & T. Pinch (eds) *The Social Construction of Technological Systems*. Page 17-50.
- Raymond, E. S. (ed) 1996. *The new hacker's dictionary*. (3<sup>rd</sup> edition) Cambridge, MA: MIT Press.
- Raymond, E. S. 1999. *The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary*. Cambridge, MA: O'Reilly.
- Raymond, E. S. 2004. *The art of UNIX programming*. Reading, MA: Addison-Wesley.
- Reitman, W. 1964. Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems. In M. W. Shelley and G. L. Bryan, (eds.), *Human Judgment and Optimality*. New York: Wiley.
- Rheingold, H. 2000. *The Virtual Community: Homesteading on the Electronic Frontier*. Revised edition. The MIT Press.
- Ricci, C. 2003. The Economics of Open Source Software: Incentives, Coordination and Diffusion. PhD thesis. SSSUP.
- Rip, A. 2002. "Regional Innovation Systems and the Advent of Strategic Science". *Journal of Technology Transfer* 27: 123-131.
- Rip, A. 2003. "Technological innovation – in context". Background text for a keynote lecture at the meeting of the Lowlands Innovation Research Network, Louvain, 14 January 2003.
- Robertson, R. 1992/1994. *Globalization : Social Theory and Global Culture*. London: Sage.
- Robertson, R. 2001. "Globalization theory 2000?: major problematics". In G. Ritzer and B. Smart (Eds) *Handbook of Social Theory*. London: Sage.
- Robson, C. 1993. *Real World Research*. Oxford: Blackwell.
- Rogers, E. M. & Shoemaker, F. F. 1971. *Communication of innovations: a cross-cultural approach*. NY: The Free Press.
- Rosenberg, D. K. 2000. *Open source: the unauthorized white papers*. Hoboken, NJ: John Wiley & Sons.
- Ross, A. 1991. *Strange weather: culture, science and technology in the age of limits*. London: Verso.
- Rothwell R. 1992. "Successful industrial innovation: critical success factors for 1990s". *R&D Management* 22 (3): 221-239.
- Rothwell, R., Schott, K. and Gardiner, J.P. 1985. *Design and the economy: the role of design and innovation in the prosperity of industrial companies*, 3<sup>rd</sup> ed, Design Council.
- Royce, W. W. 1970. "Managing the Development of Large Software Systems". Proceedings of IEEE WESCON.
- Sachs, P. 1995. "Transforming work: Collaboration, learning and design".

*Communications of the ACM*, 38(9), 36-44.

- Sahal, D. 1981. *Patterns of Technological Innovation*. Reading, MA: Addison-Wesley.
- Savage, A. 1997. "In Tent on Mischief", *Guardian* 14 August 1997. URL [http://www.savage.net/public\\_html/writing/guardian.html](http://www.savage.net/public_html/writing/guardian.html) (consulted on 28 July 2004).
- Schumpeter, J. 1939. *Business cycles: a theoretical, historical, and statistical analysis of the capitalist process*. New York: McGraw-Hill.
- Sen, A. 1999. *Development as Freedom*. Oxford: Oxford University Press.
- Sewell, W. H. 1992. A theory of structure: duality, agency, and transformation. *American Journal of Sociology*, 98, p. 1-29.
- Shah, S. K. 2003. *Community-Based Innovation & Product Development: Findings From Open Source Software And Consumer Sporting Goods*. PhD Dissertation. Management. Cambridge, MA: Massachusetts Institute of Technology: 230.
- Shapiro, C. & Varian, H. R. (eds) 1998. *Information Rules: A Strategic Guide to the Networked Economy*. Boston: Harvard Business School Press.
- Sharp, 1997. in M. Talalay, R. Tooze & C. Farrands (Eds.) *Technology, Culture and Competitiveness: Change and the World Political Economy*. London: Routledge.
- Shearman, C. 1997. in M. Talalay, R. Tooze & C. Farrands (Eds.) *Technology, Culture and Competitiveness: Change and the World Political Economy*. London: Routledge.
- Shibutani, T. 1955. "Reference Groups as Perspectives". *American Journal of Sociology* 60: 562-69.
- Shibutani, T. 1961. *Society and Personality*. Englewood Cliffs, N.J.: Prentice-Hall.
- Shy, O. 1998. "The Economics of Software Copy Protection and Other Media," In Deborah Hurley, Brian Kahin, and Hal Varian (eds) *Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property*, Cambridge, MA: MIT Press.
- Silverman D. 1993. *Interpreting qualitative data: methods for analysing talk, text and interaction*. London: Sage.
- Silverman, D. (Ed.) 1997. *Qualitative Research: Theory, Method and Practice*. London: Sage.
- Silvonen, J. 2002. "Linux User Groups and the 'Linux Community'", a paper presented at the second Oekonux conference, Technischen Universität Berlin, 1-3 November 2002. URL (consulted on 5 June 2004): [http://home.edu.helsinki.fi/~jsilvone/Oekonux/OekoN03\\_JS.html](http://home.edu.helsinki.fi/~jsilvone/Oekonux/OekoN03_JS.html).
- Simon, H. 1973. The Structures of Ill Structured Problems. *Artificial Intelligence*, 4, 181-201.
- Skibell, R. 2002. The myth of the computer hacker. *Information, Communication & Society*, 5(3), 336-356.

- Slater, D. 2000. *The Internet: An Ethnographic Approach*. Oxford: Berg.
- Sommerville I. 2001. *Software engineering*. 6<sup>th</sup> edition. Essex: Pearson education Ltd.
- Stallman, R. "The Emacs Full-Screen Editor". URL (consulted 27 November 2003): <http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt>
- Stallman, R. 1995. *GNU Emacs Manual*. 11<sup>th</sup> Edition, Version 19.29. Boston: Free Software Foundation.
- Stallman, R. 1998. 'EMACS: The Extensible, Customizable Display Editor', 11 February, URL (consulted November 2003): <http://www.gnu.org/software/emacs/emacs-paper.html>
- Stallman, R. 2002. My Lisp Experiences and the Development of GNU Emacs (transcript of Richard Stallman's Speech, 28 Oct 2002, at the International Lisp Conference). URL (consulted December 2003): <http://www.gnu.org/gnu/rms-lisp.html>
- Star, S. L. & Ruhleder, K. 1996. 'Steps toward an ecology of infrastructure: design and access for large information spaces', *Information Systems Research*, March 1996, 7(1): 111-134.
- Star, S. L. 1983. Simplification in scientific work: An example from neuroscience research. *Social Studies of Science* 13: 208-26.
- Star, S. L. 1991. Power, technology and the phenomenology of conventions: on being allergic to onions, in *A sociology of Monsters: essays on power, technology and domination*, edited by John Law. London: Routledge.
- Star, S. L. 1995. Introduction, in 'The cultures of computing', edited by Susan Leigh Star. Oxford: Blackwell.
- Star, S. L. and Griesemer, J. R. 1989. "Institutional Ecology, 'Translations,' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907 - 1939." *Social Studies of Science* 19: 387-420.
- 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 the book *Digital library use: social practice in design and evaluation*, edited by Ann Peterson Bishop, Nancy A. Van House, and Barbara P. Buittenfield. The MIT press.
- Stehr, N. 2001. *The Fragility of Modern Societies: Knowledge and Risk in the Information Age*. London: Sage publications.
- Strauss, A. 1978. A social world perspective. *Studies in Symbolic Interaction* 1: 119-128.
- Suchman, L. 2000. Located Accountabilities in Technology Production. A work-in-progress, undergoing revision from an earlier paper titled "Working Relations of Technology Production and Use", published in the journal *Computer-Supported Cooperative Work (CSCW)* 2: 21-39, 1994 and, in abbreviated form, in Mackenzie, D. and Wajcman, J. (Eds.) *The Social Shaping of Technology*, Second Edition. Buckingham, UK: Open University Press, pp. 258-265, 1999. This version was presented at the Sawyer Seminar on Heterarchies, Santa Fe Institute, October 2000.

- Suchman, L. 2001. 'Human/Machine reconsidered', a work-in-progress paper under development as the introduction to a 2nd, revised edition of *Plans and Situated Actions: the problem of human-machine communication*. Cambridge University Press, originally published in 1987.  
<http://www.comp.lancs.ac.uk/sociology/soc0401s.html>
- Talalay, M., Tooze R. & Farrands, C. 1997. "Technology, culture and competitiveness: Change and the world political economy", in M. Talalay, R. Tooze & C. Farrands (Eds.) *Technology, Culture and Competitiveness: Change and the World Political Economy*. London: Routledge.
- Taylor, P. A. 1993. *Hackers: A Case-study of the Social Shaping of Computing*, Ph.D. dissertation, University of Edinburgh.
- Taylor, P. A. 1999. *Hackers: crime in the digital sublime*. London: Routledge.
- The Economist, July 25<sup>th</sup> 2002, "Going Hybrid". URL retrieved on 9<sup>th</sup> September 2004  
[http://www.economist.com/printedition/displayStory.cfm?Story\\_ID=1251254](http://www.economist.com/printedition/displayStory.cfm?Story_ID=1251254)
- Thomas, D. 2002. *Hacker Culture*. University of Minnesota Press,
- Thomas, G. & Wyatt, S. 2000. "Access is not the only problem: using and controlling the Internet". In S. Wyatt & F. Henwood & N. Miller & P. Senker (eds.) *Technology and In/Equality: Questioning the Information Society*. London: Routledge.
- Tönnies, F. 1887. *Community and society (Gemeinschaft und Gesellschaft)*. (Trans. and Ed. Charles P. Loomis). In DeFleur and Ball-Rokeach *Theories of mass communication*. East Lansing, MI: Michigan State University Press.
- Turkle, S. 1984. *The Second Self: Computers and the Human Spirit*. NY: Simon & Schuster,
- Turkle, S. 1995. *Life on the screen- Identity in the Age of the Internet*. New York: Simon and Shuster.
- 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/tad1967.doc.htm>
- Välimäki, M. 2003. "Dual Licensing in Open Source Software Industry". *Systemes d'Information et Management* 1/2003. URL (consulted on 080504):  
<http://opensource.mit.edu/papers/valimaki.pdf>
- Vandenberghe F. 2002. "Reconstructing Humans: A Humanist Critique of Actant-Network Theory". *Theory, Culture & Society*. Vol. 19: 5/6, pp. 51-67(17)
- Voland, G. & M. 1999. *Engineering by design*. Reading, MA: Addison-Wesley.
- Von Hippel, E. 1988. *The sources of innovation*. Oxford: Oxford University Press.
- Wærn, Y. & Mattus, M. & Sundén, J. & Sveningsson, M. 2001. *Networks: Methodologies for research on Internet communication Plans for research 2000-2003*. <http://www.brunel.ac.uk/depts/cric/vm>
- Wanless, D. 2002. *Securing Our Future Health*, Department of Health, London:

HMSO.

- Washington Post. 3 November 2002. Europe's Microsoft Alternative: Region in Spain Abandons Windows, Embraces Linux.
- Waterton, C. 2003. "Amateurs as Experts: An Ethnography of Participation and Co-Production". Paper presented at ESRU, March 2003, Geography Department, UCL, London University.
- Weber, M. 1922 (1980). *Wirtschaft und Gesellschaft: Grundriss der verstehenden Soziologie*. Tübingen: J.C.B. Mohr (Paul Siebeck).
- Webster, F. 1995. *Theories of the Information Society*. London: Routledge.
- Webster, J. 1996. *Shaping Women's Work: Gender, Employment and Information Technology*. London: Longman.
- Weeks, J. 1989. Uses and abuses of Michel Foucault. In L. Appignanesi (Ed.), *Ideas from France: The legacy of French theory*. London: Free Association Books.
- Wellman, B. (ed.) (1999). *Networks in the Global Village*. Boulder, CO: Westview Press.
- Wenger, E. 1998. *Communities of Practice - Learning, Meaning, and Identity*. New York: Cambridge University Press.
- Wheeler, D. A. 2004. "Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!" (June 8, 2004) URL (consulted on 28 September 2004) [http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html)
- Williams, S. 2002. Free as in freedom: Richard Stallman's crusade for free software. O'Reilly. URL (consulted November 2003): <http://www.oreill.com/openbook/freedom/index.html>
- Winter, S. 1987. "Knowledge and competence as strategic assets", D. Teece (ed.), *The Competitive Challenge: Strategies for Industrial Innovation and Renewal*, Cambridge, Mass.: Ballinger, 159-83.
- Wired News. August 5, 1999. "Chaos in Berlin" URL (consulted 28 September 2004) <http://www.wired.com/news/culture/0,1284,21104,00.html>
- Wired News. August 9, 1999. "Geeks in Tents = Chaos" URL (consulted 28 September 2004) <http://www.wired.com/news/culture/0,1284,21174,00.html>
- Witt, U. 2002. 'How Evolutionary is Schumpeter's Theory of Economic Development?'. *Industry and Innovation*, 9(1/2): 7-22.
- Wood, N. 1985. Foucault on the history of sexuality: An introduction. In V. Beechey & J. Donald (Eds.), *Subjectivity and social relations*. London: Open University Press.
- Woods, B. and Watson, N. 2003. 'Power to Independence,' Inaugural Conference of the Disability Studies Association. Disability Studies: Theory, Policy and Practice, Lancaster University, 4 September - 6 September 2003.
- Yin, R. 1994. *Case Study Research: Design and Method*, 2<sup>nd</sup> edition. London: Sage.

Young, J. 1971. *The drugtakers*. London: Paladin.

Young J. 1973. 'The amplification of drug use', in S. Cohen and J. Young (eds.) *The Manufacture of News: social problems, deviance and the mass media*. London: Constable.

ZDNet. August 2, 2002. "Security czar points fingers of blame". URL (consulted August 30 2002) <http://zdnet.com.com/2100-1105-947409.html>

- i EMACSen, the plural form of EMACS, means a cluster of EMACS-based software.
- ii URL (consulted on 27 September 2004) <http://www.abet.org/>
- iii A software development process is a process used to develop computer software. It may be an ad hoc process, devised by the team for one project, but the term often refers to a standardised, documented methodology which has been used before on similar projects or one which is used habitually within an organisation.
- iv The phases a software product goes through between when it is conceived and when it is no longer available for use. The software life-cycle typically includes the following: requirements analysis, design, construction, testing (validation), installation, operation, maintenance, and retirement.
- v URL <http://www.extremeprogramming.org> (consulted on 30 July 2004).
- vi Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them (Hewett *et al.* ACM SIGCHI 1992, 1996, p. 6).
- vii The word 'license' is spelled in American English because it is an American branded product. It's the same for various licences below appearing with their full names.
- viii Software library or program library is a collection of software held either permanently accessible on backing store or on removable media such as tape or disk. It will include complete software packages, package modules which will only be required occasionally, and machine-code routines for loading into user programs (BIS 1995:280)
- ix Bash is the shell, or command language interpreter, that will appear in the GNU operating system. Bash is an sh-compatible shell that incorporates useful features from the Korn shell (ksh) and C shell (csh). It is intended to conform to the IEEE POSIX P1003.2/ISO 9945.2 Shell and Tools standard. It offers functional improvements over sh for both programming and interactive use. In addition, most sh scripts can be run by Bash without modification (<http://www.gnu.org/software/bash/bash.html>).
- x Kernel is the essential part of Unix or other operating systems, responsible for resource allocation, low-level hardware interfaces, security etc. (Free On-Line Dictionary Of Computing, FOLDOC, <http://wombat.doc.ic.ac.uk/foldoc/>)
- xi The full text of the GPL is available at <http://www.gnu.org/copyleft/gpl.html>
- xii The Open Source Definition (version 1.7) is available at <http://www.opensource.org/docs/definition.php>
- xiii The Jargon File version 4.4.7 is available at <http://catb.org/~esr/jargon/>
- xiv In computing, a patch is a software update meant to fix problems, bugs or the usability of a previous version of an application. Patching can be done to either the binary executable or a programme's source code ([http://en.wikipedia.org/wiki/Patch\\_%28computing%29](http://en.wikipedia.org/wiki/Patch_%28computing%29), consulted on 30 July 2004).
- xv Actually in a postmodern society, a firm definition of something is hardly made. Likewise, a 'protector' will be pointed as a 'terrorist' because his/her behaviours complying with the standard practices set in the governmental anti-terrorism act. Terms such as 'terrorists' or 'hackers' are 'labels' created to classify some people doing certain things. And if the term is borrowed by the media to introduce to the public, the problem of definition will arise because the term has been reduced to a convenient 'catch-all' level used to describe a range of disparate collective practices.
- xvi An abridged questionnaire has been disseminated onto mailing list of Belgian-based Free Software and UK-based discussion groups [comp.os.linux](mailto:comp.os.linux), [comp.programming](mailto:comp.programming), [comp.os.linux.networking](mailto:comp.os.linux.networking).
- xvii URL <http://www.london2600.org.uk/> (consulted on 28 July 2004).

- xviii This historical categorisation is according to the distinction made by Levy (1984) between the old-school hackers, who supported an ethic of ‘free access to technology’ and a spirit of free and open exchange of information in the 1960s and 1970s, and the new-school hackers, who demonstrate their belief in free access to information by exploiting software vulnerabilities from the 1980s on.
- xix European hackfest has been initiated in 1989 by the Galactic Hacker Party where 200 participants turned up. But this event was held indoors. URL <http://www.hacktic.nl/magazine/2025.htm> (consulted on 28 July 2004).
- xx URL <http://www.hip97.nl/> (consulted on 28 July 2004)
- xxi URL <http://www.hal2001.org/> (consulted on 28 July 2004).
- xxii A participant at LinuxTag 2002 travelling from the US has mentioned that the Linux conference in the US is very different from the European one. The Linux exhibition in the US is said to be too much commercial with flyers and noise all over the place.
- xxiii Unshielded Twisted Pair (UTP) is a standard form of wire cable used to provide the connections in a network. It is commonly used for data transmission.
- xxiv However, given the disappointing weather during the 3 days, most programmes were held indoors.
- xxv On the webpage of CCC 2003, it is written that ‘This is the Camp Rocket in PNG format. Probably not viewable in IE6 because Microsoft sucks. Get a real browser and do not waste your time looking at the world through dirty glasses.’
- xxvi Single-sign-on (SSO) means users log in one system and then could access to a lot of systems using the same authentication database. Passport from Microsoft is a SSO in that when you log in a Passport website, you don't need to retype your credentials if you connect to another Passport website.
- xxvii This proposal to allow copyright holders to attack computers on P2P networks used for piratical purposes, however, was not accepted in the Congress. But RIAA has won a court decision upholding its right to use the subpoenas, which take advantage of a controversial fast-track provision that allows copyright holders to obtain information about alleged infringers without first filing a lawsuit. It is written that, RIAA has filed close to 1000 subpoenas in the US District Court in Washington this in a month (Cnet News 22 July 2003). Some of the subpoena were sent to innocent users because RIAA's automated programme apparently confused two separate pieces of information—a legal MP3 file and a directory named “usher”—and concluded there was an illegal copy of a song by the musician Usher. RIAA's action is seriously criticised in that the process is hardly privacy-protective, and it allows copyright holders to learn the identity of an Internet user without filing a lawsuit or obtaining a judge's approval. RIAA's anytipiracy campaign continues. (cf. Cnet News.com ‘Subpoena's Sour Note’ 1 August 2003).
- xxviii In February 2002, Kazaa BV sold the Kazaa file-swapping software to Sharman Networks, a company based in Vanuatu, a small island in the South Pacific. The copyright lawsuit filed by RIAA and MPAA has been ruled to include Sharman Networks, which distributes the Kazaa software in June 2002.
- xxix Nimda is one of the most destructive Internet worms of 2001.
- xxx See the explanation of my little experiment in the methodology chapter.
- xxxi The story is about how ancient Taiwanese put pebbles on the ground in a pyramid shape and remove them following certain rule to perform calculation. The answer was always correct. No one knew how did this method came up so far. It is a legend.
- xxxii A display editor indicates ‘an editor in which the text being edited is normally visible on the

- screen and is updated automatically as the user types his commands. No explicit commands to “print” text are needed’ (Stallman 1998: 2).
- xxxiii Note that these narratives are quoted from the documentation written in 1998, almost 20 years after the original release of EMACS. This is to say that the apparatuses Stallman mentioned here are not developed in a day. Instead, their developments are shaped by Stallman’s interactions with users and users themselves, as well as the given material environment.
- xxxiv See the coverage on a comparison of the new Microsoft and Apple operating systems in *THE EYE* weekly coming with the *Saturday Times* on 8-14 November 2003, page 24-25.
- xxxv URL <http://agilemanifesto.org/> (consulted on 30 July 2004)
- xxxvi Here ‘commercial software’ is generally referred to the one produced to put on market mainly for profits. Most proprietary software is commercial software.  
<http://www.gnu.org/philosophy/categories.html>
- xxxvii The most collective open source practices include: 1) the right to make copies of the program, and distribute those copies (free redistribution); 2) the right to have access to the software’s source code (source code); 3) the right to make improvements to the program (derived works). (cf. Open Source Definition).
- xxxviii Nonetheless, 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 deductible in the United States.
- xxxix [http://www.debian.org/social\\_contract.html](http://www.debian.org/social_contract.html)
- xl At present, there are 90 people subscribed to the mailing list, although this figure doesn't take into account people who might be subscribed twice (eg. work and home addresses)
- xli Ubuntu Linux URL: <http://www.ubuntulinux.org>
- xlii GNOME project URL: <http://www.gnome.org>
- xliii **XFree86®** is a freely redistributable open-source implementation of the X Window System.  
URL: <http://www.xfree86.org>
- xliv Oekonux Project URL: <http://www.oekonux.org>
- xlv Some Linux developers/users based in Italy (and elsewhere) suggest to have Linux translated in local dialects. In so doing, not only the technical problem of localisation and internationalisation is solved, but also does this practice encourage local people participate in the FLOSS development. Three screenshots featuring the initial booting screen, the start-up progress status. and the final desktop environment taken from an Italian locally customised Linux in Macerata are presented in Appendix 4 (<http://lists.linux.it/pipermail/annunci/2004-August/000246.html>).
- xlvi URL <http://www.eriders.net/>
- xlvii URL <http://www.lincproject.org>
- xlviii URL <http://www.tacticaltech.org/summersource>
- xliv URL <http://www.tacticaltech.org/africasource>
- l <http://www.oreilynet.com/et2004/>