The Samba Project: Transformation of Self through Open Source Software Development

An honours thesis submitted in accordance to the requirements of the Bachelor of Arts Informatics (Honours) degree at The University of Sydney, Australia

Nicolaas Charles Earnshaw

November 16th, 2004

Supervision: Christine Crowe

Acknowledgements

The older I become
The more I realise
What work I do is incidental
Compared to the memory
Of those I encounter along the way

For the grace and glory of God.

To my supervisor Christine Crowe, thank you for the assistance, enthusiasm and perseverance. This thesis would not have been possible without your tireless efforts. You are the finest example of what a supervisor should and can be.

This dissertation would not be possible without the co-operation of the Samba Team. I thank those team members interviewed for their invaluable contributions.

Acknowledgement and gratitude to Andrea Stern who first placed this topic under my nose, thank you for your guidance and belief in me throughout my time at university. You saw things I could not and the completion of my studies are in large part because of your qualities as a teacher and colleague.

For the early advice and assistance of Joseph Davis in defining my research topic, I am grateful. I also wish to thank the School of Information Technology at the University of Sydney for the teaching and learning resources made available. Thanks to the School of Sociology and Social Policy at the University of Sydney for the recording equipment.

To my family, partner and friends, thank you for your love and support.

Thank you to the fellow residents of LG-41, I wish you all the best in coming years and have no doubt of your imitate success in your chosen fields of endeavour. I was blessed to be surrounded by such good natured, humorous and supportive fellow students. I will miss you all. A special mention to Jesse for the additional recording equipment.

Table of Contents

Index of Figures / Tables

Figures

Tables

Acronym List – Alphabetical Order

AI	Artificial Intelligence
BSD	Berkeley Software Distribution (BSD)
CS	Computer Science
CVS	Concurrent Versions System
FLOSS	Free Libre / Open Source Software
FS	Free Software
GCC	GNU Compiler Collection
GIMP	The GNU Image Manipulation Program
GNOME	The GNU Network Object Model Environment
GNU	Is a recursive acronym for "GNU's Not Unix"
GPL	General Public License
ICT	Information and Communication Technology
IS	Information Systems
IT	Information Technology
LPP	Legitimate Peripheral Participation
MIT	Massachusetts Institute of Technology
OS	Open Source
OSS	Open Source Software

Introduction: Section I

Open Source (OS) and Free Software (FS) development communities have been a long standing phenomenon (Wayner 2000), emerging with the advent of early computing technology (Rosenzweig 1998); however, the theorizing of these communities is underdeveloped in the discipline of Sociology.

There have been however numerous studies on OS from a wide variety of academic fields including computer science (Fitzgerald and Feller 2001), economics (Lerner and Tirole 2002; Mustonen 2003), marketing (Hemetsberger 2003), information systems (Lee and Davis 2003), anthropology (Zeitlyn 2003), knowledge management (Hemetsberger and Reinhardt 2004), organisational studies (Gallivan 2001; Franck and Jungwirth 2002) and law (Benkler 2002). There have also been works published by OS developers themselves (Stallman 1999; Raymond 2001; Torvalds 2001) and histories of OS aimed at the public readership (Wayner 2000). In academic literature, research on OS software development is dominated by both an economic and psychological understanding of the phenomena, which articulate the issues in terms of individual motivations. As Feller and Fitzgerald (2002) state, investigations of OS communities need contributions from a number of disciplines. This thesis examines the reasons for members' ongoing participation in a particular OS community, the Samba project, from a sociological perspective. Is so doing, the research question is articulated in terms how participants' identity as an OS community member is constituted in the context of the structure and interrelations within that community, rather then in terms of individual motivations.

The Research Question in Context

Existing research in OS has focused on why developers begin and sustain participation in such projects, particularly considering the lack of direct financial compensation offered for such tasks. Most empirical research, from both economic and psychological perspectives, has articulated the question in terms individual motivation focusing on two broad areas; 'internal factors' (intrinsic motivations) and 'external factors' (external reward) (Hars and Ou 2002; Lerner and Tirole 2002;

Lakhani and Wolf 2003). This literature reveals a number of motivational factors and sources. Table one created below encapsulates this spectrum.

Motivational Area	Motivational Factor	Reference
Internal Factors / Intrinsic Motivation	Hedonistic	(Raymond 2001; Hars and Ou 2002; Lakhani, Wolf et al. 2002; Hertel, Niedner et al. 2003; Lakhani and Wolf 2003a)
	Political	(Stallman 1984; Raymond 2001; Hertel, Niedner et al. 2003)
	Altruism	(Hars and Ou 2002)
	Community Identity	(Hars and Ou 2002; Hertel, Niedner et al. 2003; Zeitlyn 2003)
	Observance of norms	(Osterloh, Rota et al. 2001; Ghosh and Glott 2002; Hemetsberger 2003; Hertel, Niedner et al. 2003; Ye and Kishida 2003; Zeitlyn 2003)
	Learning	(Hemetsberger 2001; Ghosh and Glott 2002; Lakhani, Wolf et al. 2002; Ye and Kishida 2003; Lakhani and Wolf 2003a)
External Factors / External Reward	Career concern / self-marketing	(Hars and Ou 2002; Lerner and Tirole 2002; Hertel, Niedner et al. 2003; Veale 2003)
	Peer recognition / ego gratification / reputation	(Ghosh 1998; Kollock 1999; Hemetsberger 2001; Raymond 2001; Hars and Ou 2002; Lerner and Tirole 2002; Hemetsberger 2003)
	Low cost / high returns of opportunity	(Ghosh 1998; Lerner and Tirole 2002) (Kollock 1999)
	Role Transformation	(Ye and Kishida 2003)
	Product need	(Lakhani, Wolf et al. 2002; Lerner and Tirole 2002; Hertel, Niedner et al. 2003)

Table 1 Taxonomy of individual motivations as discussed in the literature

The main limitation of this analytical framework is that motivational forces are seen to be working on the self, that is to say the developers are seen as driven by needs that do not change as they become engaged in a community. For example Hemetsberger

(2003) states that their "results suggest that the motivational basis for high behavioural involvement derives from the correspondence of immediate benefits and future reputation incentives" (Hemetsberger 2003:27). Lerner and Tirole (2002) similarly state that a "programmer participates in a project, whether commercial or open source, only if she derives a net benefit (broadly defined) from engaging in the active" and these come in the form of the immediate benefits of hedonic and use-value or the delayed benefits of future job offers and reputation incentives (2002:213). Ye and Kishida (2003) attempt to argue that "learning is one of the driving forces that motivate developers to get involved in OS [software] projects is because it provides intrinsic satisfaction for OS software developers and the role transformation in OS software communities that go along with learning offers the extrinsic motivation" (2003:425). This role transformation seems to have no impact on the motivation of learn, it continues to the main source of motivation. Hertel, Niedner et al. (2003) recognise that personal identification is a significant factor for investigation. Nevertheless their paper again discusses personal identification as a motivational factor, rather than explaining how such identification comes about. In other words it does not examine the process through which identity is constituted (Hertel, Niedner et al. 2003:1159).

This thesis focuses on the process of change in how the individual developers come to identify as members of the Samba community. This thesis shows that the structure and interactions of developers form the context within which individuals constitute their identity from an individual seeking to satisfy a personal need, in terms of a problem with a particular piece of OS software, to that of a community member.

Theoretical and Analytic Tools

The term community has multiple meanings within the discipline of Sociology. Benedict Anderson's 'Imagined Communities' (1983), whilst principally offering an interpretation of 'the anomaly of nationalism' (1983:13) illustrates the way that community membership constitutes identity to one's self and to others. Anderson highlights that a nation is 'imagined' because "the members of even the smallest nation will never know most of their follow-members, meet them, or even hear all of them, yet in the minds of each lives the image of their communion" (1983:15). Community, then, is constructed around similarities rather then being tied to a space or place. Community exists primarily as something imagined because it is impossible to know all of those who are considered of the nation. This position, that community is not geographical bound but imagined, is pertinent in that information and communication technologies have superseded the need for 'geographical boundaries' in what constitutes a community. Delanty (2000:127) represented community as a 'shared cultural imaginary'. Following writers like Maffesoli (1996), and Benedict Anderson (1983), he stressed the ability of people, at least in liberal democratic societies, to imagine themselves as part of a deterritorialized and globalized community (Olssen 2002:485). That is as the bonds of traditions diminish in those societies the need for trust, solidarity, and autonomy, the core concepts of community, are located in a new type of community.

Zygmunt Bauman's (2000) contends that contemporary society is in state of 'liquid modernity', in which all aspects of the human condition, from community to paid work, become continuously and irreparably *fluid* (Bauman 2000:79).This in turn creates increased transience, uncertainty and insecurity for the individual. As he says inter-human bonds are brittle, breakable and have "*ad hoc* modality" (Bauman 2004:19). In such a society "[b]onds are easily entered but even easier to abandon…long-term commitments with no option of termination on demand are decidedly out of fashion *and not what a 'rational chooser' would choose*" (Bauman 2004:20). Moreover, within such a social order "[c]ommunity feels good because of the meanings the word 'community' conveys - all of them promising pleasures, and more often than not the kinds of pleasures we would like to experience but seem to

miss" (Bauman 2001:1). 'Community' is a safe place within which we can rely on of the good will of others; it is an expression of a yearning for a "world which is not, regrettably, available to us - but which we would dearly wish to inhabit and which we hope to repossess" (Bauman 2001:3). The transience of contemporary life can never be ameliorated by belonging to a community as contemporary communities are merely "zipped harnesses…and their selling point is the facility with which they can be put on in the morning and taken off in the evening" (Bauman 2001:169).

Bauman suggests that the loss of 'traditional' community, which involves long-term commitment and stable identity, has manifested in a fundamental loss of security:

> We miss community because we miss security, a quality crucial to a happy life, but one which the world we inhabit is ever less able to offer and ever more reluctant to promise. But community remains stubbornly missing, eludes our grasp or keeps falling apart, because the way in which this world prompts us to go about fulfilling our dreams of a secure life does not bring us closer to their fulfilment; instead of being mitigated, our insecurity grows as we go, and so we go on dreaming, trying and failing. (Bauman 2001:144)

Because community no longer provides a stable point of reference or a site for trust, certainty and security, the individual is forced to assume responsibility for their life, for their own 'redemption and doom' (Bauman 2000:62). People shop around for identities (Bauman 2000:83) yet remain on alert to guard their own flexibility to ensure they can move with the swiftly changing patterns of the world 'out there' (Bauman 2000:85). In other words, rather than a stable identity being secured and affirmed in relation to a community, the individual is compelled to constitute security as their individual life project. The quest for community becomes a relentless exercise of 'solitary identity-building' (Bauman 2000:16) in which identity becomes a poor surrogate for community. In this sense, 'Identity sprouts on the graveyard of communities' (Bauman 2001:16). Although a graveyard, the illusion of achieving security within community persists. The realisation of this dream will never be fulfilled since individuals experience a tension between the longing for security within a community and the curtailment of freedom to express their individual identity, or their difference, within that community (Bauman 1998):"The really existing community will be unlike their dreams – more like their opposite: it will add to their

fears and insecurity instead of quashing them or putting them to rest" (Bauman 2001:17).

Although Bauman's work may be useful to gain an understanding of the broad and general changes in western society, in terms of the decline of traditions to secure identity and the increasing ephemeral nature of human relations, his metatheoretical perspective is limited in that he does not consolidate his arguments using research on communities in relation to identity. This thesis presents an analysis of one community to investigate whether or not members of the Samba community are seeking identity through participation in such a community. In so doing, it is necessary to delineate how Samba, as a community, will be conceptualised and analysed.

Samba: A Community of Practice or Epistemic Community?

OS communities have been conceptualised as both 'communities of practice' (Scacchi 2002; Elliott and Scacchi 2003:24; Ye and Kishida 2003) and 'epistemic communities' (Edwards 2001). This section will outline the analytical constructs for understanding such communities in order to provide a frame of reference for the analysis of the Samba community. Communities of practice are "groups of people who share an interest in a domain of human endeavour and engage in a process of collective learning that creates bonds between them"(Wenger 2001:2339). It is certainly possible to view OS projects as being communities of practice centred on the software development domain (Hemetsberger and Reinhardt 2004), with the possibility of collective learning seen by some authors to be a key motivator for participation (Ye and Kishida 2003). According to Wenger (1998), communities of practice have three main characteristics: a member must have a minimum level of knowledge of the area; a community exists wherein members interact and learn together, and there must be a culture in which members "develop a shared repertoire of resources: experiences, stories, tools, [and] ways of addressing recurring problems – in short a shared practice. This takes interactions over time" (Wenger 2001:2340).

Peter Haas (1992) notes that epistemic communities can be defined in a variety of ways (1992:3) and has been used more often in relation to scientific communities as 'expert' communities which are called upon by governments to advise on policy. This

thesis adopts Cinquegrani, Haas and Edwards (Haas 1992:3; Edwards 2001:8; Cinquegrani 2002:779-780) definition, which proposes that epistemic communities are networks consisting of participates with varying experience and competence in a particular domain. Epistemic communities have four key elements:

> 1) A shared set of normative and principled beliefs, which provide a value-based rationale for the social action of community members.
>
> 2) Shared causal beliefs, which are derived from their analysis of practices leading or contributing to a central set of problems in their domain.
>
> 3) Shared notions of validity – that is, inter-subjective, internally defined criteria for weighing and validating knowledge in the domain of their expertise.
>
> 4) A set of common practices associated with a set of problems to which their professional competence is directed, know as a common policy enterprise (Haas 1992; Edwards 2001:8-9).

As Edwards (2001) points out, although OS communities are not called upon to assist with government policy decisions, in all other respects this framework is eminently suitable for the analysis of OS communities. As the thesis will show, there are aspects of the Samba community which conform to both of these frameworks. At the same time, however, the research will show that the Samba community differs in some ways, particularly in relation to the issue of identity.

The remaining section of this chapter places the research question and method within wider debates regarding acceptable knowledge in information systems. A case of a qualitative research method is given along with a detailed explanation of the steps taken to implement such a research method.

Chapter outline

Chapter One will introduce the concept of Open Source software and give a brief history of its inception and development, with attention to the debates surrounding the political implications of the initials premises of the OS software movement. Particular attention will be given to the role that licenses play in defining the community. The Samba project is then defined a both a software product and community. The chapter then introduces observations from Samba developers in relation to how they initially joined the Samba community. It is suggested the community environment is a significant factor in understanding participants' continuing involvement with the Samba project.

Chapter Two places these meanings with the community's informal and formal structures. A particular working of social capital is used to obtain a detailed understanding of how the Samba project can enrol others into a collaborative initiative. This suggests a reworking of notions of hierarchy and merit within the community. Finally code control structures are shown to be related the structuring of relationships between developers and users alike.

Chapter Three explores the value of sharing within the Samba community. By using a number of examples, from its codification in licensing schemes to indicators that the Samba community is very successful at socialising members. This chapter explores situations where that norm would seem to be challenged; namely within the act of forking. It is shown that the strength and importance of the social bonds formed in practicing OS developer limits the chance of such events regularly occurring.

Chapter Four addresses Samba developers' perceptions and responses to broader transformations within the OS community in general; particularly the changes which have taken place since the original inception of the Free Software movement. This relaters to the second issue, that of the introduction of industry sponsorship. Has this new development altered how software developers perceive their practices?

Chapter Five examines the relationship between developers' participation in the Samba community and their constitution of identity. By doing so it answers the question of how one's identification with both the Samba community as well as the product, both of which are social processes, explains a developers' ongoing participation in the project. In doing so this chapter also explores the evidence for personal bonds that extend beyond the practice of Samba development. This challenges some of the assumptions presented in a communities of practice understanding of OS projects. This is highlighted within the rarely discussed issue of how and why programmers leave such communities.

Introduction: Section II

Methodological Considerations

The dominant form of empirical work in this field, except the work of Raymond (2001) and Kollock (1999), has primarily been the use of the survey research methodology and the questionnaire method (Hertel, Niedner et. al. 2003; Hars and Ou 2002; Ghosh and Prakash 2000; Ghosh and Glott 2002; Lakhani, Wolf et al. 2002; Lakhani and Wolf 2003). Theses papers have focused on testing established theories from earlier papers (Lerner and Tirole 2002; Raymond 2001; Ghosh 1998; Veale 2003; Zeitlyn 2003) on the issue of developers' motivations for participation in OS projects. In obtaining primary data the preferred method has been to use closed questionnaires. This has lead to the application of existing research regarding motivations, ranging from economic (employability, skill development) to psychological issues. As illustrated in Table one (page 8), developers' have thus been asked to respond to closed questions based upon existing theories of why they participate in such projects.

Most of the research has been conducted within the discipline of Information Systems or Computer Science, and has used the survey research methodology and questionnaire method. As Orlikowski and Baroudi (1991) point out, "a positivist research perspective is dominant in information systems research" (1991:4), and dominated by survey and laboratory experiments with cross-sectional single snapshot time periods used (Orlikowski and Baroudi 1991:4-5). The authors assess the positivist research philosophy in relation to Information Systems (IS), and contest the assumption that "people are *not* active makers of their physical and social reality" (Orlikowski and Baroudi 1991:12 emphasis mine).

Implications of the Dominant Perspective and Methodology: An Example

Hars and Ou's (2002) *Working For Free? Motivations for Participating in Open-Source Projects* is an example of how theoretical perspective of positivism informs methodological preference and in so doing curtails the ability to investigate in depth

the reasons for members' ongoing participation in OS communities. Hars and Ou (2002) wish to examine the developers motivations to contribute, which ranged from questions concerning altruism to career incentives. As Hars and Ou state, "[t]o better understand participation in open-source projects, it is necessary to have firsthand information from actual programmers"(2002:31).

Considering that the researchers explicitly wished to understand developers' motivates and stated that such information needs to come from the developers themselves, their decision to apply existing theories of human motivation and asked developers to choose or rank them using closed question surveys, restricts the possibility of obtaining novel insights which may come from developers themselves. The limits of such closed question surveys for this type of research is well documented (Newell 1996; May 1998:110-111). Of particular concern here is the lack of credence given to the developers themselves in defining the terms of reference with which to discuss the issue of motivation. Indeed, to understand the motives of developers it is necessary to engage with them in a way that allows them to express their relationship to the phenomena, in other words within their own frames of reference. This would lead to a greater understanding of the subjects' point of view (May 1998:112). In using the survey methodology in terms of closed questions and the structured questionnaire format, Hars and Ou (2002) hinder the achievement of their stated goals and views. Such methods thwart a deeper exploration of what developers' experience, and deny their subjects, the developers', to challenge any preconceptions embedded in the researchers' terms and concepts.

Orlikowski and Baroudi (1991), on the other hand, examine an interpretative and critical perspective and use example from Information Systems (IS) to highlight the strengths and weaknesses of both approaches. In this research methodologies and methods are used in ways which, given the dominance of the positivist perspective in IS, was not previously possible (Orlikowski and Baroudi 1991:16-23). The authors open up new theoretical perspectives and ways of doing research. They argue for example that the same method can be used but the results will differ according to whether one adopts a positivist, interpretative or critical perspective.

Methodology

This research is an exploratory study into the constitution of identity in the Samba community. The results presented are based upon an analysis of documents, websites, internet-relay-chat (IRC), mailing lists, private correspondence and face-to-face interviews. The primary method of data collection was the semi-structured interview method. This data comes from five in-depth interviews (see Appendix A for the interview schedule) conducted with Samba team members, with an average of one and half hours per interview. Interview took place from August to October 2004.

A Case for Qualitative Research

Considering this is the first detailed sociological investigation of an OS community, it is appropriate to take an exploratory stance whereby the phenomenon is approached with a broad set of research questions and a flexible method of research which allows the participants to describe their experience in their own words (Bryman 2004:287). The flexibility of the research method and methodology allows developers to construct their own meaning and the investigation of emerging themes as they arise within the context of the general questions of identity and community.

This thesis uses a qualitative research strategy, in this case inductive, epistemologically interpretivist and ontologically constructionist. An inductive approach is one which begins with "detailed observations of the world and move[s] toward more abstract generalizations and ideas" (Neuman 2000:48). The specific advantage of using the inductive method is that research becomes "flexible and lets data and theory interact" (Neuman 2000:146). The interpretivist epistemological position aims to understand "the social world through an examination of the interpretation of that world by its participants" (Bryman 2004:266). This also suggests a constructionist ontological position "which implies that social properties are outcomes of the interactions between individuals, rather than phenomena 'out there' and separate from those involved in its construction" (Bryman 2004:266). These major theoretical perspectives are cornerstones that informed how the interview methodology and the semi-structured method were approached. Data was collected

using semi-structured interviews, which also had a role in framing the terminology, themes and issues discussed. This method allowed the respondents to shape the exploration of the relationship between identity and community in several ways. It allowed for probes and invitations to expand on any issues raised by interviewees (Gilbert 1996:97); as the interviews evolved some questions where added and others removed (Gilbert 1996:142-144) and the general research questions were refined (May 1998:110-111).

Doing Qualitative Research in Open Source Communities

The process of doing this qualitative research is closely akin to the steps outlines by Bryman's (2004) *Social Research Methods*, and is shown in figure one with some minor alterations[1]. Beginning with a general research question subjects were selected and data collected. This data was interpreted and shown to interviewees for their feedback. This allowed the research questions to be refined.

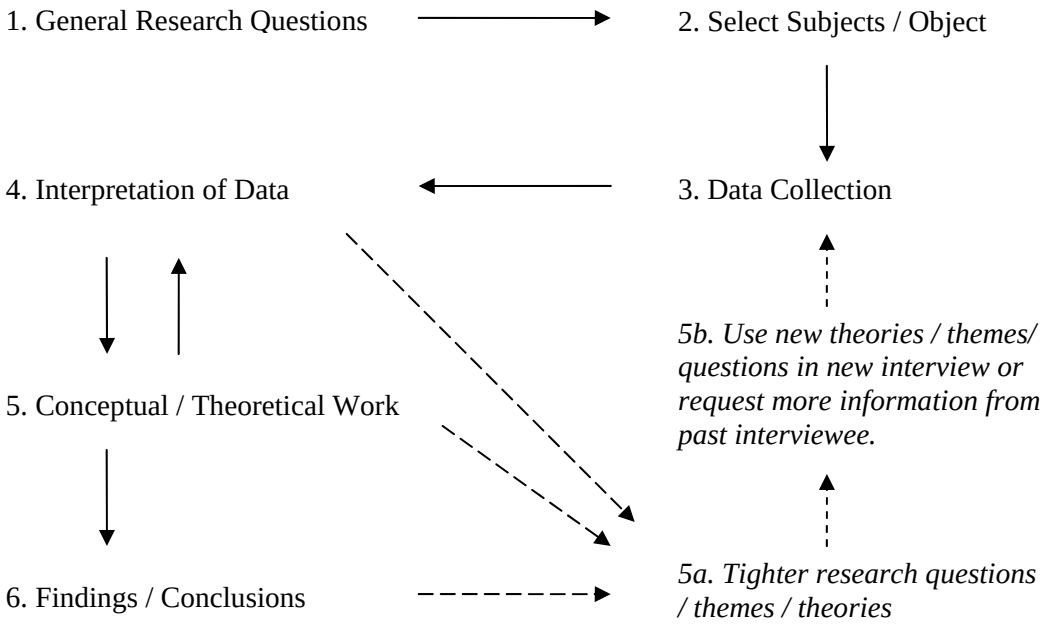

Figure 1 Steps taken in conducting interviews

[1] The original figure on which this is heavily based is figure 13.1 (Bryman 2004:269) and is replicated here with full credit to its original author.

Although Bryman's main steps were used, adaptations were made due to the specific context; that is, in researching an OS community where sharing of information is the norm all participants were made aware that this research would be made available for all to draw upon under a under *commons license*[2]. This was an attempt to operate within similar norms and values as the community. These people had shared with me information and I sought to engage in what appears to be a common principle to OS software developing, I was giving back. Beyond this sharing principle, I wished the participants to be able to contribute in the shaping of there representation beyond the interview itself. To do so, full or partial transcripts were sent to all participants and comment was encouraged. Exchanges of ideas and theorising on themes did occur and helped yield a depth of understanding and representation not otherwise possible.

Researching Samba

The Samba community was chosen for several reasons. Firstly, Samba is a mature project with a stable[3] release or set of releases; this would ensure that the project would be active during the research. Secondly, many Samba developers are local (located in the ACT and NSW), which would ensure access to community members. Thirdly, the Samba project is large enough to ensure that the number of active developers was sufficient to enable data collection. This information, as well details on its founder and other members of the Samba team, was ascertained by investigation of the Samba web site[4].

Following this, I joined the Samba technical mailing lists to learn about the activity surrounding the project[5]. It became apparent that active developers were discussing issues in an Internet Relay Chat (IRC) channel called #Samba-technical[6]. I began to 'lurk' on the channel to assess who was actively developing. Around the same time I began to create my own web site on the School of Information Technologies web-

[2] See http://www.creativecommons.org for more information on this service and method of releasing material under the licences they offer.
[3] Software can be distributed with varying degrees of 'stability'. When software is first released it often contains bugs and as time progresses these bug maybe dealt with leading to less errors. The more robust the software is the more stable the software is seen to be.
[4] Samba website available at http://www.samba.org [accessed 20th October 2004]
[5] Also available in achieve form from the samba website http://us4.samba.org/samba/archives.html [accessed 1st July 2004]
[6] Additional IRC logs are available at http://irc.vernstok.nl/ [accessed 15th July 2004]

server[7] for a number of reasons. Firstly, it is always a good idea to create a 'web presence' when engaging in this type of research. This allows potential interviewees to learn more about the researcher and have a more considered response to requests for interviews[8]. Secondly, I licensed the site and its contents under a creative commons style license, in part an attempt to signal my own stance in relation to the research process. In doing so I was seeking to build trust by identifying myself in ways which are familiar to members of the Samba project.

Throughout the month of July I refined the information statement and request for interview document [see Appendix B]. A copy of both was sent to the founder and project leader via email and traditional post. After no response for a couple of weeks I managed to have a brief discussion with the project leader on IRC. These talks eventually lead to a phone call from the project leader at the end of July. As a result of this discussion, a number of my preconceptions about researching OS projects were challenged. I had originally intended to interview 10-15 Samba developers but since there are only 3-4 current active developers in Samba within Australia I was unable to fulfill this early expectation. I had, however, made positive contact with the project leader, and used snowball sampling method (May 1998:119), where the sample is derived from contacts made through current interviewees. The risk of this method was that it possibly omits the voices of those not recommended by the project leader or others. This was minimised due to small sample size of core developers available.

Five interviews took place from August to October 2004. The duration of face to face interviews ranged from one hour and fifteen minutes to three hours, and took place at locations convenient to the participants. An average of two follow up interviews were conducted by phone and email. All face to face interviews were recorded and full or partial transcripts were provided to participants for suggestions and alterations.

[7] Personal site is available at http://www.it.usyd.edu.au/~nearnsha [accessed 10th August 2004]
[8] Indeed, one respondent even began an interview by mentioning that they had visited the site and was intrigued to know what I was doing.

Chapter 1 Definitions

1.1 Open Source

In the early years of computing from the 1960s to the 1970s, many users of computers were frequently required to share software to achieve their tasks. This necessity to share arose from the underdeveloped nature of software. Users could often address any shortcomings and faults by 'hacking' (refashioning or developing) the source code. Source code is supposed to be *human-readable* language and commands, access to it allowed these users / developers to make the changes necessary to fix issues.[9]

By the mid to late 1970s, companies began imposing restrictions on these users by the way of license agreements. The agreements limited the ability to share software, modified or not, by making it illegal to share copies. This movement to proprietary systems also became associated with an increased guard on the source code upon which such software was based. Companies had awoken to the profitability of source code, encapsulated as "Source code = the secrets to our success'.

Hence they sought to ensure their business success in terms of privately held intellectual property. It was, and still is in most quarters, espoused that if source code was made available to everyone, someone could steal the companies' secrets to success. Hence software became a product to be purchased by users on terms laid down by the companies. These terms made the modification, use and redistribution of software prohibited, or at least requires the express permission of the owner to do any such activity. Logically such software was also shipped in machine-readable code form only. This type of software became known as proprietary software, proprietary

[9] To make this phenomenon clear, consider this example. Imagine you had some software that allowed you to type the letter "a". Whilst using the program you find you want to type the letter "á" but your software does not have that feature. If the source code was *freely* available and you had the skills, you could simply hack the code to add your much needed "á" feature. Once done hacking, you would simply turn (compile) this code into machine-readable code and run your fixed up (patched) software that now allowed you to type the letter "á".

in the sense that the software was privately owned and the right to do anything with it was restricted to its owner-creators[10].

In 1971, as a freshman at Harvard University, Richard Stallman began work as a technician at the Massachusetts Institute of Technology (MIT) Artificial Intelligence (AI) Laboratory (Williams 2002). Stallman became increasingly frustrated with the increased limitations placed upon what he was able to do as a user and technician of the lab's software. As licenses became more restrictive Stallman launched the GNU Project in 1983 (Stallman 1999). The goal of the project was to create a totally free and complete operating system. Although not succeeding in that goal, the project created some important and popular software (Emacs, GCC) whilst upholding the early principles of sharing. As the GNU project grew, the Free Software Foundation (FSF) was formed soon after to guard against limitations and foster free software development[11].

The principle method used to ensure this freedom was a piece of software licensing called the 'General Public License' (GPL), also known as 'copyleft'. This license embodied both the ideological and practical goals of the FSF. A copyleft program grants "everyone the right to use, modify and distribute the program *on the condition that the licensee also grants similar rights over the modifications he has made*…[thus] everyone has to have free access to the program but it is protected from becoming someone's private property" (Mustonen 2003:101). In Stallman's own words;

> "The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program, and distribute modified versions--but not permission to add restrictions of their own. Thus, the crucial freedoms that define 'free software' are guaranteed to everyone who has a copy; they become inalienable rights" (Stallman 1999).

Whilst the GNU General Public License created freedoms it was freedom at a price; namely that no-one may add restrictions to or alter the license. The GNU General

[10] Please visit http://www.gnu.org/philosophy/categories.html for more information on categories of software.
[11] For the most up to date musing of Stallman and the GNU project please visit http://www.gnu.org/gnu/thegnuproject.html .

Public License puts software into the commons but with restrictions on its use. The GNU General Public License "copyrighted the software and then extended users very liberal rights for making innumerable copies as long as the users didn't hurt other people's rights to use the software" (Wayner 2000:87). This requirement to distribute GPL software under the same license was aimed at upholding the ideological position that no one person or company could horde software. Once software was created using the GPL it was forced to remain so; and this aspect of the license became known as the *viral* clause. In other words free software could not be placed within a piece of proprietary software. This clause aimed to ensure the freedoms of developers to use, modify and distribute software.

Many, including Eric Raymond, saw this viral clause to be anti-commercial and impractical (Raymond 2001:69-70). In addition the use of the word 'free' caused confusion outside the community with it being associated with gratis. In 1998 Raymond and others set up the Open Source Initiative (OSI) and sought to fashion a 'tamed' or modified variant of the FSF ideological concept of "free software". This is the origin of what is known as Open Source' (OS). The OSI allowed various licenses to be considered OS even if they did not require the re-release of software under the same terms. Whilst the GPL required redistribution within the same turns, in turn ensuring that no GPL code could ever be used in proprietary software, the OS definition *allowed* redistribution under the same terms but *did not require it*. Thus the ideological battle against software hording was removed, and in turn OS was seen as a more 'business friendly' since the initially revolutionary zeal had been mitigate.

It is important to note here that the OSI recognises GPL and its derivatives as OSI approved licenses. As Lerner and Tirole (2002) state;

> "These new guidelines did not require open source projects to be 'viral': they need not 'infect' all code that was compiled with the software with the requirement that it be covered under the license agreement as well. At the same time, they also accommodated more restrictive licenses, such as the General Public License" (2002:203-204).

The OSI are hence not by definition in direct opposition to Stallman, rather involved in an "effort to argue for 'free software' on pragmatic grounds of reliability, cost, and strategic business risk" (Raymond 1998). The OSI places more emphasis on ensuring

the wider involvement of business community in OS. This was met with some opposition by Stallman and the FSF based on the view that the OSI was not protecting the freedom of access to the source code.

In is important here to distinguish between the GPL and BSD style licenses in order to properly understand distinctions made later by developers, GPL being the most common OS license and BSD being the second. Taking the lead and simplifying the tables from Michaelson (2004) it is possible to look at a licenses from the perspective of the code itself, as shown in table two.

Restricted ←→Free	License	Properties
	GPL	Everything free all the time.
	BSD	Free but can be made proprietary.
	Proprietary License	The source code is not free; you have to pay for it.
	Proprietary Closed Source	The source code is not available.

Table 2 Freedom from a Code Perspective (Michaelson 2004:42)

It is also possible to view a license including a developer's choice as shown in table three.

Restricted ←→Free	License	Properties
	BSD	Take the code and do what you want.
	GPL	Take the code, but you have to make it Open Source.
	Proprietary License	Don't take anything unless you pay.
	Proprietary Closed Source	Take nothing.

Table 3 Freedom including a developers perspective (Michaelson 2004:43)

So a BSD style license gives a developer complete autonomy to take the code and make do with it as they see fit, including changing three lines and releasing it as proprietary closed source software. The GNU General Public License, whilst more restrictive ensures that such practices are not permissible under it licensing agreement.

Despite all of this in the daily workings of most developers, both Free Software and Open Source are terms used interchangeably and this is edified by the occasional use of the abbreviation FLOSS (Free Libre / Open Source Software) in much literature. Whilst FS and OS may refer to a particular ideology for some, and this is shown in artifact with the licenses they accept, the actual development of software often means those terms are equally used without the connotations associated with them in any strict ideological sense. As a method of developing and distributing software, OS and FS software development is based on the concepts of sharing and the rights of everyone to run the software, copy it, modify it, and distribute those modifications. For the purpose of this essay the term OS software is taken to mean an umbrella name for both the method of developing software and the types of communities that arise around such a task. Where respondents do differentiate between the two terms it will be made explicit.

OS Software developers have a distinct profile. The FLOSS survey (Ghosh and Glott 2002) found that males represent 98.9% of all developers. The male domination of these communities has yet to be adequately accounted for, although some illuminating debates have centered upon the gendered nature of the tasks and the bonding over technical expertise is a masculine rather then feminine socialization (Kleif and Faulkner 2003). As Holtgrewe notes, “Social exchanges and groupings around technology then are likely to be male-dominated and if they are, they can easily establish a path-dependence of not attracting women” (2004:139).

These participants are most likely to be between the ages of 16 and 36. 60% of participants live in some kind of partnership and 17% have children. They are highly educated and mostly professionals or university students in the IT sector. There is an average of 11 years of programming experience and the income level is not high but the study cited the large share of students represented in the community. Most (70%)

do not spend more then 10 hours a week on OS projects, a point confirmed by the Boston hacker survey (Lakhani, Wolf et al. 2002). The Boston survey also noted that the demographic of OS was global in composition with respondents coming from 35 countries, but is dominated by western societies. These projects, whilst global, highlight the domination of developers from western societies. As Bauman (1998) writes, although OS does is global it does not challenge the new forms of inequality with which globalization is associated with (1998:6). A typical software developer is a well educated male, most likely not paid for their effort, aged around 23, in a partnership, living in a liberal democratic society and strongly categorizing the OS community as their primary community of identification (Lakhani, Wolf et al. 2002). This final point should be kept in mind as we not look deeper into the key motivations factors and the relationship between identity and community.

1.2 Defining the Samba Project

Samba is both an OS software product and development community. Samba is one of the fastest growing OS software on the market Lee and Davis's (2003). According to Lee and Davis (2003) its growth and popularity "is attributed to Samba providing unique services across different platforms…with the ability to mediate between Unix and Windows systems" (2003:44). In other words Samba solves a common need of sharing resources, such as printers and files, between different types of computers. It so happens that these types of computers are the most used in the world in turn adding to the software's popularity. Collier-Brown et al. (1999) also point out that "[b]ridging the gap between systems as dissimilar as Windows and Unix is a complex task, which Samba handles surprisingly well"(1999:ix). This complexity, coupled with wide need for such a piece of software and the Samba's proficiency at handling such a task, has quickly lead Samba to be the software of choice in this area.

As a piece of software, Samba is available under the Free Software Foundations General Public License (GPL). On its website, the software is described as both Open Source and Free Software. Hyper links are given to both the OSI and The Free Software Definition as defined by the FSF. This implies that Samba recognises, at the level of self-description at least, the legitimacy of both the OSI and FSF in defining

what Samba is. The project is widely known within the OS development community and the product itself has a strong end-user base, in large part due to its success at achieving its goals of interoperability between distinct operating systems.

The Samba project has a structure based around the development areas within the project. Like most modern software, the production of Samba is divided into modules often worked upon by particular people. Who works particular aspects is flexible and changes over time. Developers who are on the Samba team have the ability to make direct changes to the code, by 'committing' patches to a concurrent versioning system (CVS). The CVS is a software mechanism that controls disparate versions of code as it is developed over time thus assisting several developers to collaborate on the project. New developers must submit their changes via a team member who can check the code and, if it is approved, commit it to the next version of the software.

As Ye and Kishida (2003) noted: "Although a strict hierarchical structure does not exist in OSS communities, the structure of OSS communities is not completely flat. The influences that members have on the system and the community are different, depending on the roles they play" (2003:420). The role a Samba team member plays allows them direct access to make changes to the code; this is known as the ability to make unsupervised commits. It also gives them access to a discussion list, not available to non-members, about the future direction of the project. In general most team members will work on a particular aspect of the code as the code is grouped into modules based upon the functions performed. Beyond that coding work, some team members play supporting roles such as maintaining the Samba's website or writing technical documentation. All of these roles are not fixed and movement into different areas of the project is common.

1.3 Stumbling Upon a Community: From Task Orientation to Community Participation

Any investigation of the Samba project requires an appreciation of why developers contribute to it in the first place. This section examines the paths taken from first engagement with the Samba project to membership in the Samba community. In

doing so, the section illustrates how the community environment is a significant factor in understanding participants continuing involvement with the Samba project.

1.3.1 Joining Samba: Houston…we have a problem

"Well I was a user of free software first…"
(Samba team member)

Members of the Samba team recalled their first interactions with the software in relation to a problem they had as end-users. For most developers, the reason for first engaging with the community in other then an end-user role is due to a desire to amend the software to meet a personal or professional need:

> "I had a particular problem with it so I started contributing. I started fixing problems and contributing."
>
> "There were things that were not quite right about Samba, how some bits of the [software] worked and were handled. That is certainly what started me into it…"

Thus to address these problems, developers contribute information in the form of patches or bug reporting in an attempt to meet the personal or professional need.

The ability to contribute to the project at a meaningful level, such as submitting a patch, can be a long process. Whilst the source code is available to all in human readable language it does not mean it can be easily understood.

> "Technically it takes a while to get to know a project well enough to contribute meaningfully… so sometimes there is a small bug you can patch. That might be the first step but it can take years or more to get to understand this stuff."

The movement from end-user to solving a personal or professional need can take some time and knowledge. This transition from fixing problem to membership in the team via meaningful contributions is not simply just restricted to a technical understanding; it is also based in a social process of 'enrolling'. Enrolling involves the recognition by existing team members of a developer's contribution and future worth to the project. At the same time this requires the budding developer to have shown

that they consider more then meeting the initial individual needs that brought them to the project.

It is a well recognised by the interviewees that there are stages or milestones in moving from early contributions to a membership of the Samba team. As one team member remarks:

> "It's a progression … the first time they come on the email list and then that is kind of a milestone and maybe they get more and more involved after that."

Another notes that,

> "It has usually been after six months of lots of work and when someone becomes too tired to be your commit agent[12], then you are invited onto the team having been seen as good."

Whilst another explains:

> "Most projects wouldn't give developer access to just anyone. You have to in some sense prove yourself; that you are not going to put stupid stuff in there."

So when beginning to contribute to the project, it is necessary to join the mailing lists, reach a understanding of the code to contribute meaningfully and show enough skill to warrant a place on the team. But the question still remains, how does this enrolling process occur, what is the role of existing team members and who invites someone to join the team?

1.3.2 The Social Process of Enrolling

Several developers mentioned that personal contact with team members encouraged them to become more involved with the project:

> "I was working at ANU [Australia's National University] doing a system administrator job and [the Samba project leader] was working there ... So I'd had an idea for something I'd wanted to do and I said 'would this be a good idea' to do with Samba. And he went; 'yeah go for it'. So that was how I got started in it, contributing. We had this huge tape library at [ANU where] I was working. It was interesting getting Samba working on that so people could make more use of [the tape library]."

[12] A commit agent is a team member you mail patches to, they check them and they submit them on your behalf.

The initial desire to amend Samba comes from a professional need but before this idea is turned into code, it is pertinent to note that the budding developer made contact with the project leader. There is no evidence that this type of behaviour is essential but it does indicate that, even in the very early stages of engaging with the Samba project, interaction between developers often involves an exchange of ideas and opinions about what is appropriate or negligible.

This social aspect is highlighted in another developer's account of early interaction with the Samba project.

> "I stumbled into in it [the Samba community] by wanting to change the Samba software. All very naively just getting into doing a few things and um... you know, I was on the Samba technical mailing list doing a few bits and pieces... just little bits. I started off with changes to the PAM code then got a phone call from [a team member] who encouraged me along a bit. Anyway, so a bit of encouragement from him and I got to know [the project leader]."

This is an example of how personal contact and feedback happens to spurs developers to contributing to more aspects of the project. In the first example mentioned, the project leader indicates his acceptance of a developer's idea to make the system operate effectively in his particular work environment. In the second example, a team member's encouragement and 'getting to know' the project leader are both mentioned in the early stages of contribution.

One developer discusses how he was asked to join the Samba team.

> "Over the exam period winter break I did a couple of weeks work experience in VA Linux…before going belly up. That was with [the project leader], [a Samba team member] and [another Samba team member]. I was in there for a couple of weeks [and] basically [the project leader] had offered me a position on the team to maintain the Samba build farm and that is, you know, what got me into the team. [The project leader] didn't want me to disappear elsewhere. So apparently [the project leader] and [another team member] had a talk and had decided that I would be brought onto the team…"

The skills of this particular developer had been encouraged by others already on the team. The recognition of this developer as someone of value by the project leader and another team member in turn lead to their invitation onto the team.

The significant point here is that recognition of skills alone was not sufficient to prompt budding developers to increase their participation in Samba. Established developers actively fostered suitable candidates, and this encouragement was felt by those individuals. This accords with Edwards s(2001) and Ye and Kishida (2003) use of legitimate peripheral participation (LPP), a concept developed by Lave and Wenger (1991) based on their examination of communities of practice. 'Learners', in this case potential developers, are considered worthy additions to the community and as such are seen as 'apprentices' within the community. Such 'apprentices' must be seen as legitimate peripheral practitioners, and as such their participation is fostered by existing practitioners.

Recognition and encouragement by team members is a key factor in the transitions from being an end-user to team member. It is important to mention that the skills recognised are not necessarily always coding skills. One developer explains:

> "I asked [him] onto the Samba team ... because he has ... always sat in on the Samba technical IRC channel… he has been one of the few people who has associated himself with the Samba project because he could and because he enjoyed being in an environment of those technical discussions and things."

The diverse ways in which members can contribute to OS projects is noted by (Dempsey, Weiss et al. 2002) who identify two main type of OS contributors; those that contribute code and those who participate in discussions, report bugs, produce documentation and engage in other non-programming activities Nevertheless the same overarching pattern occurs with the Samba team members interviewed here. A developer begins to contribute to the project for reasons of utility, then becomes engaged in a social process of recognition by team members and finally is invited onto the team. These findings are similar to those presented by Krogh et al (2003), who propose that a 'joining script' as 'a level and type of activity a joiner goes through to become a member of the development community' (2003:1227).

The capacity to enrol others into the community is held by existing team members. The transition from engaging with Samba in order to solve a personal or professional software problem to that of community member involves a number of social interactions that create an environment which fosters their ongoing participation.

The significance of an encouraging community environment can be seen in respondents' observations concerning the enjoyment they experience when coding. All respondents stated that they enjoyed coding:

> "It is fun… the programming itself is fun… I've always loved programming."
>
> "It is what I want to do, I enjoy doing it... I do enjoy a good programming problem. I do enjoy building those types of things."
>
> "I think one of the things I get a lot of satisfaction from is solving bugs. It's like solving a puzzle."

There is pleasure in solving programming problems at an individual level, but this is also connected to the space in which they contribute. An encouraging environment prompts developers to work on problems that are not simply their own personal and professional needs, and facilitates the shift from meeting individual needs to identification with community needs. As one developer states:

> "So some things I did for fun or because there was a need to look at it."

1.3.3 Conclusion: Problem to Participation

The complexity of why developers participate in such a project and the role of others is best expressed by one respondent who states:

> "I'm not doing it for money; it comes down to recognition and appreciation from others. Like my other major hobby is photography, I don't do that for money or whatever, it's another creative thing I do that I hope others will appreciate and it's something I enjoy."

The complexities of making the movement from a first time end-user, to a contributor, to finally team membership is thus not simply a matter of contribution and more of the same until you are invited onto the team. It involves a set of social interactions, themselves changing a developer's view, from having just their own needs addressed to the needs of the greater community and project. This movement takes place within a particular social structure and it is to this structure that this thesis now turns with the express aim of understanding its operation and impact on developers.

Chapter 2 Samba Community Structure

> "I don't know if I'd say there are no formal ties [structures]. They are not as formal as a work arrangement where you have signed a work contract ... But there are social formalities."
>
> *Samba team member*

The Samba community is organised around a set of structures both informal and formal. In this chapter the structure of the community will be explored through the eyes of the Samba team, with existing understandings of OS community structure both confirmed and challenged. In doing so it is firstly suggested that the role of social capital is an important structuring agent and secondly that this structuring agent is best understood as operating within a network rather then a hierarchy. The basis upon which developers interact is not a simple work agreement with the threat of being fired looming over decision made, rather a set of succinctly social relationships are entered into that structure the development process.

2.1 Decisions & Power Structures

> "…the people who contribute the most tend to be the project leaders and the ones people tend to follow in the project…"
>
> *Samba team member*

As Ye and Kishida note, "[t]he roles and their associated influences in OS [Software] communities can be realized only through contributions to the community" (Ye and Kishida 2003:421). All OS projects have leaders or leadership teams who possess some type of authority in decision making within the project community. The role, rights and fragility the project leader has in imparting those two facets is best expressed by this Samba team member:

> "The project leaders tend to get the right to choose which direction the software engineering tends to go in…and it's not power in the traditional sense. [C]ertainly if somebody did a bad job of being a project leader then eventually they would become not the project leader (sic), either through

> neglect or bad decisions and somebody else, you know, would emerge as project leader. The project leader tends to be the most active programmer or the one who is most deciding the structure or the plans for the project."

Project leaders can claim a right to choose the direction of the project (Ye and Kishida 2003:420). From the remarks presented above it is obvious that the most active programmer or the one making the big decisions tend to be *recognised* as the project leader. The project leader is not a fixed position, as leading a project successfully requires other people to follow ones decisions and bad decisions imply the lost of those rights.

As an organisation, the Samba community and its participants cannot directly force any other person to behave in a certain way. There is a fundamental lack of explicit top-down power[13] evident in the day to day operation of the community. The freedom to modify source code permitted by the licensing schemes is a condition which makes possible the development of non-authorities power structures in my OS projects. As one developer commented:

> "Open Source projects don't have much power to compel people to do something ... In the army they can tell you, 'you must do this or you will go to jail'. And [my employer] can say, 'well you must work on this and if you don't we are not going to pay you anymore'. And then Samba; they can't not pay you right? ... all they can do is stop listening to you ... not read your email or not take or consider any changes from you."

And

> "…the power only comes from the people wanting to be controlled and being willing to go along with the decisions of others... if they ever disagreed with them, then they can choose to do things a different way if they want to."

The concept of a developer having "power over" another is not something evident in the day to day operation of the Samba community. The social sanction of ignoring another developer does occur but it is not a means to ensure people will follow through with a decision; the project leader must rely on the other developers giving him authority to make decisions on their behalf. This lack of top down power in

[13] By top-down power, I am referring to a notion or mode of power that imply a hierarchical use of force. The ability for actors to choose to either recognise and thus comply with or ignore leadership decisions means that no-one can be directly forced to do anything.

compelling people to behave in a particular way does not mean that the ability to influence others does not exist or that some type of power relations are not evident. However, in the absence of a traditional type of power relations, the question is; how do project leaders enact the right to choose the direction of a project and upon what basis are such rights based?

2.1.1 The Final Say Principle.

In matters of community composition, such as who is a team member or non-team members, a project leader can make decisions for others;

> “There is [a power structure], but it's rarely invoked. So probably [the project leader] has the final say. You've probably heard before where somebody got kind of removed from the team for being pretty destructive. So in a sense, that is power right? I guess it's more like a moral authority than a power, a moral and intellectual authority.”

Whilst this act of forcibly removing a team member is an example of top-down decision making but it is noted that this act is seen by the interviewee to be based upon the moral and intellectual authority of the team leader rather then the operation of a power relationship. This wording may seem odd at first but consider that this type of decision making rarely occurs. As shown in figure two, power in terms of force (that is making decisions for people) does exist within the community, but this type of power is rarely exercised.

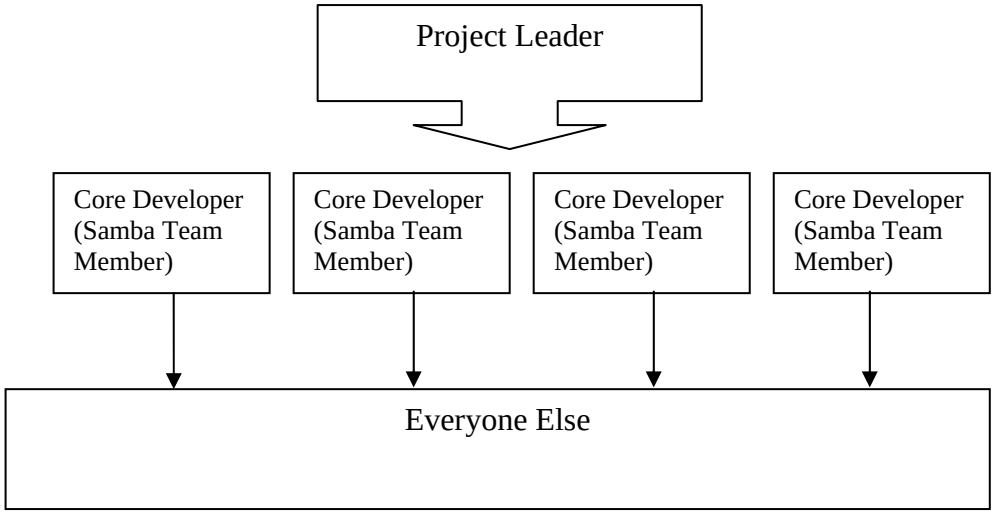

Figure 2 The final say principle

All developers interviewed were quick to assert the freedom of their own decision making rather then the implementation of someone else's will and minimised the impact of power relations within the community. Developer where quick to note that whilst power relations do exist they are more complex than a top-down power structure:

> "Well sometimes [the project leader] will come out and involve himself in an argument and say, 'this is the way it will be'. It's like Linus, he does this too. There will be this raging debate and Linus will say this is the decision and if you don't like it you can go away ... And ... he has a lot of respect because he usually is right. But he [Samba project leader] will go back on stuff if it turns out he is wrong. He is less authoritarian ... certainly there is a role of a decision maker that is respected amongst the developers which is a result of status."

The condition of 'final say' is here of course not absolute. The project leader can make final decision within the Samba community about its composition but the results of such decisions still rely heavily on the leader's status or social capital as discussed below. In recognising the moral or intellectual authority of leader, the emphasis is placed on the qualities of the individual developer rather then their current position, say as a project leader or Samba team member.

2.2 The Role of Recognition / Status / Respect / Influence

For these developers recognition is articulated in terms of respect and status. Recognition and status then is very much dependent on the *perception* of others. The path to acquiring status is seen by most team members to be related to the work done within OS projects:

> "Status tends to come from your contributions over the years."
>
> "It's your... level of participation in terms of helping people, helping out with admin stuff; the technical stuff, its contributions at all levels that helps build status."

As a developer, status is built over a substantial time period and is directly related to contributions made to the project.

Status gives a developer the potential to influence others.

> "People tend to listen to you more when you say something about a project… and so people then place a bit more weight on things you say."

And more in:

> "Once people listen to what you have to say you can accomplish things using that [status]… so if lots of people think what you say is important then you can persuade them to do something you may want them to do."

As status increases developers become more and more receptive to ones ideas. Having other developers consider what you say at all should not be underestimated. Developer's time is short with information overload common, picking out worthwhile ideas, solutions and suggestions can be difficult[14]. Status is one indicator of information worth considering:

> "I got a bug report from [a person] who is one of the core kernel maintainers and ... the first thing that tells me is that he is a smart person, who [I'm] not going to be under enormous pain to support…"

Recognising the person as someone with high status ensures that this team member will read and consider the bug fix. The intellectual authority that this developer places

[14] Most Samba team members recognised that no developer has ample time on their hands to wade though the main sources of information such as the mailing lists. It is quite common for Samba team to leave emails unanswered. This often happens, not due to being unwilling to communicate, rather the shear volume of emails received. In subscribing to the email list, I once did not check it for two days and had 160 emails unopened when I finally did check.

on the patch sent to him is founded in the knowledge of what being a kernel maintainer requires in terms of skills.

It is necessary here to introduce the notion of social capital as a tool to understand how influence can operate in environment lacking in overt power relations. Social capital refers to "relations among persons that facilitate action, embodied in the collective norms of communities that extend beyond immediate family members and the trustworthiness of the social environment on which obligations and expectations depend" (Jackman 2001:14216-14217). It is also thought that "high levels of social capital foster cooperative and perhaps altruistic behaviour, a theme with communitarian origins" (Jackman 2001:14217). As Portes (1998) points out, "the consensus is growing in the literature that social capital stands for the ability of actors to secure benefits by virtue of membership in social networks or other social structures" (1998:6). The possession of social capital translates to the ability to secure behaviour within a community such a Samba where cooperation is not guaranteed and reciprocal behaviour not assured.

As one's social capital increases so does the ability to enroll others in collaborative ventures. This capacity was explained in an illuminating way by one team member:

> "It gives you this force multiplier if you will. [The project leader] is a very bright person but he only has X hours in a week to do stuff but he can influence ten people who are less smart to also spend X hours per week ... [the project leader] can say, 'ok so we are going to rewrite Samba from scratch for version four and it's going to be like this'. Rewriting from scratch is a really dangerous thing in big engineering projects because there is some large chance of failure... it's pretty much the most risky and expensive thing you can do. But if you've been running this thing successfully for ten years then you have so much status that you can say, 'ok we are going to do it'. He cannot make them start working on it but he can go off in that direction and try to make people follow."

The social capital of the team leader is clearly evident. With the risk so high in attempting to build anew the next version of Samba, the need to enroll others takes centre stage because such an initiative could easily fail. Yet other developers have followed the project leaders' initiative and Samba four is well under development at

the time of writing this thesis. The use of social capital to enroll others in one's initiatives, however, is a necessary but not sufficient element in successful enrolment. Community members must perceive the initiative as viable.

2.2.2. They tend to be right.

In everyday decisions, the willingness of others to go along with such a task is related to the belief that the decision being made 'makes sense'. Certainly it is easy to over-emphasis the role of social capital in enrolling people. Developers stated that people follow a particular person's decision because it tends to be right.

> "We respect our elders to an extent, [the project leader] and [another team member]; they have been around the longest. That's also because they tend to be right".

It is a mixture of social capital and the merits of an initiative that yield the collaboration of other developers.

Although social capital operates within the Samba community, developers do not explicitly seek such status, nor is it ranked within the community.

> "There is no explicit status apart from your membership to the team. Because status is never explicitly measured or judged, people just know who is an expert in a particular area and they respect their opinion on code that they understand and you don't. So we don't go and try to rank status. Samba isn't a measuring event. It's a project to achieve these goals of creating a Windows compatible server".

Whilst there is a desire to have software appreciated by the other members of the community, contributions are perceived primarily in terms of achieving the goals of the project itself, that is, the creation of a Windows compatible server. As stated previously, although social capital operates to enroll others, the initiative must be perceived as having merit in and of itself in terms of Samba's overall goals. It is the merit of the initiative is the overriding determining factor. This ascription of merit in general is a dominant feature within the Samba community. The question arises: is there a relationship between the concept of merit and the structure of the Samba community?

2.3 Problematising Meritocracy

> "I think [meritocracy] is very important. If you are no good you won't have any respect or status."
>
> *Samba team member*

Meritorious communities are ones whereby the 'the best' or 'most correct' thought is the determining factor in making decisions. According to Raymond (1998) OS communities are structured on merit (Raymond 1998). The person with the best ideas is recognised by others and given more responsibility and authority within the community. There is a connection between the merit of a developer's work and their status and moral or intellectual authority. As Thomas and Hunt (2004) state it:

> "you are your reputation…submit poor code to an OS project, and the rest of the team will let you know in no uncertain terms-it's their reputations on the line…[and] as a result, OS communities are meritocracies"(Thomas and Hunt 2004).

Yet this depiction of meritocracy is oversimplified. Developers were ambivalent as to whether or not Samba operated as a meritocracy.

> "There is no formal hierarchy usually, but there is hierarchy because meritocracy leads to hierarchy based on merit. So I'm definitely part of a community that *tries to be a meritocracy* where possible and you can only really judge if you really are a meritocracy in hindsight because, of course, two different people might have different opinions as to what is correct way of doing things is. Supposedly the best method wins out, but in three of four years we might find out that, well no, the other person was right" (emphasis mine).

The community tries to select ideas based on the perception of their intrinsic merit. For this particular developer, there is a hierarchy based on merit, imperfect but in operation.

For other developers the term misrepresents the particular dynamics of the Samba community.

> "Meritocracy kind of implies that... the smartest person should decide ... Whereas in open source it is more like, whoever wants to do it can do it. So even if you have a person who is more qualified, if they are not going to do the work they don't have to, right? If they are busy at the moment and somebody else wants to do it then they can. It is kind of [a] market of ideas or competition of ideas... rather then a competition of people."

Many developers take issue with the connotation that 'the smartest person' should, by virtue of their intelligence and skills, be given authority to make decisions. This contradicts the notion that there exists developers who have moral and intellectual authority within the community. In other words whilst developers are happy to ascribe to the *ideal* that the best ideas should be chosen they retreat from a connection between the validity of a developers ideas and the rights that should give them to make decisions.

The idea of meritocracy acting as a structural dimension is not clear and comes across as contradictory at times. For some developers the connection between authority and reputation is one that simply leads to a meritorcractic authority:

> "I think 'authority' really means 'reputation'…So it is a meritocractic kind of authority, rather than a hierarchical kind of authority. Think more of respected scientists than field-marshals"

Whilst the perceived merit of one's contributions is a factor in deciding to collaborate in a certain initiative, one's social capital, constituted partly by the moral and intellectual authority ascribed to them is also important n understanding the structure of the Samba community.

It is interesting that the Samba team member above mentions the community as consisting of a market of ideas. This creates a 'value neutral' space, in which ideas are divorced from the developers promoting them; it is the ideas, rather than individuals, which are in competition. At the same time that this developer espoused the community as a 'market of ideas', others recognised that informal structures exist within the Samba project, and that these structures are not simply based upon the merit of ideas:

> "…there is definitely a distinction between core and non-core members of the developer community, but it doesn't really translate into any kind of

> favouritism. No-one is treated unfairly or anything like that… [but] there is a tighter knit social group within the Samba developers."

The team member wished to establish that this type of distinction does not translate to nepotism:

> "There is a definite social distinction between people who are really good personal friends as well. It is bonds of friendship from working together personally and professionally and ... it's a bit unspoken. There is this other level. It's based on personality too ...I'm sure that is present in other projects too you know, where someone will have more pull for perhaps no good technical reason at all…[but] I think that [meritocracy] is very important. If you are no good you won't have any respect or status."

When attempting to explain the structure of the community, in terms of how code is chosen, there is a tension in explaining how such decisions are made. This tension is between the realm of 'disembodied' ideas (merit) and the recognition of social capital. To say that the Samba project is solely structured around a meritocracy is to miss the interplay of these two factors. Rather than Samba being structured as a hierarchy based on meritocracy, it was suggested that the Samba community be understood in terms of a network.

2.4 Community structure as a Network Structure

As discussed earlier, OS communities are often seen as lacking a hierarchical structure. At the same time, however, there is recognition that some developers have a greater reputation and accompanying authority than others. The suggestion that the Samba community be perceived more as a network than as a hierarchy based on meritocracy came from the perspective of the developers themselves. As the interview with this developer proceeded, I drew a schematic representation of the structure being described and received feedback:

> "It is more like a network then a hierarchy. So like normally there is kind of [D1] and [D2] ... but... it's not constrained to that shape. If someone else wanted to take over something then they don't need to get permission."

This shape is shown in Figure three.

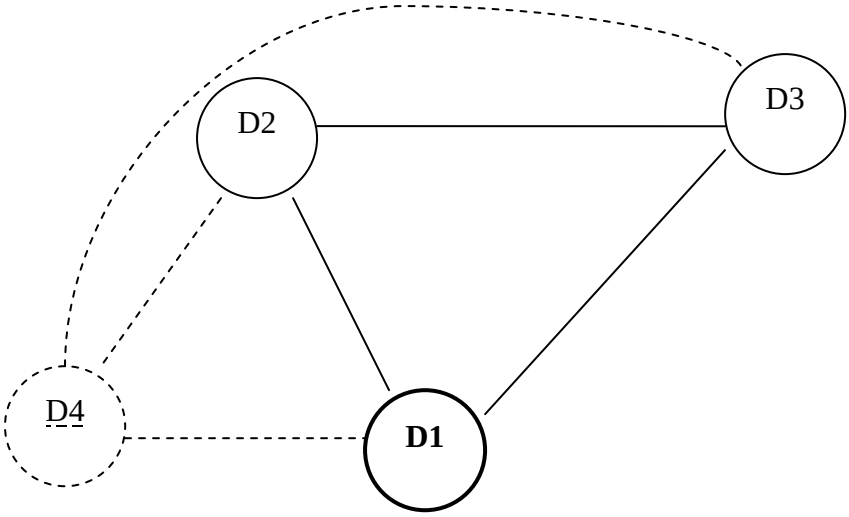

Figure 3 Samba as a Network structure

D1 in this figure represents the project leader and due to his ability to enroll others has a bold line and text. The authority to add new developers, represented by the dotted lines of D4, does not disrupt the existing community structure. There is also no reason why D2 could, over time, become a project leader, or for D3 to become more inactive. The benefit about thinking of the community structure as a network, is it gives us analytical flexibility to see, for example, how new nodes can be added or the strength of a developer's reputation can change. Yet by giving more weight to particular nodes it is also possible to incorporate some of the vertical relationships, i.e. the final say principle (as shown in Figure one**,** page 19) as well as the changing states of social capital:

> "Well I think there is obviously this sort of hierarchy in most projects like Samba… There is the leader type of person which is typically the person who started the project or whatever. So for Samba that would be [name removed]. And everyone sort of looks up to him; if he says something it is law. He is really the top dog and ... often there is a discussion on the mailing list and [the project leader] has the final say and while ... people can challenge [him] ... it doesn't happen very often. Most of the time people accept what [he] says."

In sum, the network has a central node that people respect, represented in Figure four below as a central node to which more nodes point[15].

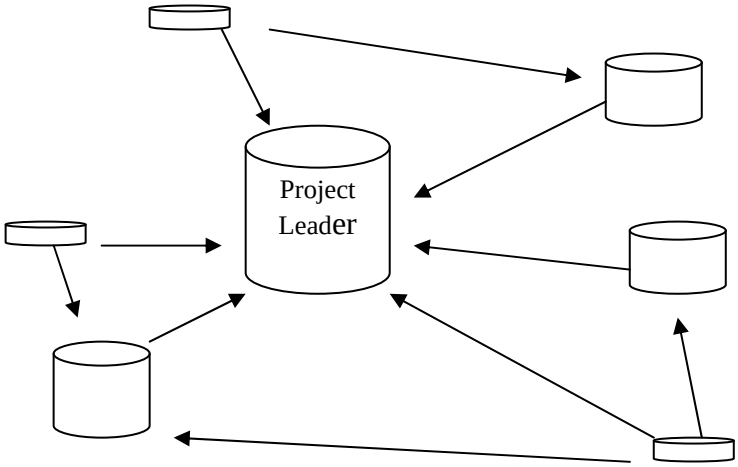

Figure 4 Authority incorporated into a network

In this figure each cylinder represents a developer. The larger the cylinder, the larger the reputation, and thus the increased likelihood other nodes will seek direction from them. The greater the reputation of a node, the more other nodes take heed of the decisions and opinions that node espouses. The relationships shown in figure four are not static; they are able shift over time.

As well as developers distinguishing community structure in terms of social capital, the merit of the idea, and personal friendships, they also addressed the access to the CVS:

> "the distinction ... between the people who have CVS access and those who don't."

It is necessary at this point to address CVS as more then just a code control structure.

[15] Please note this is not intended to be a conventional network diagram.

2.5 CVS: More then Code Control

> "It's a small trusted group of people who have commit access to the source code"
>
> *Samba Team member*

The ability to make changes to the Samba software hinges on direct or indirect access to the CVS. Samba team members will commit their code, as well as patches sent by others, to areas that they are currently working on. As shown previously, the ability to commit alterations and additions to the Samba tree is not given to everyone rather it is something earned.

> "Most projects wouldn't give developer access to just anyone. You have to in some sense prove yourself; that you are not going to put stupid stuff in there."

Developer access simply means commit access to the software. Once again, developers are quick to minimise the impact this code control structure has on individual developers:

> "…there is a control structure over the software itself, but I wouldn't call it a control structure because that implies we have control over other people... if somebody doesn't want to work on the project we can't force them to."

Nevertheless, the code control structure does impact how the community is structured because the ability to make commit changes influences how people approach the team members.

> "The Samba team…are the ultimate ones who I need to approve my patch so they are the ones I really care about, whereas if other people out there say things [about my work] it is not as important to me because they are not the ones ultimately making the decisions. And obviously I'm looking for feedback from the Samba team and someone who will actually take it and commit it."

Others will ultimately pay more respect to those with CVS access, since they are the ones with the ultimate decision concerning the acceptance or rejection of suggested changes to the code. The CVS, then, is not just a code control structure; rather, it plays a part in structuring the community and in turn impacts on how individual developers approach one another.

2.6 Conclusion Community Structure

The structure of the Samba community is best described as a network with formal differentiation based on membership to the Samba team and access to the CVS. Social capital exists as an informal but significant capacity that enables developers to enroll others in their initiatives within the community. Its significance lies in the fact that in lieu of formal power structures which can force an individual to comply with another's will, such as those existing within (paid) workplace relations, social capital is the only dynamic which can influence another. This influence may consist of enrolling others to collaborate in an initiative, or to comply with the norms of the community. As discussed, the operation of social capital firstly problematises the notion that the Samba community operates on purely meritocratic principles. Secondly, the CVS can be seen as more than a code control feature because those with access are recognised as being more significant due to their ability to accept or reject changes in code. This recognition of significance may constitute an aspect of social capital. So far this thesis has utilised the concept of social capital to investigate both the formal and informal structural differentiations within the Samba community. Having examined the structure of the community, the thesis now turns to an exploration of the implicit and explicit norms and values which exist within this network of ideas, social capital and personal friendships.

Chapter 3 Norms, Values and Shared Code

3.1 Sharing Norms in the Samba Community

The dominant norms and values in the Samba community centre on contributing source code to the project. It is shown that the ‘share and share alike’ principle is universal throughout the community and is codified within the licensing agreement. It is noted that such a principle is edified as ‘common sense’ and finds its way into the communication norms between developers and even end-users requiring assistance. In doing so the diversity and commonalities of norms and values exhibited within the community is explored with essential and universal principles spelt out. The act of coding itself is emphasised to be a social experience with consequences for how feedback is distributed and the risk of embarrassment managed.

3.1.1 Share and share alike

All developers expressed the opinion that the sharing of ideas, often in the form of code, is essential to the operation of the Samba project and is the core value of the community:

> “In the OS community as a whole [there is] the idea of sharing generally; that ... idea of standing on the shoulders of giants... sharing your code and your ideas rather then keeping it all secret.”

There is a culture of having ideas made available to others, and for others to engage in a reciprocal process. The concept of keeping ideas and solutions secret was rebuked using several rationales, including pragmatic considerations:

> “One is just a pragmatic answer, which is, if you kept [you patches] to yourself then every time there was a new release you'd have to apply the fix again and possibly update the fix. It would just be a big pain. So, like I run [an OS operating system] on several different machines and if I send a patch to them then the fix will automatically get onto all of my machines in fairly short order. Whereas it would be more work for me to do it myself”.

Recognition of the normative dimension was also evident:

> [T]here is kind of this social in-group thing... the people you most care about helping and impressing [are the] other people who are maintaining projects.

Developers also expressed the general wish to assist others:

> "...it is nice to help people... it is also a shame to waste the knowledge that you have built up… [by] not contributing it. So in a certain respect it is nice in an altruistic way to help someone out."

Firstly, then, developers recognised that if you keep your ideas and solutions to yourself, maintaining those ideas in the future versions of the software is difficult. In this way it makes much more pragmatic sense to submit a patch and have it fully incorporated into the software. Secondly, there is the normative dimension of wanting to communicate ideas to the people with CVS access, and lastly, developers acknowledged that there was pleasure in helping others. The wasting of knowledge by not contributing is reflective of a community environment within which such sharing is equated with assistance.

The rationales for 'sharing and sharing alike' only apply to developers who modify or require modification of the software. Developers have no expectations that end-users will contribute; however, should a end-user require assistance with using Samba, developers are clear about the conditions under which they would be willing to work with end users:

> "The community I would assist ... would be the [Samba] team [members] and a few individual [end users] who I've had interactions with and worked with positively…my notion of community would extend the users who are contributing back in some form…."

Sharing and sharing alike, expressed as contributing back in some way, is the basis upon which community is defined for this developer. The principle of sharing thus potentially applies to anyone engaging with the community, including requests for assistance made by end users.

3.1.2 GPL code: A material manifestation of community values

The normative principle of 'giving back to the community' is also evident in the way developers describe the behaviour of those who use Samba in relation to the license it

uses. In this way, the licensees are material, technological representations of this sharing principle. As one developer articulates it:

> "[T]he price that people pay for using my code is that they have to obey the license, a fundamental of which is they must contribute modifications of it back to the project, making it available, the base requirements of the GPL."

At its core the GPL has the requirement to share and share *alike*. The cost of using GPL code is that you abide by that principle.

> "The GPL it wouldn't stop me from keeping a patch but it would stop me from re-selling that patched version to people, without sharing the code."

GPL does not restrict people from keeping their ideas or solutions to themselves, but should their solution be made available to others it does require that their source code, hence their ideas, are also made available.

The requirement to return code and hence ideas back to the community also serves the Samba project well in terms of input from businesses. When comparing the GPL to other less liberal licenses, one team member states:

> "I think the GPL … encourages people to come back into the community. Like with the BSD you can take a copy and leave and go do your own thing and that is not black and white. If Samba had been BSD licensed [a large technology manufacturer like my employer] probably would not have contributed their changes back ... whereas if the license requires it then [my employer] is like, 'well ok we have to do it."

Another developer commented that:

> "Samba exists because it's had people working on it for years and those contributions have been contributed back… Companies who would have had a propriety interest in the software have been very good OS citizens. They are happy to contribute back but I know they just wouldn't get around to it if they didn't have to."

And finally:

> "I would say that somebody taking Samba and adding something to it and then selling it is a bit of a negative experience because the culture is based around the GPL to some extent. … [I]t does come a lot from [the project leader] because he has made a couple of statements as to why he chose the GPL for

> Samba. Which is basically along the lines of; 'we have done this and we want people to share their changes as well'."

The GPL, with its stricter requirements ensuring the continual freedom of derivative works, encourages developers and companies who modify it for use in their products, to submit their changes back to the community. The normative principle of sharing can be evidenced in the adoption of specific code which consolidates, in a material form, the fundamental principle of the Samba community. In so doing, this norm becomes consolidated as a 'rule' within the technology.

3.1.3 Breaking the Rules

The power of community norms to promote collaboration can be appreciated on the (rare) occasions when an individual contravenes the norm. In the case following, the individual had contravened the norm and broken its material manifestation as rule, the GPL code. In this instance one (since expelled) Samba team member attempted to circumvent the GPL and make a commercial product. As one developer recounted:

> "I guess this guy thought he could make this decision to bypass the GPL so he could write a bit that he didn't have to share with people and he could sell that bit. That is the essence of it. And [the project leader] basically said no, 'we don't want to.' [The project leaders] idea was that we shouldn't make technical decisions for political reasons."

Whilst this developer described the issue in terms of a technical decision, this example highlights the strength of normative behaviour and practice within the community in relation to how a product should be used and the role of licensing in that use. As stated previously, such occurrences are rare. This indicates that the normative principle of 'share and share alike' is very strong within the Samba community. As Elliott and Scacchi (2003) point out, the recognition of strong norms within a community minimises conflicts and foster collaboration.

The strength of this norm and the degree to which members have been successfully socialised into the sharing culture of the Samba community can be seen in the way that developers explain why the principle of sharing is so important within the community. Rather than, for example, explaining the adoption of this principle with reference to 'community dictates' or an external imposition, developers showed that

this principle had become so 'naturalised' and engrained that it was seen by many as 'common sense' behaviour in terms of 'fairness':

> "Well ... it's more a sense that we have done something in Samba and if you build on it then you should not keep that to yourself. You have used our stuff and in exchange for using our stuff we want to use your stuff. I think that is just a fairness thing."

The idea of sharing with others and 'standing on the shoulders of giants' has become normative to the point that developers explain it in terms of a personal ethical code relating to 'fairness'. In other words, not only does this show that the norm is strong; it also shows that Samba developers have incorporated the community ethos of sharing to the extent that it constitutes part of their ethical position[16].

3.2 Norms of Communication: Assistance and Persistence

> "If you want to fix a bug, then you just need to report the bug or fix the bug [yourself]. If that is all you want to do then you are done. So it's kind of nice that way."
>
> *Samba team member*

Although this is the perception of one Samba developer, their experience of the process of asking for and receiving assistance cannot be taken as indicative. The quote above represents a simplified view of having software bugs fixed or fixing them yourself within OS software development communities. Success in gaining assistance for software problems is strongly linked to existing social norms regarding the sharing of information and contributing back to the community. Developers preferred to assist those who had shown a willingness to persist in solving the problem and to 'give back' to the community:

> "Some people stay around and contribute back, just answering basic questions. I enjoy working with those people…"

The same developer gave a detailed example of such an individual:

[16] This ethos can be found in interactions Samba team members and end users, most of which occurs on the public mailing lists where discussions are posted between individuals and seen by all.

> "I have very particular respect for [a systems administrator at] the BBC who has been extremely persistent with Samba and [the BBC] to the point of over a period of three months debugging an issue that was preventing Samba working at the BBC. He was willing to go the full nine yards to finding the actual issue. That I have incredible respect for him and I mailed him back to tell him basically that I usually object to users mailing me….but I told him, consider me your personal contact and if you are getting stuck somewhere just rattle my cage…I'm happy to help him out ... because of what he was willing to do to find that particular bug and because of the persistence he showed…I was flabbergasted when he was still there with the bug and still willing to work on it."

Displaying persistent effort on a problem and providing detailed feedback is rewarded in this case by the inclusion into this team member's identification of the Samba community. In contributing back to the community by debugging the software the user is enacting the community norms of 'share and share alike'.

To consider assistance a norm that operates throughout and between all levels of the community is an oversimplification of the process. It is also a limited view in that it does not consider other types of differentiation, such as that between end-users and developers:

> "Developer to developer [assistance] tends to happen a bit. It seems that developers tend to help other developers and users tend to help other users. Seems to be a bit of a divide there."

The Samba community does recognise a distinction between requests for assistance between end-users and the developers. This distinction can be seen in the practices involved in answering requests for assistance. This distinction is not absolute but it does rest on the idea of contributing back to the community in certain ways:

> "Yes there are norms in being a user requesting assistance, in showing what you've got is bizarre and … you have chased down the usual suspects".

An end user must firstly show that hey have a unique problem and secondly that hey have persisted in chasing the usual sources of the problem. The case of the systems administrator at the BBC is an extreme example of making sure both these areas are covered.

In the case of developer to developer, however, this normative practice shifts:

> "Developers can get away with more as developers are usually more to the point. They usually will not have gone to incredible lengths detailing every element of their problem but they know where it is and can start pulling out some details. The usual rule in OS development is that developers can get away with more then users because they are respected and I think other developers realize that they are short on time."

Between developers there is a recognition that their expertise involves firstly an understanding of the technical language and social norms required when communicating the problem and secondly that since a developer's time is limited, expectations of the quality of the information required as well as the tone of reply is treated accordingly[17].

Developers recognised that end users may have different expectations when seeking assistance:

> "Some people tend to think OS is about ... [giving others]...this replacement of windows for free and they tend to assume that even though they haven't paid for it… they should get supported as much as they want. So the expectation is that someone will hold their hand to get it all going. Which, I don't want to say it shouldn't happen, but…I think most people doing OS are not doing it because they want random people in the world to have software for free."

For the developers 'doing' (i.e. contributing to) the development on Samba, the goal is not necessarily to all support end-users.

Many end-users use the program without requiring direct assistance from developers, but those who do, must accede to the norm of sharing and a technical and social knowledge in order to effectively communicate within the Samba community. As one developer puts it:

> "I don't mind spending a reasonable amount of time on some of these user support things but I will enjoy spending the time on people I know are getting it and are contributing back… it's that they really understand it and are perhaps writing a bit of it up or answering a few questions. That adds to your willingness to work with someone."

[17] Curtness and 'getting straight to the point' are not an unusual tone seen between developers. This is particularly so on the mailing lists.

Another states:

> “I particularly enjoy working with experienced UNIX administrators that appear on the lists that understand what is going on and just need some assistance on the specifics. Where you can tell them what to do and they'll figure it out rather then command by command reference.”

Developers prefer to work with people who are ‘getting it’ and ‘really understand it’. “It” refers here to the knowledge being exchanged and the ability to apply that knowledge, in other words expertise. Thus there is a level of technical skill, as well as an understanding of the social norms of communication, that is required to be able to effectively engage with team members as an end-user.

There is also the potential for end-users to miss the fundamental responsibilities that come with being able to post requests for help:

> “There are social formalities if you know what I mean… communications in OS projects tend to be fairly frank and to the point but you don't want to kind of go too far past that and offend people…You can say; `so there is a bug in this here'. But you don't want to say; 'there is a really stupid bug in this terrible program', that is too far…For example it is…rude to say `there is a bug in this', and not give any details what will help the person fix it. But there are benefits and responsibilities that come with actually being able to report the bug to the person and get it fixed.”

Understanding the social responsibilities that come with being able to report bugs and have them addressed requires a recognition of the link between the existing code and those developers working on it. It is quite common for developers to ignore such requests as described above, or as is sometimes the case, challenge that end-user to do a better job.

> “So you will sometimes get people who are like; ‘this [program]’sucks. But, that is kind of hurtful. But it's also; ‘well I don't care what you think … you can make a better one if you want’.”

For an end user to receive help from a Samba developer they must firstly exhort all the usual errors. Secondly they must show that their problem is unusual and place it the mailing list with sufficient information. Thirdly they will receive support if they are seen to be contributing back to the community. Fourthly, they must recognise that

to attack the code or program is to attack the developers who have contributed; an attack on the code and generally ensures the requests are ignored. Finally, end-users will benefit greatly if they can demonstrate a willingness to explore a problem themselves and do some work in finding a solution.

3.2.1 Coding as a social experience

For those developers involved in the explicit development of Samba, the project often involves the direct exchange of ideas and solutions with others. This takes the form of social exchanges within the mailing lists and also on IRC. As a common practice, developer feedback is an expected part of creating engineering solutions:

> "In OS software because of the way the community is built if somebody has done a miserable job at something, you usually say it ... and usually there is review processes."

Any exchange of information, including the code review process, involves norms of communication that ensure constructive exchanges. All patches, unless the developer has direct access to the CVS, are peer reviewed by the member of the Samba team working on the area to which the patch relates. It is understandable that a great emphasis is placed on creating a constructive environment as Samba relies on the continual input of developers. For the developers sending in patches, it is suggested that this process of feedback is a social one steeped in the exchange of ideas and norms, rather then simply a submission of code to an impersonal CVS program. As one team member points out:

> "…it is a team effort; it's not just a project that we individuals contribute to."

The very social and team nature of such software development is exemplified by the fear and embarrassment of having ones' patch reviewed and possibly rejected by others in the team.

The risks associated with such a social feedback system is not lost on the project founder, or any other developer interviewed:

> "…you've not got to be too embarrassed. If you're too embarrassed about your code then you're not suited to being a free software programmer... and I think that embarrassment is one of the biggest impediments to people becoming free software developers. People are worried what other people will say about their

> code, they are worried of being laughed at. And the thing is people don't laugh at you. It's really disappointing when you hear people say it, you have to try and convince them, `well, every bit of code is bad, everyone is embarrassed about their code, but your code's not really going to improve unless you get some feedback and some ... usually constructive criticism. Because you'll usually find when you put out some code under a free software license that people are in fact delighted and even if it's terrible code they make constructive comments and they tend to try and help you."

There was a general emphasis that the contributions will receive constructive rather than destructive feedback, in keeping with the community culture of providing an environment conducive to positive interactions and reciprocal assistance.

Continual contribution increases the skills of developers. As one developer notes, this impacts on the potential sense of embarrassment:

> "I think over time I get less and less of that [embarrassment]. I know when I was first getting into [Samba] I would be enormously embarrassed if I made a mistake. I think as you get confident in your work and established that lessens..."

The connection between rejection and improvement in skills is explained by one developer as constructive:

> "I did my first bit of code which was ... rejected. At the time it certainly kicks you in the gut a bit but I went back and … I remember talking to [the project leader]. Yes, the approach I took the second time around was far superior. You try… to be constructive when you work with people, you know, human decency."

The team member affirms that rejection of ones' code can be disappointing. However, the social interaction with the project leader, amongst others, feeds into the ability to take the rejection of code as feedback and try a better approach. The emphasis here is on the type of supportive environment that ensures the ethos of 'constructive criticism' occurs, based on 'human decency'. Although good intentions are always espoused by Samba community members and they express a genuine wish for harmonious social relations, there are occasions when differences can not be reconciled and the conflict reaches a point where forking takes place.

3.3 The right to fork

> "There are lots of different reasons why people might fork and there is no particular one or one driving force."
>
> *Samba team member*

According to one developer,

> "...a fork is usually a very powerful thing in OS software. Linux Weekly News this week characterized it as a… much like a divorce, something not contemplated easily and a big decision to make but sometimes a decision has to be made[18]. It occurs when developers' feel that their project is going in entirely separate ways and ... when ... their differences are irreconcilable."

Forks represent a shift in priorities that can motivate developers to move outside the existing community and form another project based on that code; that is, to take that code in a different direction. It is not a decision that is easily made and it is argued here that this is in part due to the threat it poses to already established relationships within the community[19].

When developers describe forks they often classify them as either positive or negative:

> "There are good forks and bad forks. I guess it's more about the details of it. So if there is say a good objective reason to do it, then that can be fine.

Positive forks typically are seen as having a good technical (i.e. 'objective') reason for their occurrence. Bad forks, however:

> "[J]ust fragment [the community]. Typically, leading up to that there is going to be some kind of interpersonal situation that cannot just be worked out, so everyone is probably feeling bad already."

---

[18] See http://lwn.net/Articles/98482/ [Last Accessed 17th November 2004]

[19] As a potential conflict resolution method there are two types of forks, a traditional fork and a parallel fork. There can be different priorities within a project that are resolved by having a parallel fork: "Parallel forks [are] where two projects will continue and one might be, say the developer branch, and the other the stable branch. But they will see themselves as closely aligned but there will always be this clear distinction between this version of it which is maintained for the latest features [and the other stable version]. So sometimes this is within a software project were you know, we multiple versions of Samba" Not all forks thus involve separation of the community itself and can in fact be instigated by a project leader rather then other developers.

A bad fork on is often seen as unjustified in a technical sense as well as a potential waste of effort. Part of this waste of effort is related to the potential damage such behaviour can have on the existing community. In this environment, where differences cannot be resolved, a decision is usually made by a developer and his potential followers to fork a project. In doing so, however, a difference in technical reasoning or goals becomes mingled with personal issues.

> "Well a lot of the time it's got to do with a lot more personal things than it has to do with code differences although often the two are related."

The splitting of the community in any fashion is considered negative because it can be an acrimonious process that spills the technical into the personal and the personal into the technical. It is often an emotive process as it represents a fragmentation of the former community and the necessity to re-cast the community's form and technical and social relationships.

Despite the difficulties that forking represents to the cohesion of any community, it is an overarching concept central to OS (Joode 2004) software development:

> "It's often that the right to fork is highly cherished within the community and that is certainly true. One of the working definitions ... of a piece of free software is that you can fork it if you want too. You can take it down a different path ... and if you can't do that then it's really not free software."

3.3.1 The Social Dimension of Forking

Although structurally there are no limits to forking a project, to do so involves disengaging from established interpersonal bonds. Forking also requires an assessment that the developer(s) have enough social capital to enroll others in the breakaway group and to sustain interest in the new project. Members of Samba, however, are reluctant to join a forking:

> "I work with the expectations that other people have, so... if I wanted to do something then I could ... fork it, but like I said, I don't feel a need to do that."

Working with the expectations of others in the team and the remoteness of the possibility of forking can be seen in terms deciding the personal worth and meaning of the existing community:

> "To some extent, if you are part of the team, forking the software and going off on your own is kind of frowned upon because it is fragmenting the community. Like [a previous team member] went off with a couple of other guys, so there is this fragment of the community that got cut off. Personally I think it is often a waste of time because you need to maintain two bits of software that often need the same bug fixes. The other alternative is to maintain a patch separately."

The fragmenting of any community is a risky decision and ultimately can be a damaging one. The desire of one or more team members to take the project in a particular direction was meet with opposition in this case and that fork eventually became unviable.

3.4 Conclusion: Norms, Enculturation and Social Bonds

This chapter has shown the operation of 'sharing and sharing alike' is the dominant principle and guiding norm of the Samba community. The strength of this norm can be appreciated with reference to the GPL which, it was suggested, represents a material manifestation of the social norm of sharing knowledge in the form of code. The chapter has also illustrated how this principle has shaped the normative communication practices, specifically in relation to requesting and receiving assistance between developers themselves and end-users and developers. The evidence of successful socialisation into the Samba community can also be appreciated in the stories recounted by developers, who reported that when beginning to submit patches to the software they experienced both embarrassment and rejection. These feelings, however, diminished as they not only expanded their technical skills but also became accustomed to the submission of code as a standard, indeed, vital practice within the community. As they grew to feel more comfortable within Samba, and to identify with the community, their confidence grew to the point that submission of code no longer presents such a problem. These illustrative examples: the sharing of code as a strong community norm; the institutionalisation of that norm in the form of a technical apparatus (the GPL); the specific social 'codes' required to ensure response from developers, and the illustration of the relationship between enculturation into the Samba community and the degree of embarrassment experienced upon submitting code, all indicate that the Samba community is much

more than simply a number of individuals who submit code for a particular purpose. Samba has its own culture, and this culture must be learned for successful practice to take place.

The final section of the chapter, which discussed the right to fork, can be seen as an example of an attempt at ‘conflict resolution’, albeit not a particularly successful one. The right to fork is a shared understanding within Samba. Developers emphasised that forks are not so much about the splitting a project; they discussed it in terms of the social relations which are harmed by such an event. For the instigator of the fork, the stakes are also high. He must make a judgment as to whether or not his social capital within the community is significant enough to enroll others in his new project. The metaphor of ‘divorce’ was used when discussing the issue of forking. The use of such a metaphor implies that strong social bonds exist between community members, and that the severance of these social bonds will not be easily managed, nor quickly forgotten.

Chapter 4 Samba as an Open Source Community: Politics and Business

So far this thesis has argued that developers continue their involvement in the Samba community because the social interactions and relationships are strong within that community, and are perceived as having equal significance with the pleasure of submitting code. Their identity within that community has been differently constituted as a result of practicing and interacting within that community. This chapter discusses developers' perspectives on broader issues which are related to OS communities in general. In so doing, it may be possible to ascertain whether or not Samba developers have similarities in their perspectives. This is of interest, in that similarities in political perspectives may also be a factor in explaining why such strong identification with the Samba community takes place. At the same time, by investigating the political and ethical positions of Samba developers, it is possible to gain insight which may assist the initial analysis.

4.1 A Spectrum of Beliefs

The two broad issues discussed in this chapter are pertinent to OS development projects in general. Firstly, the ongoing debate as to whether or not the initial principles of OS software have be n 'corrupted' or thwarted (Himanen 2001), or has the 'encroachment' of industry involvement been perceived as an acceptable, indeed welcome development (Weber 2000; Hippel 2001)? Secondly, how do OS developers now conceptualise their activities in light of partial industry sponsorship; do developers perceive that the OS phenomenon has altered, and if so, what are the consequences for them in light of how they now see their own activities in OS?

As mentioned prior in the thesis, Stallman ideological reasoning has been seen by many to be a core shared value within the OS community. For Stallman information wants to be free and he sees anything less then that notion being fostered within the development of software as unacceptable.

> "A lot of people [are] involved in the development because they... some are form the technical and some are from the political... sort of more moral aspect of free software because software really shouldn't be sold. [T]here are a lot of

people and you'll find that there are people on different parts of that spectrum."

The spectrum referred to here ranges from a pragmatic argument for doing OS software development, where. This argument sees the current phenomenon of OS software, with partnerships in varying forms being made with the IT industry, as a reasoned response to solving software problems and that yields better software, a point which is itself contested by many. The opposite in this spectrum consists of the view, that regardless of the technical benefits, OS software is seen as morally and ethically better then proprietary software. In other words, some members hold the view that partnerships with industry violate the principles of free software, whilst others 'just want to write code'.

Developers within the Samba team differ in their approaches to this issue. Part of this is due to the ideology at work from the early days of computing and in particular the work of Richard Stallman's FSF and GNU project. As OS has grown in size and application, the political field has also diversified:

> "I guess OS has a larger political dimension to it then compared to someone who just goes a writes Visual Basic code[20]. There is a bit of an ideology behind it. Different people seem to pick it up or do things with it more then others. There is a difference between Stallman's 'no passwords' type of freedom on anything and Linus, who is still committed to the freedom of the thing they are working on but not as extreme."

For the project founder, Stallman's views, as expressed in his GNU manifesto, are considered with some caution.

> "There are many aspects of the GNU manifesto / GNU philosophy that I would agree with and that I would think are important, but I do think free software is important to society and I… would rather see the world moving back to a more free software approach to the software industry then the way it currently is, very much propriety orientated… I do see free software as being more then just a better way of making software. It's not just the pragmatic, 'it produces better software'. When I'm arguing to business people I often argue

[20] Writing visual basic code is a reference to a simple method of programming not often associated with large scale OS projects.

> from the lines... the classic open source arguments of `it's a better way for producing software, for producing better quality software etcetera'. But even if that wasn't the case, I think it would still be justified. I think it is also ethically and philosophically better then proprietary software. But I'm of the school that doesn't try and force that on anyone. I'm of the school that it should be a programmer's choice as to how they license and release their software."

In identifying with the worth of OS software development, the Samba project leader articulates a moral perspective at the same time that he emphasises the individual freedom of the developer to choose how they release *their* software. A significant question at this point is to make here is whether or not there is any correlation or pattern between where developers place themselves on the spectrum (from ideological to 'doing software')and their identity as OS developers.

One Samba team member recalled an experience at a conference whereby a guest speaker began to praise the work of OS software developers:

> "He was like, `I think it's great what you people are doing to defeat Microsoft etcetera etc etc.' Which I think many of the people in the room were just like; 'Huh? That's not what we are trying to do'. Like Linus says, defeating Microsoft would be an unintentional side effect... the point here is to do good software. So from that point of view it's not political. But, I think free speech in important and free markets are important. I think having the whole of people's digital world and internet controlled by just one company would be really unhealthy."

For some developers there is a separation between the reasons for doing OS and the role that any ethical views play in that decision. 'The troops are not rallying to destroy the empire', in this case often viewed as Microsoft. Rather there is an emphasis on the benign developer who simply wants to develop good software in a way that is beneficial to themselves and other developers. As one Samba team member states:

> "To some extent I like the idea of things being open and free but also there is the more realistic view that companies need to make money and so forth. I'd say that OS has potential to produce better software and it's useful for users and so on, rather then the evangelical side of it as you put it."

Samba also has the added dimension of directly entering into Microsoft's operating space and methods of making money.

> "With Samba there is always this political thing to do with Microsoft because we are creating something that is in competition with Microsoft so...in one sense we are costing them money in lost sales. You know it's 'us' verse 'them' type of thing which I guess is counter productive in some respects."

When asked if this developer identified with that 'us' and 'them' view, he stated:

> "Oh a little bit. There is kind of the whole underdog thing going which is kind of interesting. That is one extreme. The other is we just want to write something useful that people can do things that they cannot with windows."

In having software that interacts and competes with the sometimes maligned (with the OS community) Microsoft creates an expanded political dimension. The potential to see Samba as in direct competition to Microsoft is interesting because it contradicts the 'unintentional side effect principle' that Linus mentions. Perhaps this contradiction is more subtle as Samba is seen by many team members as simply giving people the option to not depend on such propriety systems with restrictive licensing agreements.

> "...and I honesty believe that Samba is a force that provides a check on Microsoft monopoly. Samba leverages Linux into networks in a way that no other piece of software does... I find it scary that [people] are off paying licensing fees in orders of thousands of dollars…just to get basic work done. What I believe is that in doing something useful to others, I find that very important. It's a interesting problem to solve but its a interesting problem to solve that assist others and if what I'm doing means... people have a choice in how they build their network, flexibility in how they build their network, then I think that is a good thing."

Samba allows you, the user, to do things that by using just Microsoft could be potentially costly.

The view of epistemic communities as having shared values in is here problematic. Edwards (Edwards 2001) states that the common policy enterprise in OS generally emphasises the opposition to closed source commercial software (2001:16-17). As shown above this is not a universally held belief and as such this view is simplistic and unrepresentative of the complexities that exist with OS communities such as Samba. Considering developers are found to reside of different locations of this

spectrum, it is a worthwhile question to ask how such a diverse group can work in a functional and consistent fashion?

> "Provided you can tolerate people and their views and things, some people can be in it for their political views... others are involved in it for a moral standpoint; they don't believe that software should be allowed to be closed. Others are involved in it for practical reasons. If we can get along with each other for long enough and we can get the things done, it doesn't really matter why people are involved."

The ability to get along does rely on a shared understand about the goals of the project to produce a windows compatible server. The idea of doing Samba for both or either technical and ethical reasons is a relative debate as provided there exists a willingness to work and share knowledge together, the reasons behind it, whether political or not become not so significant as the likes of Stallman (1999) and Raymond (2001) would debate about.

4.2 The meaning of work

It has been emphasised in this thesis and by developers thus far that everyone in the Samba team is doing this work as volunteers. The implicit understanding that people are under no obligation to participate in the project directly shapes expectations, community structures, norms and values. Increasingly in the OS world, the interjection of cooperate companies has come to the fore (Franck and Jungwirth 2002), with some members of the Samba team being directly compensated for their work with and knowledge of Samba. Lerner and Tirole (2002) state that the "key to a successful leadership is the programmers' trust in the leadership: that is, they must believe that the leader's objectives are sufficiently congruent with theirs and not polluted by ego-driven, commercial, or political biases" (2002:222). Considering four number of team members interviewed are or at some stage were a directly compensated, for work done on Samba, the potential disruption this shift to paid work could cause within the project is worthy of exploration.

4.2.1 Workings Verses Volunteering

The first interesting distinction that developers make regarding the work they do within the Samba community is whether they consider it as a working or volunteering

activity. All developers interviewed bar one where, at some stage, paid by a company to do Samba related activities. For some the difference between both ideas rests on the concept of choice:

> "I consider what I do for the community to be volunteering because I choose to do it."

The idea of working is thus tied to the concept of freedom. Volunteering as a concept places more emphasis on the rights of the individual to be involved or not. Yet for one developer the concept of volunteering in itself is problematic:

> "I don't know; volunteering doesn't seem like the right word. It's more like there is this project I'm trying to do ... like how some people are writing a novel every morning or evening ... and they're not getting paid for it and they might never get paid for it but it's this thing they want to do."

Picking the right word is difficult because at the heart of this project is developers who are enjoying what they do. In wanting to do it, the boundary between what is paid work and what is non-paid becomes blurry for some:

> "A bit of both [volunteering and working]…and I'm pretty sure the number of hours that adds up to is awful lot more then is in my contract. So ... you could say I'm volunteering for most of the time. So it's both my hobby and my job. And I don't firmly distinguish between the two."

Yet for others, in doing a bit of both, it is obvious which parts are work and which are volunteering:

> "Yeah I guess a bit of both really. I guess I managed to compartmentalize it, 'this stuff is work and this stuff is [volunteering]... and generally they didn't mix, well in my mind they didn't mix."

This ultimately leads to any category of defining the meaning of work, that of paid verses non-paid.

4.2.2 Paid verses Non-paid and the role of 'sponsorship'

The idea of receiving compensation for work performed as a Samba team member could be seen as providing a conflict to the freedom of choosing what to do and for which reasons. Considering the project leader is an employee of one of the world's largest information technology service providers, there could well be seen a conflict between the independence of such projects and the freedom of a developer to choose

to work on such projects. The question is what does employing a team member or project leader buy the company.

This potential conflict is resolved by the project leader in the following ways, and is significant enough to warrant an extended recounting:

> “When you’re a free software developer and you’re paid to work on free software by a company, notionally what that buys the company is… the right to set your priorities so that. [C]ompanies I’ve worked for, they can't say; ‘you will put the following feature into Samba’… because I wear two hats. I wear a hat as a member of the Samba team and I wear a hat as an employee of that company. And if they say you must put the following feature in, then I have to evaluate that before I accept the feature in Samba. I have to evaluate it with my Samba team hat on and say, ‘is it good enough’. But at the same time we have a whole long list of things we want to do with Samba. If I’m being paid to work on the project then the company that is paying me can set the priorities. They can say; `well there is a great long list of thing you want to work on, we'd like you to work on this first, we'd like you to prioritize the particular things that would benefit our business'. And that has often happened [but] that doesn't mean I stop work completely on other features, but it just means the emphasis just tends to be on the things that the company, depending on who I’m working for, is interested in. There might be a quick fix and a correct fix and so often I might create the quick fix for the company wearing my company hat and wearing my Samba team hat I reject my own code. So I’ve got a number of patches, actually I collect them in a particular spot on my [web] site. The patches are still made public, anyone can download them and use them if they want to, but wearing my Samba team hat I've rejected dozens of bits of code I've written myself because I had different priorities when I wrote that code. And with my Samba team hat on I evaluate that code and say, ‘no, it doesn't meet the standards of a good, long term, solution or the maintenance cost of it is too high’. So I reject it. Then sometime later I might solve the problem in a different fashion, in a way more acceptable to my Samba team hat wearing side of my personality if you like <laughs>… and in which case I accept it.”

So the project leader has to consider two interests that at times may conflict. What having an employee who has commit access to projects CVS really does is buy the company the ability to set priorities but certainly does not buy them control over the project. Indeed, even as a project leader, the decisions made here are for the long term benefit of the 'project' rather then the company paying that team leader. It is quite a novel solution that when instructed to alter or amend Samba in a particular way, this project leader in acts a split personality that will make the changes public so they are on the record for all to see, yet reject that very same code from the final release (i.e. not commit it to the CVS).

At this point it is worth considering an alternate term that sums up the relationship between working for a company and the rights it gives them with the work an OS developer.

> "The fact I'm being paid at the moment is auxiliary to [my contribution to Samba]. It pays the bills. It's savings I can keep for when I'm not paid to do it, which will happen. But it's not why I do it and I spend many more hours on it then I'm paid to do. Certainly at times of holidays and things I can spend a lot of time on it... I have been paid by various companies to work on Samba. They are effectively my *corporate sponsors*; they don't tell me what to do anything more then some very general directions they are willing to pay for. Whether I get corporate sponsorship for any particular piece of work really just means money in the bank. I don't change what I work on particularly…and the stuff I'm working on is pretty much the stuff I was going to work on anyway. It is focused around their particular needs set and because they are paying me a s***-load to be doing it. But everything I do is given directly to the community."

The key word here is corporate sponsors. Sponsorship implies the provision of funds to do a certain task with a looser emphasis on ownership of ideas or persons. Once again the corporate sponsors can set the general direction and priorities of their employee but it does not translate into a dictation of where the Samba project will head. Also the primary role of sharing and sharing alike is evoked again with 'everything being given directly back to the community'. Just as any alterations done by the project leader for his sponsor are made public but not necessarily committed to

the CVS, the work this team member does for his sponsor may have been motivated by the companies own need yet it is still made public.

4.3 Conclusion

A spectrum of positions and perspectives was found when developers were asked to comment of the issue of OS and its apparent movement away from its original strong ideological position. This suggests that Edwards view of epistemic communities, in which he proposed that knowledge building communities, in which there are strong shared values and norms, are more likely to have similar positions in relation to broad but pertinent issues, is not appropriate in this regard. Developers had differing opinions as to the relevance of the debates. The developers were well aware of the political writings and views of others; however they most often shied away from such debates. They did eventually admit to recognising the political scope the work they are involved in but clearly emphasised that the technical goals of the project and separate those goals from any political, moral or ethical reconsideration relating to the 'Free Software' debate. The day to day acts of developing Samba are not permeated with such issues; everyday practices are, rather, focused on the goals of project itself.

The meaning of the work done is an exciting area particularly considering the view of OS communities as volunteer communities in which there exists no direct obligation for developers to do anything. This area of OS research is under researched. The impact of the increased corporate 'sponsorship' of projects that industry is showing has implications for employment of team members or project leaders. The preliminary findings here suggest the ongoing autonomy of the project ensured by the 'share and share alike' principle and the transparency of the development process. It is found that the role of volunteering or working itself is not well defined by developers. It is anticipated that future research will draw a more detailed and perhaps more complex assessment of this increasing phenomena as the 'corperatisation of OS' grows.

This chapter has examined the perspectives of developers in relation to the alteration of the initial principles upon which OS software was developed and found that although the Samba community has strong and accepted common principles and norms, there was quite a diversity of perspectives on this topic. The fact that members

of the Samba community cannot be said to be ‘like minded’ testifies to the argument that it is the social experience as well as the learning of technical skills which form the context in which identification with the Samba community takes place.

Chapter 5 Community, Code and Identity

This chapter examines the relationship between developers' participation in the Samba community and their constitution of identity. As outlined previously, the majority of literature which seeks to address the issue of why participants in OS communities continue to participate after their initial individual need had been met has focused on individual motivations, whether extrinsic or intrinsic. Edwards (2004) puts it plainly: OS software developers, he argues:

> "are motivated by need and interest in solving the problem, and far less by socialising. And here we touch the heart of [the] problem. Persons wanting to contribute can not be expected to share the same mindset as the community" (2004:18).

As with the research mentioned above, where individual motivations are the key conceptual mode of understanding, Edwards use of 'mindset' implies too that the' self' is a somewhat static phenomena within which essential qualities are either present or absent. Rather than conceptualising the issue in terms of whether or not there was individual compatibility to begin with, or whether or not the self is 'driven' by internal or external motivations, this thesis suggests that in the process of fulfilling an initial need the budding developer engages in community practices which influence a refashioning of identity. This chapter now reports on the developers' experience of social relations within the community so as to analyse this question in terms of how one's identification with both the Samba community as well as the product, both of which are social processes, may explain developers' ongoing participation.

This chapter will explore the impact of the Samba project on developer's lives and highlight that, as a social entity, the Samba project is filled with social beings who create bonds that extend beyond the specific aims of the project. Evidence of the power of these bonds is explored in relation to the seldom discussed issue in the literature, that of leaving or becoming inactive in a project. The limits of a communities of practice framework is shown through the difficultly experienced in leaving such communities.

5.1 Identification with the Samba community

"Samba and the Samba team is a strong part of what I do."

Samba team member

As developers' become increasingly involved in the Samba project the impact of the community in shaping its members and in turn being shaped by members becomes more pronounced. In this way Samba as a product and Samba as an OS community, becomes part of the self that is expressed through the OS development process.

> "I mean it certainly is a social thing [in] managing and contributing to what is going on."

This 'social thing' refers to the interrelationships formed within the community as work is done on the project. This community of practice requires contributing to the community in terms of the community's norms and values and this creates bonds between its members. This all points to the co-construction of the self as a Samba team member and more broadly an OS software developer.

For some developers, the idea of community is limited to those who can contribute to the project. In this way membership extents to both those who are members of the Samba team as others who are not on the team yet contribute in some fashion. Thus identification with the Samba community can be seen as a 'practice':

> "It's like being a member of club or something like that... with people of a similar interest. Like I'm a member of triathlon club and we regularly meet to train and that kind of thing and I guess in doing Samba stuff, it's not like you are by yourself. You are always interacting with other people who are contributing for the same reason, either because they enjoy it or [its] part of work ... it is more of a social experience ... in the same way as going to some kind of meeting or meeting a group of people to share a common task or common interest… If you had ten random people who just submitted code it wouldn't be as much of... a social experience."

The analogy with club membership it apt in that, members are participating or practicing with others in order to achieve a common task. For a club to function is requires the participation of others in an enjoyable social experience; a point affirmed by another developer who states:

> "I enjoy working in those communities and I have extremely good relationships with those people I work online with Samba."

The experience of bonds or ties with other members of the community is not equally strong for every team member. One team member likened working in the Samba community to a 'community of practice' where ties are not as strong:

> "…you have a place where you have built up you reputation and your knowledge. A lot of the sense of community is attached to the knowledge of the area... It is a lot more [a] community of practice ... I mean I will always give them a hand but as much as I would any other bunch of people I know and trust."

Participating not only occurs within the norms of the club but also exerts influence of how developers act and view themselves as club members. The community of practice builds reputation and knowledge but in order to know and trust someone it is argued in this thesis that some form of personal bonds need to be created between developers.

As shown by Lee and Davis's (2003) earlier study of Samba as the "community members *identify* themselves with the development of Open Source software…[and]…this identity encourages unity and collaboration on information and expertise" (Lee and Davis 2003:46). The norms and values which encourage unity and foster collaboration have been explored in chapter four, but it is important to emphasis here that these norms and values only become operationalised when there is identification with the community.

But what is meant by 'the community'? There is a universal recognition that there exists a social dimension to practicing Samba, but the impact of this on the self is placed within a wider community then simply Samba.

> "[Samba] is still very much a project I identify with closely of course… I sort of see myself as a free software developer….There are lots of categories I could fall into, but within the IT community, I see myself as a free software developer."

This identification as a member of the Samba team member was universally related in more general terms to identification with…

> "…open source broadly I guess. Pretty much, it would be in the top handful of things that identify me. [The Samba team] are generally nice people, smart people and good to hang out with. I guess people like being in groups with people they like."

Whereas the concept of communities of practice does not address the issues of interpersonal bonds, this thesis found that the developers strongly articulated that such personal bonds and friendships are significant in accounting there ongoing participation in the community.

For those team members who have stopped practicing, the degree of identification may begin to weaken but not to the point that a previous practitioner does not identify with the Samba community.

> "I feel a bit more on the outside cause I haven't been working on [Samba] for a couple of years ... I'm still on the Samba team list[21] so I see all the philosophical discussions that go by. To some extent I still feel part of the Samba community."

So even when inactive in the project there often remains a sentiment of being a part of the community. If anything this highlights the strength that such a community can have on individual developers, so much so that after they having finished practices development within the community they still consider themselves part of it.

But this is not a place where people only create social bonds, it is also a site for product development through a community. As such, when developers identify with the community it also imparts on their actions within it.

> "Does [Samba the community] influence my actions? Well obviously, you know, when I write code I put it out into a code control system that's available…to the community and I discuss the development of the software with other members of the community. So everything I do around Samba I do in conjunction with the rest of the community."

All direct contributions to the community are open to other members and as such the development process involves an inter-relationship between the social exchange of ideas and knowledge with the constitution of identity.

5.1.1 Identification with the Product

[21] The Samba team list [has] more discussions on the long term issues of Samba and copyright issues or that sort of thing rather then just technical things. Longer term internal things are on the team list which is a private list [for Samba team members].

To be a developer involves engagement in an activity of ‘building up’, in this case the construction of software. Developers are active in the community construction of not only of a software product by also of their identities. They become artisans of software and identity.

> “I think that to some extent when people ask me to describe myself, one of the things I would list would be being a OS author and so I think Samba and [another OS project] have made me identify myself as an OS author as a hobby lets say.”
>
> “So [Samba and my other project] I'm most active on at the moment; I really see them as part of who I am. The same way as a bit of work you identify with a lot. I guess I'd come back to the idea of people who write a book, they really, you know, it's a part of their self, their extended self.”

In this way the experience of the Samba community and the Samba product enter part of the understanding of their self identity. The difference between writing a book and producing OS software is that OS software development is a far less solitary enterprise. The community frames the way that self-identity is constituted. In other words, the collaborative effort involved in the production of public licensed software creates a situation in which members come to identify with both the community and the product itself

This is illustrated by one developer’s analogy with making furniture:

> See those two sets of book shelves over in the lounge room ... Both look fairly similar. One of them [my wife] and I built ourselves… and we lacquered and sanded it ourselves and put all the effort into it ourselves. The other one we went and bought from a store. One of them means a lot more to us then the other. They are both about the same. As objects they are very similar. The one that we just bought from the store is probably better built then ours... in fact considerably better built then the one we built ourselves, but if we were running out of space and had to throw one of them out you can be sure it would be the commercial one, not the one we put all of the effort into ourselves. We identify more with the one that we have put more effort into. The same with a lot of the photos we have got up on our walls. Ones we have taken ourselves, even though they may not be as good as ones we could buy, we identify more with them, because we took them ourselves. A lot of the

> paintings come from my mother-in-law and they have more meaning to us because they are [done] by a relative. So it isn't necessarily stuff you do yourself, but if you know the person who did it, again it has more meaning too you. That's not just common in software; it is in almost any field of endeavour.

To identify with something you put effort into is indeed a universal trait in fields of endeavour. Another team member pointed to the difference in persona meaning between making a cabinet and buying a replica:

> "So I think that is why the people who contribute to it see there involvement as different from those who just use it. Like the person who made the original cabinet, the level of attachment to it is much greater then people who have replicas"

The cabinet replicas refer to the copies of software that anyone can be downloaded by end-users. Working on projects such as Samba encourages and fosters a deeper identification with the product itself also allows developers to obtain a deeper sense of identity in the product itself. This ability comes from being involved in a community whose product is commons based. The significance of social relations in this personal and communal identification process can be seen when developers spoke of the role the CIFS conferences[22] in relation their identification with community.

5.1.2 CIFS Conferences

Developers where asked to describe an aspect which for them embodied the experience of being a Samba team member. The majority of developers stated that attending the regular CIFS conferences was extremely significant for them. This conference provides the one chance where all the team members can meet in person and discuss Samba.

> "A couple of weeks after I joined the team I was off to Seattle for the first SIFS conference I went to and that gives you are strong sense of being in the team. The team pays you to go over to a conference[23]. That, I always felt was to be part of the team."

[22] These conferences are year events whereby people come together to discuss work related to the computer protocol upon which Samba is based.

[23] As one developer notes: "Well Samba has collected some money from donations and we do sponsor people to travel to the SIFS conference and that. We don't just give our money to anyone; people

And for other:

> "One of the interesting things I did, I went to the CIFS conference in 2000. So it was good to go the Silicon Valley and meet people and so on."

The interaction with other team members in a face to face environment is a social encounter that imparts a strong sense of team membership; it also can solidify the importance of one's place on the team.

> "I turn up to a CIFS conference and give a one and a half hour tutorial about CIFS authentication."

Conferences also prove that social contact is helpful to encourage on development.

> "CVS commits…in the last few days have sky-rocketed because people are working after the conference…. it energizes [people]. At the conferences you have everybody full time for a week, by full time we mean 9am in the morning to 11pm at night, working on Samba. You've got the team energy about what we have talked about, enthusiasm that has been built up and you get... both of those. [The] project that occasionally runs statistics over the CVS tree and there is a consistent pattern that at each conference commits just go through the roof because everyone is working at the conference. They are not dragged onto it by their companies; this is time purely for Samba."

The kind of intensity described by this team member highlights the role of the social environment in producing this type of software. Intense social-work situations energize people to voluntarily commit to the project.

By knowing others within the community in this way, Samba team members are able to deeply identify with the product as both their own work and the work of others. In this way OS developer is unlike almost any field of endeavour such as working on a project in a private firm, because it takes place in a virtual commons where work is linked at once to the individual and the community at large, that is something beyond 'the self'. In this way most team members see their identity linked to Samba as both a community and an end product. One way in which this link is brought out into the light is the developers' view of the community as acting as the developers extended conscience.

---

within the team can get some expenses paid if they need something like a bit of software or a computer. We have paid for a computer for one the developers in Europe, in Poland or so."

5.2 Community as conscience

> "I'm a great procrastinator and I think a lot of Free Software [OS] authors' start off as procrastinators."
>
> *Samba team member*

OS software developers are the first to admit they can be great procrastinators at times. The ability to do anything but the task in front of them is a skill that seems often universal as much as there is an ability for them to persevere and work through complex problems. Part of this tends to flow from the boredom or burnout that can happen when focusing on one problem or task for a extend length of time. Many seek other side projects, yet others may choose to go work on other parts of the same software, returning to the problem later. Such is the freedom afforded to an OS developer. As a counter to this trait, the community is seen by many to also act as a conscience in the development of software.

5.2.1. Checking Yourself

To have a community as a conscience is an interesting idea, one born out by developers themselves:

> "I like the fact that my code will eventually be out there. It encourages me to do stuff like document and keep my code tidy...if I know it is going to be OS and other people are going to use it… It's like a self checking mechanism; it makes sure I do things in a sensible way cause then other people are going to look at it and check."

And more so:

> "I think the main role of the community is to serve as your conscience, to me anyway. It is certainly a good way of thinking about it, [F]or me that is an important thing. I think it reflects on you as a person if you submit something really stupid or half-baked."

The fact that code is shared in a public commons encourages developers to make sure that their procrastination does not lead to poor coding solutions. It operates as an effective self-checking mechanism because of the connection between a developer's code and how others view that person because of their code. The point emphasised

here is that although developers stated it was in their personal interest to create good code they also strongly emphasize their feelings of responsibility to others in the community. This aspect can be related to the discussion of the strength of community norms and values as well as, for many, the successful socialisation into the Samba community.

The idea of having responsibility to others demonstrates the strength of community ties, especially when most work is unpaid and people are under no direct obligation:

> "Well in practice there is the responsibility I feel to get a release out the door and X number of months or whatever. But if I don't someone else will take it and release something. It's mainly a responsibility to people who use it but I'd feel bad if I let the project slip. Like at the moment there are a fair number of changes in the CVS that haven't made it into release"

Another developer stated that:

> "There is a sense of responsibly of being part of that community is that you ask other people about decisions… Obviously I make decisions on my own. There are some things I feel I need to consult with the community."

Yet in practice the obligation comes from the nexus of individual work, collective engineering and community. To have a responsibility to something requires some identification with the object. The object here is both the community (other developers who have contributed back) and the product, and those who finally use it. This harkens back to the very social nature of these development communities and the personal bonds they foster. In contributing to a community and product with which you identify there is a distinct disincentive to commit poor work as poor work will reflect poorly not only on the individual but also on the how individuals view their relationship with others in the community. These relationships fostered by the OS community are not limited to the practice of sharing code or ideas, but have significant impact on the life course of developers.

5.3 Turning code to life: between Hacking Binges

> "All of my experiences are part of a Samba community, or at least the broader free software community."
>
> *Samba team member*

For many Samba team members the impact of working on Samba and becoming part of the team cannot be underestimated. This is particularly so when their own life course is placed within the context of OS software development.

> "My whole life has been dominated, for the past, 14 or 15 years with things related to free software. So I would have been living quite a different life... Nearly all of my friends are people from the free software community. Nearly all of my travel is to conferences related to the free software community. All of my income for the past 15 years or so has been in some way connected to the software I've written...[my life is] all related to the free software world."

To say that your life is dominated by free software is perhaps a little misleading as it implies no life outside the realm of OS. Rather one may say that the worlds of code and social life become blurred, and the life of programming influences factors in life 'outside':

> "It's been a big part of my life for a long time personally and professionally…"

Developers readily indicate how participation in the Samba community has shaped their life course and easily point to the impact such participation has beyond the Samba project itself:

> "I wouldn't be living in Canberra without [Samba] and I ... indirectly met my girlfriend through it. So those to things ...are kind of accidental but they shape my life quite a lot."

It is also useful to consider the way that Samba and similar projects are situated within the long-term life of a developer.

> "Also [levels of contribution] change over your life. I live with my girlfriend now so that ... cuts down on that hacking time. So conversely, [a well known

developer's] girlfriend is out of town at the moment so he is having a great kind of hacking binge while she's away."

The ability to juggle time for software development around the ebb and flow of the rest of life is not necessarily unique to OS communities. The point clearly made, however, is that such juggling is understood within the community as normative behaviour.

5.4 The problem of leaving

> I got out of very actively coding on it, I'm kind of one of the supporting players rather then the main thing...
>
> *Samba team member*

Having established the ways in which a budding developer may become a member of the Samba community, the question now raised is: how and why do developers leave the community? The thesis has established that the social networks and practices within Samba are conducive to socialisation, and can account for developers' continual appreciation. It is pertinent therefore, to investigate why and how team members leave Samba.

In comparing working on proprietary software for a company to working on a project such as Samba, a team members notes that;

> "…with a proprietary license you kind of put this big wall around people who at their company who have signed their [Non Disclosure Agreement]….so you can't have people drift into it so easily as you can with Samba."

It is true that Samba's lack of formalised work arrangements does give individual developers a large scope to drift into the project without taking major steps. However, one area not often discussed is the act of leaving a project. Is the leaving of Samba as casual as the entry? When the same developer again compares the two:

> "[It is] not a big step between me being in or outside the community. If I wanted to work somewhere other then [my current workplace] then it's a binary step. If you resign you might still have friends here but you are not part of that [work] community really. Whereas, if I scale back my involvement

> with any one project [in Samba] I haven't really lost anything I'm just not doing [as] much for them."

But how is it possible to stop working on Samba and still be part of the community? At a formal level, team members can be asked to remain part of Samba, but such an 'emeritus' status is symbolic[24].

As one developer notes, it is ok to fall back from a project, to withdrawal from active involvement in a project like Samba.

> "I think that I still see myself as part of that community if I [scale back]. I mean it is all so fluid"

And other developer states:

> "Samba doesn't come with a set of responsibilities like you must do the following sets of things. You know, people do become inactive and that is not seen as some dereliction of duty or something. People become inactive and they eventually come forth and say; `yeah I wish to retire' and they get put on the Samba team alumni."

Yet in the changing flows of life and the levels to which a team member is directly involved with the Samba project, it is still recognised as a big change to suddenly disappear 'off the radar'.

> "You might move between projects at some rate, but you are still within this thing and that would be a big…. to drop out of it entirely... you know that would be a big change. …maybe it's not on the scale of breaking up with your girlfriend or spouse or something but."

Many Samba team members have gone on to create or take on their own OS projects yet remained part of the team and in some ways that community.

> "It's like the way people change between different groups of friends. You don't just suddenly drop all your friends but... you know if you used to play sport with one group or go the pub with one group you might just go to the pub less and less over time cause you are doing something else. I think that is the way people can gradually drop out of contributing to something."

Leaving Samba then, does not mean that one leave the OS development world. Most who have left Samba have continued to participate in OS development. To leave

[24] Some ex-Samba members have choosen to keep their @samba.org email address. They are also listed on the team page as alumni.

Samba abruptly is difficult because of all t he social relationship that have been forged. An individuals' movement from one project to another is a common route for those who leave, but the sudden disappearance of a team member without explanation is almost unheard of.

There are subtle differences exhibited in the level to which a developer will identify with the project, community and thus the strength of the relationships formed. As one team member explains:

> "If you are more a social type of person you would value more those parts of your contribution or your relationships to other members of the team. I guess [some people] feel more of a connection to the technical kind of contribution then the social side of things. I'd like to think I could live without Samba! …its, in the same sense that, whether you're a member of a football club...I guess it depends on your level of involvement and how passionate you are is how much of it is linked to you.

This illustrates the fact that developers are not individual islands that commit code to a central repository. They are social entities involved in a social process of developing OS software. The depth of involvement and social bonds varies, and depends upon on the level to which each individual developer invests in that aspect of OS software development. As outlined, strong social dimensions which are vital to Samba's development is extremely conducive to constituting ones identity in relation to the Samba community.

Evidence of those bonds and the relationship between community and self can be seen also long after a developer has stopped being active on the project. The limitation of viewing the Samba project as a community of practice is that such a theory whilst accounting for shifts in the structure of the community as people stop directly contributing to the project, it also implies that membership to the community is revoked at the same time. The connection between stopping practice and no longer being considered a member of a community is not binary. A good portion of this is found in the type of bonds that are formed and the relationship between how developers view themselves as members of both the Samba and OS community more broadly.

To conceive of life without Samba usually involves the imaging of another OS community:

> "I often wonder what would happen if Samba went away? I'd probably find something else to do, something else OS to work on I guess as a kind of hobby thing. Well I guess there are friendships things you make that are beyond the project itself which are always important and I guess you do meet the same people in different projects and contexts. Like you'll be trying out a bit of software and you will see someone else's name and go 'oh I used to work with them'. I guess people on projects are tied together but they are not...you know you can quite easily cross the boundaries into other projects and areas…. It is comes into the meritocracy thing again also. You know, 'hey this guy used to work on Samba' and you go 'oh ok Samba' and maybe that affects they way people can move. But I don't think hackers see that mobility as an end, it's more a means to expressing yourself or you know making cool code. It's expressive for me...it's different for different people, but for me it's about the quality of simplicity and elegance."

There are bonds that go beyond the project but it is the project that fosters the environment for those bonds to form. The ability to easily move between projects is based in part on social capital yet this mobility is not an express goal of developers. They are not, as much as Bauman (2001) would have us believe, in the business of ensuring they can more to the next peg community or in this case OS project. If anything this community perhaps offers the ability to be both secure and mobile in the stage of Liquid modernity:

> "I guess the jobs come and go but the people you meet generally tend to stay around longer, relationships you have."

The skills that the Samba team gain maybe related to the Samba project and they may even gain employment out of there involvement with it, but the bonds form have a lot more permanence about them. For at the end of the day:

> "Doing Samba is what I do. So that is what I do. I enjoy doing it, solving interesting problems, solving interesting problems *for people*"

5.5 Conclusion: The Self and Samba

This chapter has shown how the community and developers interact so as to constitute an identity as a Samba community member. The Samba project is shown to have real life impacts on notions of self and a developer's life course. The difficulty of leaving highlights the complexities and often strength of bonds that team members form as they both produce and use Samba. Samba team members identify with both the immediate Samba community, the Samba product, and the wider OS community itself. It is to the broader issues facing OS communities in general that the thesis now turns.

Chapter 6 Conclusion

It has been argued here that developers, in moving from early contributions to membership within the Samba community, engage in a social process which reconstitutes their identity. This is shown to be a fundamental aspect of why developers continue to be involved with such a community, even when they are no longer actively contributing to the project with code. This thesis argues that there is a relationship between the Samba community and the formation of an identity as a member of that community. This explanation, it is argued, offers a disciplinary perspective which augments but also challenges the dominant literature which seeks to explain and analyse why developers continue their participation within an OS community after their initial technical need or interest has been met. Rather than conceptualise the 'self' as a static or fixed entity, which can be driven by a mixture of intrinsic or extrinsic motivations, this thesis argues that a sociological account of the self and the processes of identity formation shows that the self is not a 'unit' which is essentially 'given', which can be motivated or driven, but rather is one permanently undergoing transformation according to its historical and social circumstances. In other words, the thesis argues that the Samba community has both structures and strong shared norms and values which are very conducive to successful socialisation.

This thesis also presents a challenge to sociologists writing on community and identity, in particular the writings of Zygmunt Bauman, who, as discussed earlier laments to passing of communities the location of a secure and stable identity. In a state of 'liquid modernity' he argues one seeks both security of community as well as the individual freedom to be 'different'. This thesis suggests a reconfiguring of Baumans's view of community in the wake of the research presented here. Social ties are fostered within Samba and these ties extend beyond active contribution to the project. The ability for developers to strongly identify with Samba, both as a community and as a final product, allows the developer to co-construct the self with other community members. The identity and relationships formed have lasting implications, even when the developer has stoped contributing code to the project. The idea that one simply hangs their identity in a community, only to take it down and

move to the next community hold no significant weight within the context of OS communities.

The initial contact with the Samba community was, as is common for OS development communities, to fulfil a technical question relating to code. As the thesis shows, from this initial contact the process of enculturation into the Samba community has been very successful for those interviewed. In other words, these developers were not seeking a community within which they could achieve a solace from the fragmented, globalised society but 'stumbled upon a community', as one developer describes it. Whilst developers might not initially join a community to seek identity, the practice they engage in does involve them in a process of 'self re-fashioning' resulting in stable identity that can be taken and used in a number of OS projects. As such communities like Samba do influence a developers' life course and actually go some way to constituting who they are.

It is also shown that all aspects of the software development process involves social exchanges in the form of interpersonal, abet electronically mediated, communication. The desire to look at individual and essential locations of motivation is thus extremely limited because it does not take into account the very social nature of software development that occurs within OS communities. The combined use of communities of practice and epistemic communities as analytical tools has suggested that OS projects like Samba are based around common norms and values that themselves are contested. There exist strong sets of normative behaviour around the concepts of sharing and openness. The use of social capital suggests that both the community structure, as a network, and operation of power are based upon the ability to facilitate action of others. Power and a form of hierarchy operates within such communities but relies heavily on the ability to use social capital to enrol others.

Glossary

- Binary Code – Machine readable instructions.
- BSD License - The BSD license is the license agreement that the BSD software (largely, a version of UNIX) is distributed under.
- Compile – The act of taking source code and turning it into binary code.
- Copyleft – Another term use as a synonym for the GNU General Public License
- CVS - Concurrent Versions System or Concurrent Version System, both of which keep track of all work and all changes in a set of files, such as a software project, and as such allows several developers to collaborate. The CVS has become popular in the open-source world.
- Developer – Any person who is engaged in the development of software.
- Emacs – This is a text editor program with a comprehensive set of features that is particularly popular with programmers and other technical computer users.
- End-User – Refers to any person who simply makes use of the software to achieve a task. End-users do not engage with the development process in any way.
- Free Software – No to be confused with software that does not cost any money to use, free software in this thesis refers to software that ensures the ‘freedom’ of anyone to access, modify and distribute source code.
- Kernel – The core of any operating system
- Linus Torvalds - Began the development of Linux, an operating system kernel, and today acts as the project coordinator. He is well known throughout the OS community.
- Linux – What is called the core of the operating system started by Linus Torvalds.
- Samba Team – A formal structure within the Samba project that consists of mostly core developer who control access to the code base
- Source Code – Human readable instructions.
- UNIX – An operating system
- User – *See end-user*.

Appendix

Appendix A: Interview Schedule

Section 1: Personal History

1. Could you describe your history in open source software development and the Samba project in particular?
a. When did you start contributing to an open source software (OSS) development project?
b. Profile stuff – age, number of years developing, paid work, how long paid for Samba.
c. Why did you start contributing to such projects?
d. How many projects do you actively contribute to at the moment?
e. Why do you continue to contribute to such projects (active projects)?
f. How much time per 24hr is spent involved with any aspect of OS development (i.e. hacking, user-to-user assistance, administrative tasks)?

Section 2: Experiences of OSS development?

1. Are there any distinctly positive moments you have had related to the Samba project? I.e. Stand out moment related to Samba, if so why do they stand out (mine for information of relationship between event and self).
2. Are there any distinctly negative moments you have had related to the Samba project? I.e. Stand out moment related to Samba (mine for information of relationship between event and self).
a. How did those experiences make you personally feel about yourself, who you are and what you do
3. What - are there any - achievements/experiences could have only been possible with your participation in Samba?
4. Have you acquired any 'skills' that would not be possible without the Samba project?
5. In your opinion, why does forking in an OSS project happen? Much literature sees it as a pragmatic reaction to problems.
a. How would/does "forking" make you feel if it happened to a project you were involved in (i.e. Samba)?
6. Is there a political dimension to why you do what you do? Would you identify what you do and why with the likes of Richard Stallmans vision that "information wants to be free"? (If no, depolitization of OS development, depolitization self)

Section 3: Community

1. Do you see yourself as part of a Samba community? (If not, then how would you describe the Samba group?)

2. Has there been any time when you have assisted other members of the samba community? Would you describe your actions as altruistic or is assistance the norm i.e. it's expected that Samba people assist each other?
3. How does being paid to do, amongst other things, Samba related activities? Do you see yourself as volunteering or working?
4. Could you describe any experiences which have illustrated to you that you're part of a (form of) the samba community?
5. OSS development groups have been described as places were "like minded volunteers engaging in projects of interest, with no formal ties nor hierarchy but based instead on meritocracy, and reward comes in the form of recognition and intrinsic pleasure". Then ask: do you see yourself as participating in such a group? Are there key differences to your experience of OSS development groups?
6. Do you think there are any 'unwritten' norms and values within the Samba project?
a. Would you say that there is a formal and/or informal hierarchy in the Samba project? If yes, could you elaborate? Would you define the Samba community as having some form of hierarchy? (i.e. is there a power structure, what is the power structure based on?)
b. What, if at all, is the role of status in the community? (i.e. what is ones status based upon, is it quality of contribution? What does status give those who have it?) Are there any indications as to one's status? (ie who can do / say what?)
7. Can you think of an example where such norms and values have been made clear / enacted?
8. Have you ever felt that you were NOT part of the Samba group / community? How did that make you feel?
9. Does a sense of community have any relationship to the licensing used (GPL)?
a. Do you believe that a license such as GPL has any impact on the Samba project in terms of community and how people engage in it? Has it helped in countering any negative experiences?
10. In your own experience and your knowledge of others, what makes someone a "Samba person"?
a. What type of personality would you say a Samba person has?
b. Do you have to submit code to be recognised as participating in the Samba project?
c. What about helping with documentation or answering technical questions on IRC/lists?
11. Do you meet people from Samba socially at all?

Section 4: Self and community

1. You have commented earlier that you see yourself as being part of a Samba community. Does this feeling of being part of the Samba community greatly influence you actions as a developer in Samba?

Or, depending on earlier answer:

You have said earlier that you don't see yourself as being part of a community with your work in Samba. Do you think that that feeling of not being part of a community greatly influence you actions as a developer in Samba.

2. Does a sense (or lack therefore of) of being part of a community influence your actions as a developer?

3. Does being involved in the Samba project have a big impact on how you see yourself/how you identify who you are?
a. Do you see parts of yourself in the Samba project/product?
4. Would you define yourself as a hacker?
a. If so, what does being a hacker entail in terms of beliefs and actions?
b. Is Samba part of that definition of who you are?
5. Does a sense of who you are in the Samba community mean anything to you beyond the community? I.e. Does it mean something about who you are full stop.
6. Is being part of a community like Samba pleasurable? If so, in what ways?
7. Do you think there is any relationship with how much you personally identify with a OSS project like Samba and your level of commitment?

Appendix B: Interview Request Form

The Faculty of Arts

NSW 2006 AUSTRALIA College of Humanities & Social Sciences

'Building Open Source Communities'

Are you a core developer or active developer[25] who participates in the Samba project on a volunteer basis? Are you located in NSW or ACT?

If so, you are invited to take part in a research study into the construction and maintenance of open source communities. This study is being conducted by Nicolaas Earnshaw and will form the basis for the degree of Bachelor of Arts (Honours).

If you agree to take part in this study, you will be requested to participate in an individual interview with Nicolaas Earnshaw. The interview will cover a number of general topics relating to your experiences as a participant in the Samba project, and will take place at a time and location convenient to you. The interview will take no longer than 20 minutes. Results of the study will be made available to all participants.

If you wish to participate in this research, please visit the following website: http://www.it.usyd.edu.au/~nearnsha/os/

Should you register your interest in participating, Nicolaas will be in contact soon after to arrange an appropriate time for interviewing to take place.

All aspects of the study will be kept strictly confidential. Individual participants will not be identifiable.

Participation in this study is entirely voluntary. If you do participate, you can withdraw at any time. If you withdraw from the study, there will be no need to provide a reason and there will be no consequences of such withdrawal. You will have the option of having any data already collected destroyed.

[25] For the purposes of this research, a core developer is defined as any member who is responsible for guiding and coordinating the development of an OSS project; basically, all Samba Team members. Active developers are members who regularly contribute new features, fix bugs and are well recognised for their sustained contribution within the project.

Appendix C: Creative Commons Deed

Creative Commons

creative commons

COMMONS DEED

Attribution-ShareAlike 2.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

Appendix D: Creative Commons Legal Code

Attribution-ShareAlike 2.0

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a

moving image ("synching") will be considered a Derivative Work for the purpose of this License.

c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

d. **"Original Author"** means the individual or entity who created the Work.

e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. **"License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

b. to create and reproduce Derivative Works;

c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.

e. For the avoidance of doubt, where the work is a musical composition:

i. **Performance Royalties Under Blanket Licenses**. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.

ii. **Mechanical Rights and Statutory Royalties**. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).

f. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF

LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at http://creativecommons.org/.

References

Anderson, B. (1983). Imagined Communities: Reflections of the Origin and Spread of Nationalism. Norfolk, Verso.

Bauman, Z. (1998). Globalization: The Human Consequences. Cambridge, Polity Press.

Bauman, Z. (2000). Liquid Modernity. Cambridge, Polity Press.

Bauman, Z. (2001). Community: Seeking Safety in an Insecure World. Cambridge, Polity Press.

Bauman, Z. (2004). Zygmunt Bauman: Liquid Sociality. The Future of Social Theory. N. Gane. Bath, Continuum**:** pp17-46.

Benkler, Y. (2002). "Coase's Penguin, or Linux and the Nature of the Firm." The Yale Law Journal **112**(3).

Bryman, A. (2004). Social Research Methods. New York, Oxford University Press.

Cinquegrani, R. (2002). "Futurist Networks: Cases of Epistemic Community?" Futures **34**: 779-783.

Collier-Brown, D., R. Eckstein, et al. (1999). Using Samba. Calif, O'Reilly and Associates.

Delanty, G. (2000). Modernity and Postmodernity. London, Sage.

Dempsey, B. J., D. Weiss, et al. (2002). "Who is an Open Source Software Developer?" Communications of the ACM **45**(2): 67-72.

Edwards, K. (2001). Epistemic Communities, Situated Learning and Open Source Software Development. Available at http://opensource.mit.edu/papers/kasperedwards-ec.pdf [2004 20th August].

Edwards, K. (2004). "An Economic Perspective on Software Licenses-Open Source, Maintainers and User-developers." Telematics and Informatics **22**(1-2): 111-133.

Elliott, M. and W. Scacchi (2003). Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration. Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work, Sanibel Island, Florida, USA, ACM. 21-30.

Fitzgerald, B. and J. Feller (2001). "Open Source Software: Investigating the Software Engineering, Psychsocial and Economic Issues [Guest Editorial]." Information Systems Journal **2001**(11): 273-276.

Franck, E. and C. Jungwirth (2002). Reconciling investors and donators - The governance structure of open source. Lehrstuhl für Unternehmensführung und -politik Universität Zürich. Available at http://opensource.mit.edu/papers/jungwirth.pdf [20th June 2004].

Gallivan, M. J. (2001). "Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies." Information Systems Journal **11**(4): 277-304.

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): Available at http://www.firstmonday.dk/issues/issue3_3/ghosh/ [2nd August 2004].

Ghosh, R. A. and R. Glott (2002). Free/Libre and Open Source Software: Survey and Study. R. A. Ghosh, International Institute of Infonomics, University of Maastricht and Berlecon Research GmbH.

Ghosh, R. A. and R. Glott (2002). Free/Libre and Open Source Software: Survey and Study. Available at http://www.infonomics.nl/FLOSS/report/ [12th June 2004].

Gilbert, N. (1996). Researching Social Life. London, Sage Publications.

Haas, P. (1992). "Introduction: Epistemic Communities and International Policy Coordination." International Organization **46**(1): 1-35.

Hars, A. and S. Ou (2002). "Working for Free? Motivations of Participating in Open Source Projects." International Journal of Electronic Commerce **6**(3): 25-39.

Hemetsberger, A. (2001). Fostering Cooperation on the Internet: Social Exchange Processes in Innovative Virtual Consumer Communities. Free / Open Source Research Community. Available at http://opensource.mit.edu/papers/hemetsberger2.pdf [7th July 2004].

Hemetsberger, A. (2003). When Consumers Produce on the Internet: The Relationship between Cognitive-affective, Socially-based, and Behavioral Involvement of Prosumers. Free / Open Source Research Community. Available at http://opensource.mit.edu/papers/hemetsberger1.pdf [14th July 2004].

Hemetsberger, A. and C. Reinhardt (2004). Sharing and Creating Knowledge in Open-Source Communities: The case of KDE. Free / Open Source Research

Community. Available at http://opensource.mit.edu.au/papers/hemreinh.pdf [14th April 2004].

Hertel, G., S. Niedner, et al. (2003). "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel." Research Policy **32**(7): 1159-1177.

Himanen, P. (2001). The Hacker Ethic and the Spirit of the Information Age. New York, Random House.

Hippel, E. v. (2001). "Innovation by User Communties: Learning from Open-Source Software." MIT Sloan Management Review **42**(4): 82-87.

Holtgrewe, U. (2004). "Articulating the Speed(s) of the Internet: The Case of Open Source/Free Software." Time & Society **13**(1): 129–146.

Jackman, R. W. (2001). Social Capital. International Encyclopedia of the Social & Behavioral Sciences. N. J. Smelser and P. B. Baltes. New York, Elseview**:** 14216-14219.

Joode, R. v. W. d. (2004). "Managing Conflicts in Open Source Communities." Electronic Markets **14**(2): 104-113.

Kleif, T. and W. Faulkner (2003). "“I’m No Athlete [but] I Can Make This Thing Dance!” - Men’s Pleasures in Technology." Science, Technology, & Human Values, **28**(2): 296-325.

Kollock, P. (1999). The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace. Communities in Cyberspace. M. Smith and P. Kollock. London, Routledge**:** pp200-269.

Krogh, G. V., S. Spaeth, et al. (2003). "Community, Joining, and Specialization in Open Source Software Innovation: A case Study." Research Policy **32**: 1217-1241.

Lakhani, K., B. Wolf, et al. (2002). The Boston Consulting Group: Hacker Survey. Boston, The Boston Consulting Group.

Lakhani, K., B. Wolf, et al. (2002). The Boston Consulting Group: Hacker Survey. Available at http://www.osdn.com/bcg/ [12th May 2004].

Lakhani, K. and R. Wolf (2003). Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects. MIT Sloan School of Management**:** 28.

Lakhani, K. and R. Wolf (2003a). Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects. MIT Sloan School of Management**:** 28.

Lee, M. L. and J. Davis (2003). "Evolution of Open Source Software: A Study of the Samba Project." Systemes d'Information et Management Journal **8**(1): 43-62.

Lerner, J. and J. Tirole (2002). "The Simple Economics of Open Source." Journal of Industrial Economics **50**(2): 197-234.

Lerner, J. and J. Tirole (2002). "Some Simple Economics of Open Source." The Journal of Industrial Economics **L**(2): 197-234.

Maffesoli, M. (1996). The Time of the Tribes: The decline of Individualism in Mass Society. London, Sage.

May, T. (1998). Social Research: Issues, Methods and Process. Philadelphia, Open University Press.

Michaelson, J. (2004). "There's No Such Thing as a Free Software Lunch." Queue **2**(3): 41-47.

Mustonen, M. (2003). "Copyleft - The Economics of Linux and other Open Source Software." Information Economics and Policy **15**: 99-121.

Neuman, L. (2000). Social Research Methods: Qualitative and Quantitative Approaches. Boston, Allyn and Bacon.

Newell, R. (1996). Questionnaires. Researching Social Life. N. G. (Ed). London, Newbury Park**:** pp.94-115.

Olssen, M. (2002). "Michael Foucault as "thin" Communitarian: Difference, Community, Democracy." Cultural Studies, Critical Methodologies **2**(4): 483-513.

Orlikowski, W. and J. Baroudi (1991). "Studying Information Technology in Organizations: Research Approaches and Assumptions." Information Systems Research **2**: 1-28.

Osterloh, M., S. Rota, et al. (2001). Open Source - New Rules in Software Development. Available at http://www.ifbf.unizh.ch/orga/downloads/OpenSourceAoM.pdf [4th May 2004].

Portes, A. (1998). "Social Capital: Its Origins and Applications in Modern Sociology." Annual Review Sociology **24**: 1-24.

Raymond, E. (1998). OSI Launch Announcement. Available at http://opensource.feratech.com/pressreleases/osi-launch.php [June 8th 2004].

Raymond, E. S. (2001). The Cathedral And the Bazaar. Calif, O'Reilly & Associates.

Rosenzweig, R. (1998). "Wizards, Bureaucrats, Warriors, and Hackers: Writing the History of the Internet." The American Historical Review **103**(5): 1530-1552.

Scacchi, W. (2002). "Understanding the Requirements for Developing Open Source Software Systems." 23rd International Conference on Software Engineering **149**(1): 24-39.

Stallman, R. (1984). The GNU Manifesto. Available at http://www.gnu.org/gnu/manifesto.html [March 25th 2004

Stallman, R. (1999). The GNU Operating System and the Free Software Movement. Open Sources: Voices from the Open Source Revolution. C. Dibona, S. Ockman and M. Stone. Calif, O'Reilly.

Thomas, D. and A. Hunt (2004). "Open Source Ecosystems." Software Construction **21**(4): 89-91.

Torvalds, L. (2001). Prologue: What Makes Hackers Tick? a.k.a Linus's Law. The Hacker Ethic. P. Himanen. New York, Random House**:** xii-xvii.

Veale, K. J. (2003). "Internet gift economies: Voluntary payment schemes as tangible reciprocity." First Monday **8**(12): Available at http://www.firstmonday.dk/issues/issue8_12/veale/ [3rd March 2004].

Wayner, P. (2000). Free for All: How Linux and the Free Software Movement Undercut the High-Tech Titans. New York, HarperBusiness.

Weber, S. (2000). The Political Economy of Open Source Software. BRIE Working Paper 140. Available at http://brie.berkeley.edu/~briewww/pubs/pubs/wp/wp140.pdf [3rd March 2004].

Wenger, E. (1998). Communities of Practice: Learning, Meaning, and Identity. New York, Cambridge University Press.

Wenger, E. (2001). Communities of Practice. International Encyclopedia of the Social & Behavioral Sciences. N. J. Smelser and P. B. Baltes. New York, Elsevier**:** 2339-2342.

Williams, S. (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software. Calif, O'Reilly.

Ye, Y. and K. Kishida (2003). Toward an understanding of the motivation Open Source Software developers. Proceedings of the 25th international conference on Software engineering, Portland, Oregon, IEEE Computer Society. 419 - 429.

Zeitlyn, D. (2003). "Gift economies in the development of open source software: anthropological reflections." Research Policy **32**(7): 1287-1291.