Received 17 January 2004
Revised 18 August 2004
17 October 2007
Accepted 6 December 2007

# Mobilization of software developers: the free software movement

Margaret S. Elliott and Walt Scacchi
*Institute for Software Research, University of California, Irvine, California, USA*

## Abstract

**Purpose** – The paper has three purposes: the first is to provide a deeper understanding of the ideology and work practices of free and open source software development, the second to characterize the free software movement as a new type of computerization movement and the third to present a conceptual diagram and framework with an analysis showing how the free software computerization movement has evolved into an occupational community.

**Design/methodology/approach** – Qualitative data were collected over a four year period using a virtual ethnography in a study of free and open source software development and, in particular, a study of a free software community, GNUenterprise, located at www.gnuenterprise.org, which has the goal of developing a free enterprise resource planning software system.

**Findings** – It is concluded that the ideology of the free software movement continues to be one of the factors which mobilize people to contribute to free and open source software development. This movement represents a new type of computerization movement which promotes the investment of time in learning a new software development process instead of investment of money in the acquisition and use of new technology.

**Research limitations/implications** – The research findings are limited by a detailed study of only one free software development project.

**Practical implications** – This paper is of significance to software developers and managers of firms who wish to incorporate free and open source software into their companies.

**Originality/value** – This research presents an original conceptual diagram and framework for how computerization movements have emerged into an occupational community.

**Keywords** Computer software, Public domain software, Worldwide web, Product development

**Paper type** Research paper

## Introduction

Free and open source software (F/OSS) development projects have been growing at a rapid rate. In 2000, the Sourceforge web site (www.sourceforge.org) listed 2370 F/OSS

The authors would like to thank the anonymous reviewers for their help and suggestions, and the outstanding critique and help offered by the associate editors, the late Roberta Lamb and Steve Sawyer. Professor Roberta Lamb, Institute for Software Research, University of California, Irvine, who was a colleague, sadly passed away in November 2006 prior to the completion of the paper. The authors wish to acknowledge her advice, support, and enthusiasm toward the completion of this paper. The research described in this paper is supported by grants Nos 0083075, 0205679, 0205724, and 0350754, from the US National Science Foundation. No endorsement implied. Mark Ackerman at University of Michigan, Ann Arbor; Les Gasser at University of Illinois, Urbana-Champaign; and others at ISR are collaborators on the research described in this paper.

Information Technology & People
Vol. 21 No. 1, 2008
pp. 4-33

0959-3845
DOI 10.1108/09593840810860315

projects and 15,060 F/OSS registered users, and, by 2007, it listed 100,000+ projects with 1 million + registered users. Globally dispersed virtual communities have formed over the internet with large groups of F/OSS developers contributing time and effort often without ever meeting face-to-face and without remuneration. F/OSS development represents a relatively new approach to the development of complex software systems (Feller and Fitzgerald, 2002; Scacchi, 2007) and differs significantly from typical methods of software development (Sommerville, 2006). In recent years, businesses are beginning to invest time and money in the use of F/OSS systems such as Linux (www.linux.org), a F/OSS Unix-based operating system, and are funding in-house support or consulting services for F/OSS customization often integrating them with pre-existing information systems (IS) systems (Dedrick and West, 2008). As this trend continues to grow, companies and individuals who are interested in investing time and resources into the development or use of F/OSS systems will need to understand the ideology and work practices of F/OSS developers.

One driving force behind the development of this phenomenon and the mobilization of F/OSS developers is the free software movement, a 20 year-old social movement (Blumer, 1969). The purpose of the free software movement is to promote the global use of free software with the goal of eliminating proprietary software. Richard M. Stallman, the founder of the FSM, and its ideological center, the Free Software Foundation (FSF), support the mission to promote free software and to protect the rights of free software users to experience this freedom (Stallman, 2002; Williams, 2002). In 1998, the Open Source Initiative (OSI) (www.opensource.org) was formed by a group of free software advocates who wanted to market free software as something more amenable to commercial business use. They coined the term "open source", established the concept of the open source definition (www.opensource.org), and began a marketing campaign to entice businesses to use free software such as the GNU/Linux operating system. In this paper, we address the ideological beliefs that mobilize people to contribute to the production and maintenance of "free" software, and we trace the progress of the free software movement from its ideological beginnings as a computerization movement (Elliott and Kraemer, 2008; Iacono and Kling, 2001; Kling and Iacono, 1988) to how it later split into two organizations, the FSF and the OSI with differing ideologies, to its more recent evolution into an occupational community (Van Maanen and Barley, 1984).

Computerization movements are social movements which promote visions of sweeping changes that organizations and society will experience by investing time and/or resources into specific computer-based systems (Kling and Iacono, 1988; Iacono and Kling, 2001; Elliott and Kraemer, 2008). The F/OSS movements represent a new genre of computerization movements which focuses on a change to a process of software development rather than the acquisition of new technology. In this paper, we characterize the F/OSS as a general computerization movement (Kling and Iacono, 1988) with two specific movements within it: the free software movement and the open source movement (which advocates the combination of free software with proprietary software). During the 1990s, as the F/OSS systems began to proliferate and expand into businesses and Web sites for the FSF and the OSI began advertising their causes, the occupational community of F/OSS developers evolved.

Occupational communities consist of a group of people who are engaged with the same type of work and who share similar identities, values, norms, and perspectives that apply to work but also influence life outside of work. Occupational subcultures

form within occupational communities and share beliefs, values, and norms with the occupational community, but develop some that are unique to a particular subculture (Schein, 1992; Trice and Beyer, 1993). We present the F/OSS developers as an occupational community with the F/OSS movements representing separate occupational subcultures with similar but unique ideologies. Various researchers have studied the IS personnel as an occupational community (Duliba and Baroudi, 1991; Gregory, 1983) and as various types of occupational subcultures within organizations (Marschall, 2002; Martin, 2002; Schein, 1992; Trice and Beyer, 1993). Researchers have also identified F/OSS developers as an occupational community (Elliott and Scacchi, 2003a, 2007; McCormick, 2003). Research has shown that the ideology of F/OSS developers has a positive impact on team effectiveness (Stewart and Gosain, 2006).

The contributions of this paper are:

- a deeper understanding of the ideology and work practices of F/OSS development and of how beliefs in free software influence choices in F/OSS software development methods;
- the characterization of the free software movement as a new type of computerization movement in which people are mobilized to invest time in learning a new software development process instead of the monetary investment in the acquisition and use of new technology; and
- a conceptual diagram and framework with an analysis showing how the free software movement emerged into an occupational community consisting of F/OSS occupational subcultures.

Practical implications of this paper include the discussion of the recent legal allegations and resolutions between Microsoft Corporation and the F/OSS community that directly affects F/OSS developers and companies considering the use of GNU/Linux.

The paper is organized as follows: first, background on the free software movement, computerization movements, technological utopianism, utopian communities, and occupational communities is presented; second, the research approach, data collection methods, and description of the research site are presented; third, F/OSS development is described in comparison with typical software development methods; fourth, frames which mobilize people to join the free software movement are discussed; and finally, the paper ends with a discussion, research implications/limitations, and a conclusion.

## Background

### *Free software movement*

The free software movement is a social movement promoting a new method for the development, maintenance, and release mechanism of software with the goal of creating and converting all software to "free" software (Blumer, 1969). The free software movement was envisioned in 1984 by Richard M. Stallman (widely known as RMS in F/OSS communities) due to his personal beliefs in the social need for "freedom" of information and to his increasing disillusionment with software vendors who were not supplying source code along with executable code for hardware devices like printers. In 1984, he started the GNU (GNU's Not Unix) project (www.gnu.org) by developing and distributing a UNIX-like operating system as free software. This system has evolved into the GNU/Linux system comprised of the Linux kernel, developed by Linus Torvalds and

managed by him today (www.linux.org), combined with GNU utilities. The GNU project led to the formation of the Free Software Foundation (FSF) in 1984. The FSF (www.fsf.org) is a tax-exempt charity whose purpose is to promote computer users' right to use, copy, modify, and redistribute computer programs. Stallman advocates the use of free software as a moral obligation (Williams, 2002).

As a social movement takes hold with members, it becomes more organized, with a recognized set of social rules and values, often with a movement organization (Zald and Ash, 1966), and leader(s) who promote the movement through written and oral treatise. As more people joined Stallman in promoting and developing free software, he became a key figure in the free software movement serving as the movement's leader(Stallman, 2002; Williams, 2002) with the FSF acting as the movement's organization offering a large selection of written articles on the free software movement. Recently, Eben Moglen, the lawyer representing the FSF, formed the Software Freedom Law Center at in 2005 with the purpose of providing legal representation and other law-related services to protect and advance F/OSS (www.softwarefreedom.org). One of the free software movement norms that Stallman invented and which has contributed to its continuing success (Elliott, 2008) is a licensing technique known as "copyleft".

Copyleft is a general method for licensing a computer program as free software with the requirement that all modifications and extensions to the software are free as well. One way to license a program is to place it in the public domain as an uncopyrighted piece of software for others to copy, modify, and potentially distribute as a proprietary program. The FSF recommends that free software developers use the GNU General Public License (GPL) or one of its derivatives (www.fsf.org/licenses/licenses.html/#WhatISCopyleft). By including a GPL-type license with every free software program, developers spread the philosophy and freedom intended by Stallman when he founded the FSF. As part of the copyleft procedure, the software is first copyrighted back to the FSF with distribution terms added giving everyone the rights to use, modify, and redistribute the program's code or any program derived from it but only if the distribution terms are unchanged. In this way, Stallman ensures that the freedoms that he so fervently advocates are bound legally to the code. In his opinion, proprietary software developers use copyright to take away users' freedom and the FSF uses copyright to guarantee users' freedom.

*Open source initiative*

In the early 1990s, Torvalds, a graduate student from Sweden, began work on a "free" Unix kernel. This system eventually merged with the Linux kernel software to become GNU/Linux. Although Stallman's GNU project (www.gnu.org) involved the sharing of free software with other programmers, Torvalds was the first person to make his "free" software available to others over the Internet, asking for help with testing and finding bugs. Since that time, there are 1 million+ registered F/OSS users on the Sourceforge web site.

During the late 1990s, a small group of free software advocates began suggesting that free software needed a different nomenclature and one that would attract commercial business support for producing and using free software. In 1998 a group of these advocates met and a decision was made to start a "marketing campaign" to bring free software to the attention of the world (Raymond, 2001). The term "open source" was coined and this gave rise to the open source definition and the OSI organization (www.

opensource.org). In this way, the open source movement split from the strict "free-only" ideology of the free software movement. This ideological split coincided with the formation of the Sourceforge web site (www.sourceforge.org) which offers support for the promotion and maintenance of open source projects giving people who form an open source community an opportunity to advertise for contributors in hopes of attracting more volunteer software developers. The FSF offers a list on their web site (www.fsf.org) for those F/OSS projects that consist of "free-only" software. Stallman describes the difference between the free software movement and open source movement on the FSF web site: "Open source is a development methodology; free software is a social movement." (www.fsf.org/philosophy/free-software-for-freedom.html)

The FSF describes their difference from open source on their web site (www.fsf.org):

> Another group has started using the term "open source" to mean something close (but not identical) to "free software." We prefer the term "free software" because, once you have heard it refers to freedom rather than price, it calls to mind freedom.

Stallman reiterates these beliefs in his article, "Why 'free software' is better than 'open source'" (www.gnu.org):

> The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As on person put it, "Open Source is a development methodology; free software is a social movement.".
>
> We disagree on the basic principles, but agree more or less on the practical recommendations. So we can and do work together on many specific projects. We don't think of the Open Source movement as an enemy. The enemy is proprietary software.

On the OSI web site (www.opensource.org), the following describes how it differs from free software:

> The Open Source Initiative is a marketing program for free software. It's a pitch for "free software" on solid pragmatic grounds rather than ideological tub-thumping. The winning substance has not changed, the losing attitude and symbolism have. See the discussion of marketing for hackers for more. So that it is clear what kind of software we are talking about, we publish standards for open-source licenses.

We classify the free software movement and the open source movement as related computerization movements with the FSF and the OSI as the respective movement organizations. Figure 1 illustrates a conceptual diagram of how these computerization movements evolved into an occupational community. The free software movement is shown as initiating the mobilization of members and later, the occupational community emerges with two occupational subcultures representing the free software movement and the open source movement respectively.

### *Computerization movements*

Computerization movements are social movements whose activists promote mass computerization of specific computing technologies to bring about a new social order (Kling and Iacono, 1988; Iacono and Kling, 2001). The majority of computerization movement mobilization for membership is advertised and successfully recruited through their public discourses in the mass media, professional organizations, public speeches, popular stories, television shows, and magazine articles (Kling and Iacono,

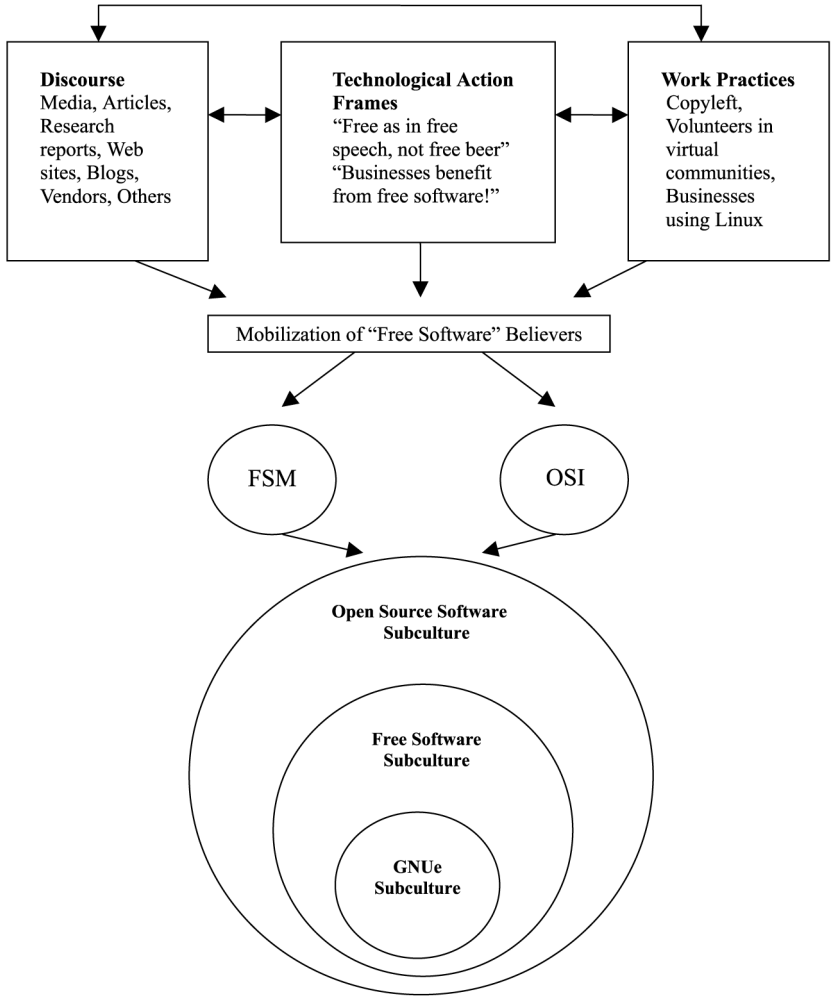


**Figure 1.**
Free software movement evolves into occupational community

1988). Often this discourse contains forms of technological utopianism which is a rhetorical form that places the use of some specific technology, such as computers as the key enabling element of a utopian vision. Often the writings promoting a computerization movement are replete with utopian visions of the wondrous things that new technology can offer organizations and businesses. These utopian visions of computerization often legitimate continuing large investments of time and/or money to technology based on the benefits promised by an ideal vision. However, there is often a gap between the vision and reality for various reasons including limitations of the technology itself, social adaptation to the technology, or problems inherent in the vision itself (Elliott and Kraemer, 2008).

In their study of internetworking as a computerization movement, Iacono and Kling (2001) combined the sociological notion of "collective action frames" (Snow and Benford, 2000) with the definition of "technological frames" (Orlikowski and Gash, 1994) to coin the term "technological action frames". Technological action frames are collective understandings that mobilize people to invest time and money into new technology and form core ideas about how the specific technology works and should or could be used. Such frames work at a macro level to influence micro-social uses in specific settings such as organizations or personal use. Technological action frames serve to mobilize people to join a computerization movement and are circulated in public discourses designed to encourage membership in the computerization movement.

These three elements – technological action frames, public discourse, and organizational practices – work in a recursive fashion mobilizing people to support investments of time and money into the use of specific technology. The free software movement is a new type of computerization movement in which the focus of change is on the process of software development not on the acquisition of new technology (Elliott, 2008). Much of the discourse promoting the free software movement which is circulated in books, web sites, user groups, academic conferences, and conventions is written in the form of technological utopianism.

*Technological utopianism*

Writers who emphasize technological utopianism characterize societies in which people lead ideal lives with technology as a key enabler of the utopia. While the abundance of technological advancements over the last 50 years has inspired technological utopianism, the original writings of utopian worlds were penned over 2,500 years ago. Plato's Republic first depicted the ideal world of utopia. The actual term utopia was initiated by Thomas More in a story he published in 1516 of an ideal society where people lived harmoniously without stress or conflicts. The word utopia actually meant "nowhere" and has since been applied in various forms of writing in which society is portrayed as making people the happiest that they could possibly be.

In the twentieth century, writers started predicting drastic utopian changes to society due to technological advances. Even before the advent of desktop computers, Vannevar Bush, in 1948, portrayed a utopian vision of a futuristic electronic device called a "memex" which would enable researchers to electronically search through large amounts of articles, books, and notes via a remote-controlled, desktop device (Bush, 1988). His vision included an astonishing prediction of what is currently technologically possible:

> Wholly new forms of encyclopedia will appear, ready-made with a mesh of associative trails running through them, ready to be dropped into the memex, and there amplified. The lawyer has at his touch the associated opinions and decision of his whole experience. The patent attorney has on call millions of issued patents, with familiar trails to every point of his client's interest (Bush, 1988, p. 32).

Such visions of computerization ignore the social implications of the use of technology and idealize the technology as though all people could easily coordinate their work through technology. They ignore social relationships in work groups such as those marked by conflict, competition, coercion, and even combativeness (Kling and Lamb, 1996). However, they serve a purpose in portraying the outer limits of potential outcomes when incorporating new technology into organizations and society (Kling and Iacono, 1988).

The free software movement promotes a utopian vision of all software being "free as in free speech, not free beer", lists these freedoms in its GPL, and links this to the ideology of people's general freedoms embedded in the US constitution and bill of rights. Unlike other computerization movements which advocate the use of specific technologies, the free software movement also promotes a life style or attitude change to a process – the development of free software. As such, it represents a virtual community of like-minded believers in the value of free software and in the moral obligation to produce free software.

The free software movement's utopian vision that some day all software should be free is anathema to the typical software development firm in the USA. In fact, big business enterprises like Microsoft Corporation have claimed anti-utopian outcomes from free software such as the threat to capitalism and the downfall of enterprise-oriented software companies like Microsoft. Later in this paper, we describe the legal battles that the FSF has fought with Microsoft over issues such as Microsoft's claim of patent infringements for users of GNU/Linux. Previous attempts at forming communities based on utopian visions have failed for various reasons including the alienation and rejection from general society (Hine, 1983). However, it appears that the free software movement and its partner, the open source software movement are still growing despite the negative rhetoric from Microsoft. In the next section we give a brief overview of utopian communities that have formed and disbanded in the past and compare these to virtual communities based on utopian visions.

*Utopian communities*

During the eighteenth and nineteenth centuries, hundreds of utopian colonies existed in America, each sharing a vision of communal living in a utopian society. Robert Hine (Hine, 1983), author of *California's Utopian Colonies*, defines a utopian colony as "a group of people who are attempting to establish a new social pattern based upon a vision of the ideal society and who have withdrawn themselves from the community at large to embody that vision in experimental form." Many of these colonies such as the Amana and Shaker colonies (Service, 2007), were attracted to the US constitution's policy of freedom of religion and came to America hoping to form utopian societies, self-containing religious or secular communities, based on a communal model of living in an agrarian setting isolated from the vagaries of city life. Others formed with the pursuit of human happiness as its goal combined with a belief in cooperative life. However, most of these colonies were not successful over the long term and most disbanded by early twentieth century for various reasons including: conflicts of an agrarian existence with an increasingly industrialized world, hostility from external forces in the larger, encompassing society, and death of inspirational leaders.

In the nineteenth century, California was the site of several utopian religious and secular societies which lasted no more than ten years. These utopian colonies played a role in establishing California as the land of opportunity for people to pursue hopes and dreams. Kling and Lamb (1996) portray the California utopian colonies as intentional communities (Kozeny, n.d.) or groups of people who have chosen to live together with a common purpose, working cooperatively to create a lifestyle that reflects their shared core values. For example in 1875 a man named Thomas Lake Harris started Fountain Grove, a religious utopian colony near Santa Rosa, California which alienated themselves from general society and eventually disbanded.

During the 1960s and 1970s several California communes formed for religious and political reasons alienating themselves from general society with their communal living and most disbanded by the 1980s. Kling and Lamb (1996) characterize these intentional communities as a fragile social form due to the high ideals of the organization and the requirements for members to make significant personal changes for the community to survive. While they may offer members a stronger sense of community than mainstream America, few of these communities are self-sustaining; people must keep in contact with the outside world for food and clothing.

Kling and Lamb (1996) suggest that in the spirit of California's quest for utopian communities, planners of suburban communities in the 1970s adopted three principles: planned security with minimal regulation, efficient consumption, and integration of production and consumption (Kling and Lamb, 1996). Kling and Lamb (1996) describe the life of people living in one of these planned communities, Irvine, California, as detached, efficient, clean, and safe in a community without planning for social interaction. What is missing from the utopian vision of planners is that they did not consider how to politically empower its citizens so Irvine residents live in a somewhat sterile social environment without a sense of political power.

Kling and Lamb (1996) likened some California visionaries' expectations of future digital lifestyles as extensions of the post-suburbia where they live and work such as Irvine's planned community. During the 1990s, California visionaries such as Howard Rheingold (Rheingold, 1993), the founder of the WELL, portray people who join virtual communities as fulfilling a desire for community that they do not find in their real personal life. The WELL was an online discussion forum offering a kind of political medium for people feeling isolated and powerless in their neighborhoods. Another utopian visionary, (Mitchell, 1995) depicted a virtual city in his book, *City of Bits: Space, Place and the Infobahn* as a market-driven networked utopia where school and university libraries will become more like information-brokering services. Digital technology will transform how we do everything from "smart" appliances to an interconnected more efficient city. Kling and Lamb (1996) suggest that the social and economic embeddedness of the utopian colonies of California will strongly influence the future shape of cyber-utopian social forms.

*Free software movement as a cyber-utopia*

What parallels can we draw between the descriptions of cyber-utopia and the free software movement's utopian vision? First, the free software movement is not a utopian colony like those of the nineteenth century where people cohabit a particular physical place and need to change their lifestyle completely to live in the utopian colony. Utopian communities that are collocated and encompass a complete change in life style are more difficult to attain than movements such as the free software movement which are based on utopian ideals, but tempered by the realities of modern day living. Second, the goal of the free software movement and projects like GNUe is to "work" towards the software development and maintenance of a computer program developed mainly by volunteers. Unlike nineteenth and twentieth century California colonies who lived in the same area, participants are free to come and go as they please – contributing one week and not the next – while still holding onto the belief system of the free software movement. While the utopian vision of replacing all proprietary

software with free software may never be obtained, the F/OSS belief and the promise of freedom of information still draws people to participate in the movement.

*Occupational communities*

One way of viewing groups with shared goals in organizations is to characterize them as occupational communities (Elliott, 2000; Van Maanen and Barley, 1984) or as occupational subcultures (Martin, 2002; Schein, 1992; Trice and Beyer, 1993). Occupational communities consist of "a group of people who consider themselves to be engaged in the same sort of work; whose identity is drawn from the work; who share with one another a set of values, norms and perspectives that apply to but extend beyond work related matters; and whose social relationships meld work and leisure" (Van Maanen and Barley, 1984, p. 287). They are bound by socially constructed rules and ethics that promote formation of shared ideologies and cultural forms.

Van Maanen and Barley (1984) suggested that computer programmers formed an occupational community due partly to programmers' control of scarce and unique knowledge in organizations, contributing to their social identity. Various researchers have studied the IS personnel as an occupational community (Duliba and Baroudi, 1991; Gregory, 1983) and as various types of occupational subcultures within organizations (Marschall, 2002; Martin, 2002; Schein, 1992; Trice and Beyer, 1993), but few have identified F/OSS developers as an occupational community (Elliott and Scacchi, 2003a; McCormick, 2003). Recently, Barley and Kunda (2006) identified itinerant technical professionals as an occupational community in the new global economy where outsourcing substantial technical work is the norm. Like itinerant technical professionals (Barley and Kunda, 2006), F/OSS developers build networks with other members of their occupational community to locate potential job opportunities, to understand software development techniques unique to F/OSS projects, keep abreast of important technical advancements, and socialize using internet chat rooms and mailing lists. Marschall (2003) identified Internet specialists as an occupational subculture or community. As in our GNUe case study, McCormick (2003) studied F/OSS developers and found that they used concertive control to monitor and manage fellow programmers.

In this paper we characterize the F/OSS development community as an occupational community with occupational subcultures (Schein, 1992; Trice and Beyer, 1993) forming to support free software projects and open source software projects. Figure 1 shows occupational subcultures within the F/OSS occupational community and how the GNUe project forms a subculture sharing the basic ideology and work practices of the free software movement. The free software occupational subculture shares the ideology of the free software movement, while the open source occupational subculture advocates the open source ideology of combining free with proprietary software. Free software projects such as GNUe share the free software movement ideology and work practices while using methods unique to their specific community such as wide daily use of the IRC channels.

In this paper, we present a conceptual diagram and framework showing how the F/OSS movements have evolved into an occupational community. We conclude that the F/OSS movements represent a new type of computerization movement where investment of time to learn a new software development process is the motivating frame, not just monetary investment in new technology. These findings are the result

of a five-year virtual ethnography of F/OSS communities and their software development techniques including a detailed study of a free software development project centered around a virtual organization. In the next section, we present details of our research approach and data collection methods.

## Research approach and data collection methods

The sources of data for this virtual ethnography (Hine, 2000) include books and articles on F/OSS development; instant messaging; transcripts captured through IRC logs; threaded email discussion messages; and other Web-based artifacts associated with the FSF (www.fsf.org) and the GNU project such as Kernel Cousins (summary digests of the IRC and mailing lists) (www.kernel-traffic.org). In addition, the GNUe project records design decisions, detailed documentation of code, virtual meetings on IRC logs, and summary digests on their web site.

This research also includes data from email and face-to-face interviews with GNUe contributors, and observations at Open Source conferences. The first author spent over 200 hours studying and perusing IRC archives and mailing list samples during the open, axial coding and analysis phases of the grounded theory. See Elliott and Scacchi (2007) for a more detailed presentation of the research presented in this paper. The FSF and GNUe web sites offered public access to downloadable IRC archives and mailing lists as well as lengthy documentation – all facilitating a virtual ethnography (Hine, 2000). For a presentation of the variables and relationships, see Elliott and Scacchi (2003b), and for more detail on the methods entailed for this research, see the Appendix.

### *Research sites*

Our research sites included the FSF (www.fsf.org), OSI (www.opensource.org), and other related F/OSS web sites. We selected the FSF as one of the research sites due to its central role in establishing and perpetuating the free software movement We selected the GNUe (www.gnuenterprise.org) virtual community as our research site to study processes of free software development and the work practices of free software developers (Elliott and Scacchi, 2003a, b, 2004; Scacchi, 2002a, b).

GNUe is a meta-project of the GNU (www.gnu.org) project that exists on the web, but otherwise has no physical place of operation or business. GNUe is nonetheless a going concern that collects enterprise software for developing electronic business and enterprise resource planning applications in one location on the web. GNUe seeks to develop:

- a set of tools that provide a development framework for enterprise information technology professionals to create or customize applications and share them across organizations;
- a set of packages written using the set of tools to implement a full enterprise resource planning system; and
- a general community of support and resources for developers writing applications using GNUe tools.

Within the GNUe community, there is a strong, often reiterated belief that project contributors should only use software tools that are also free software or that support non-proprietary data exchange formats, rather than proprietary, closed source products.

**F/OSS development**

F/OSS development represents a relatively new approach to the development of complex software systems (Feller and Fitzgerald, 2002; Scacchi, 2007). Software development techniques used in F/OSS projects are informal and self-managed with decisions generally made by meritocracy (Mockus *et al.*, 2002; Scacchi, 2004). In most situations, the resulting software system and web-based documents or development artifacts are globally accessible at little or no direct cost. F/OSS developers contribute their time and effort in F/OSS projects for a number of reasons including work-related activity, beliefs in free software (Lakhani and Wolf, 2005), achieving "geek fame" (Pavlicek, 2000), and desire to be generous and share time and source code (Bergquist and Ljungberg, 2001).

Recent studies of the motivations of individuals for contributing to free/open source software development coincide with our findings regarding the ideological motivation of free software contributors. Intrinsic motivations such as pleasure in programming and identity with the open source community were primary reasons for people to participate in free/open source programming (Hars and Ou, 2002; Lakhani and Wolf, 2005). In the survey of 79 free/open source programmers (Hars and Ou, 2002), 16.5 percent of respondents rated high on altruism and 30 percent identified strongly with the open-source community or had a kin-like relationship with other open-source programmers. In a larger survey of 684 free/open source contributors from 287 projects (Lakhani and Wolf, 2005), 42 percent strongly agreed and 41 percent somewhat agreed that the "hacker"[1] community is their primary source of identity, while 30 percent believed that all source code should be open. Similar to our GNUe case study results where interviewees expressed the importance of "giving back to the free software community", 28.6 percent surveyed were motivated by the desire to give code back to the F/OSS community (Lakhani and Wolf, 2005).

While beliefs in free software might draw people in with utopian images and promises, in reality, as with other computerization movements (Elliott and Kraemer, 2008), the utopian ideal of "freedom" in software production does not always meet expectations. As shown in our previous publications of the GNUe case study (Elliott and Scacchi, 2003a, b, 2004), developers might need to compromise principles by using a non-free support tool to promote progress. Developers must conform to core maintainers' standards and their contribution may not be included in a release unless formally accepted by core maintainers. As in other norming mechanisms used in typical software development shops, suggestions may be made to the contributor to change the code or be simply rejected. Bug reports and patches can become stymied in a massive number of unresolved issues (Ekbia and Gasser, 2008).

The management and control of F/OSS projects in virtual organizations are quite different from techniques used in typical organizations (Scacchi, 2002b). While F/OSS software development differs significantly from the typical waterfall method of software development in (Sommerville, 2006), there are corresponding key roles that both groups use to manage a software project. Table I shows the similar roles of the F/OSS software development method versus the closed method. The core maintainer(s) is someone who oversees the project and often is the person(s) who initiated the project on the web. An analogous position in a business would be the project leader. The module owner is comparable to the lead programmer on a project where a major module of software is assigned to someone who is responsible for developing and

testing it. In business there usually are a group of developers who report back to management on a routine basis, whereas in F/OSS projects contributors work on various aspects of a piece of code, either consistently or occasionally (see Jensen and Scacchi, 2007).

Table II gives a brief comparison of the F/OSS development method versus the typical closed organization. While software engineering techniques such as modular programming are borrowed from closed source development, the software development processes of F/OSS are more informal and fluid. One major difference is that proprietary software companies like Microsoft release new versions of software formally and infrequently for profit, while F/OSS communities release code informally and frequently for the benefit of the community. Another difference is in the control of software developers and the work that is accomplished. In the next section, we discuss the use of "concertive control" in F/OSS as a means of group control of work.

*Management and control*

Effective management and control of software development processes is an important factor leading to the deployment of successful IS projects (Henderson and Lee, 1992). One informal method of control is known as clan control (Ouchi, 1979) where the use of selection processes and social mechanisms are used to control the behavior of a clan defined as any group of individuals who have common goals and who are dependent on one another. This type of control is similar to concertive control (Barker, 1993; Tompkins and Cheney, 1985) except that in concertive control, the emphasis is on a team of people establishing self-management in order to accomplish a goal or task in an organization.

In a study of two F/OSS projects, Mozilla and FreeBSD, Holck and Jorgensen (2005) analyzed the tension between anarchy (supposedly necessary for attracting and keeping voluntary developers) and control (supposedly necessary for the effective production of high-quality software). Holck and Jorgensen found that the Mozilla and

**Table I.**
Software development roles – open versus closed

| F/OSS role | Typical closed role |
|---|---|
| Core maintainer | Project leader |
| Module owner | Lead programmer |
| Software developer | Contributor |

**Table II.**
Software processes – open versus closed

| Software development processes | F/OSS software development | Closed software development |
|---|---|---|
| Requirements documentation | Loosely organized | Formally organized |
| Software design and documentation | Volunteer for assignments; informal documentation | Assigned tasks; required documentation |
| Coding | Tentative schedules | Scheduled deadlines |
| Testing | Informal with beta testing | Formal testing |
| Management of code changes, bug fixes and releases | F/OSS systems (eg. CVS) used for case management; Informal frequent releases | In-house case management programs; formal infrequent releases |

FreeBSD projects rely on self-management, but also provide cores of formal hierarchy and bureaucracy, including top-level management, release management, module owners, super-reviewers (Mozilla), and several staff functions. The only area of control is in the committing process: all contributions are reviewed by a committer before being included in a release. Finding the right balance between anarchy and control is an ongoing process for all F/OSS projects as we show in our GNUe examples where concertive control is used to resolve technical conflicts.

*Concertive control in GNUe*

Concertive control grows out of group consensus about values, high-level coordination, and degree of self-management by members or workers in an organization (Barker, 1993; Tompkins and Cheney, 1985). Tompkins and Cheney (1985) coined the term "concertive control" to represent the type of control arising in decentralized, participative, and more democratic systems of control. Barker (1993) explored the question of whether or not concertive control is able to really alter the bureaucratic control it is intended to replace. He found that the workers in his experiment developed their own system of value-based normative rules that controlled their behavior more powerfully than the management that it replaced. The powerful combination of peer pressure and rational rules in the concertive system created a new iron cage "whose bars are almost invisible to the workers it incarcerates" (Barker, 1993, p. 435). The phrase "iron cage" comes from Max Weber's theory of bureaucratic control (Weber, 1958). Weber viewed bureaucracy and bureaucratic control as an inevitable by-product of rational organizations. As organizations become more dependent on rule-based systems of hierarchy, the bureaucracy becomes a powerful dominant and oppressive force that Weber called the "iron cage".

F/OSS software projects use concertive control to monitor others' behavior. They socially construct a system of rules for their virtual organization using concertive control to manage the software development. Contributers achieve concertive control by reaching a negotiated consensus on how to shape their behavior based on a set of core values. In the GNUe project, the web-based discourse regarding the control of software development takes place on internet relay chat, mailing lists, and summary digests of the IRC and mailing lists.

We presented several case studies (in Elliott, 2003 No. 10) of the conversations on a GNUe internet relay chat archive. Here we present findings from those studies to illustrate how a free software project uses "concertive control" to persuade fellow developers to adhere to the belief system of using "free-only" software for GNUe software development. In this example, a GNUe programmer, who had used Adobe Photoshop to produce a graphic for documentation, received criticism from an outsider from the European Software Foundation who was checking out GNUe to determine if they were adhering to the free software movement free software principles.

< CyrilB > Hello
< CyrilB > Several images on the GNUe website seems to be made with non-free Adobe softwares,
I hope I'm wrong: it is quite shocking. Does anybody now more on the subject?
< CyrilB > lynx -source
http://www.GNUe/modules/NS-My_eGallery/gallery/GNUe/GNUePkgArchitecture.png| strings |
head
< CyrilB > We should avoid using non-free software at all cost, am I wrong?

This criticism fueled a two-day debate on the IRC regarding how this should be handled. In that sense, they have created their own "iron cage" of rules that govern their virtual communities. While attempting to free themselves from the copyright rules of typical proprietary software production, they have created their own rational set of value-based rules that people are expected to follow in order to be a team player in a free software project. Although Neilt, the creator of the offending graphic, eventually agrees to replace the graphic with one made with free software, he makes a point of complaining about the criticism. He expresses his concern that strict belief in free software use limits the freedom of choice that should be inherent in free software projects:

< CyrilB > reinhard: neilt just said Adobe non-free software make him avoid loosing time.
< CyrilB > reinhard: and free software DO make him loose time
< neilt > CyrilB: i agree to goal GNUe, that is to use free software for stuff in cvs
Action: CyrilB is shocked
< neilt > so if there is a free software alternative, i will support that
< CyrilB > neilt: I've used xfig a couple of time, I can help you.
< neilt > i did not know xfig existed
< neilt > i am installing now
< neilt > i'll let you kow
< neilt > know how it works
< neilt > otoh i see no reason to avoid non-free software either
< neilt > if this is really a freedom thing then we should be free to use whatever we want...
...
< reinhard > neilt: i agree
< reinhard > neilt: as long as we don't take away freedom from others
< reinhard > for example
< neilt > in you case you are saying its not about freedom i guess, its about using what the free software movement tells you to use
< neilt > that is just another form of bias

During this discussion, several other lurkers suggest URLs for free graphics tools for Neilt to try. More debate follows between neilt and others and, finally, neilt reluctantly agrees to change the graphic using a "free" tool. This is an example of how the "concertive control" of the community influences the decisions of the GNUe developers. In order to preserve the "free" nature of GNUe, all work together to find a solution.

*Emergent structures*

Researchers of technology have developed structurational models of technology to show the role and influence of technologies in organizations (Barley, 1986; Orlikowski, 2000); (Orlikowski and Robey, 1991).Such models suggest that designers of technology build in structures which are then appropriated by users during their use of the technology. Technology does not itself embody structures, but rather, in practice, structures of technology use are constituted recursively as people interact regularly with certain properties and in turn, shape the rules and resources that serve to shape their interaction (Orlikowski, 2000). Thus, structures are emergent, not embodied in the technology. Orlikowski (2000) termed these enacted structures of technology use technologies-in-practice, the sets of rules and resources that are (re)constituted in people's recurrent engagement with the technologies at hand.

Like most F/OSS systems, the GNUe system is always evolving with constantly changing requirements (Scacchi, 2004). When a new company downloads and modifies

the software to suit its individual needs, according to "copyleft" licensing, it may (or may not) give these GNUe code changes back to the central repository of GNUe software. In that sense, the F/OSS technology never fully "stabilizes" its properties and its purported functionality never fully reaches "closure" (Bijker, 1987).

The GNUe software has been designed and structured to encourage modular software development, and ease of software maintenance. Research has shown that the Linux software was designed in a modular fashion by Torvalds so that hundreds of contributors would be able to understand the design enough to contribute in a meaningful way, develop new requirements, and identify and fix software bugs. Without this modularity built into the design, F/OSS projects would not be successful since it would be too difficult for contributors in a virtual community to anonymously work on small sections of the code on a sporadic basis. The GNUe Web site provides detailed documentation of the original software design and offers scenarios for how people could use GNUe in their businesses. The core maintainers purposely designed the system in modules so that frequent contributors can easily download the GNUe systems, work on their piece of the code, and upload the module when completed for further testing. The system consists of a set of GNUe tools including a user forms interface, a reporting system, and an application server, providing a development framework for software professionals, and a set of packages which are written using the software tools. The goal is to develop a complete enterprise resource planning (ERP) system but most of these modules are still in the planning phases. However, the software design enables consultants or company employees to download the GNUe tools software and use the tools to develop software customized to a particular company's needs. The goal is for those "templates" of various types of businesses to agree to donate back the software to the GNUe community so that the ERP packages portion of the software project can be shared by other GNUe users and software developers. Thus, the structure of the GNUe software reflects the free software movement's beliefs in free software.

Researchers of technology use in typical organizations (i.e. not distributed F/OSS) (Barley, 1988; MacCormack *et al.*, 2006; Orlikowski, 2000) have shown how a technology-in-practice is recurrent such that human actions toward technology use become reinforced with the technology-in-practice serving as a "behavioral and interpretive template" (Barley, 1988, p. 49). During the course of computerization movements (Elliott and Kraemer, 2008), technological action frames evolve with technology use until usually, a dominant frame emerges which drives future technology use. We view the frames of the FSM which we describe in this section as similar in practice to the "behavioral and interpretive templates" (Barley, 1988).

## Frames and mobilization

### *Origins of free software movement*

From the late 1970s to early 1980s, Stallman worked in the MIT Artificial Intelligence Lab with a group of programmers (Stallman, 2002; Williams, 2002). During his tenure there, when difficulties arose with printers, Stallman would routinely fix network problems by changing the source code to the printer drivers. However, when Xerox Corporation donated a new printer to the lab, Stallman was unable to access the proprietary source code to fix network problems. This experience was one of many that prompted Stallman to start the free software movement to promote the importance of

free software through his "copyleft" principle. Stallman formed the FSF (www.fsf.org) in 1984 while writing code for the GNU software project whose original intention was to replace the Unix operating system with "free" source code for all users (Williams, 2002).

*Freedom frame (1984-1998)*

During the period of 1984-1998, the freedom frame appealed to many software developers who contributed volunteer efforts to projects such as GNU (www.gnu.org), Linux (www.linux.org) and others. The belief that "software should be free as in free speech, not free beer" initially motivated software developers to join the FSF's cause and became the dominant frame of the computerization movement. This frame still dominates the FSF discourse today as represented by Stallman's many essays on the FSF web site (www.fsf.org) emphasizing the importance of copyleft in distributing free software and extolling the following five freedoms:

(1) freedom to run a software program for any purpose;

(2) freedom to study how the program works and adapt it to their needs;

(3) freedom to redistribute copies of the software at will;

(4) freedom to improve the F/OSS program and to distribute the altered version;

(5) required distribution of the originating license that specifies the freedoms and rights concerning the preceding properties.

In 1982, Richard Stallman began the initial GNU (Gnu's Not Unix) operating system project known as HURD - the construction of a Unix "free" software klone written in C.

In the early 1990s, Linus Torvalds, a student from Helsinki University, started developing his own free kernel using the GNU tools. Torvalds used the internet to attract fellow hackers to write software for the project and by 1991, he released a workable version of Unix, calling it Linux, a combination of his kernel and the GNU utilities. The remarkable feature of this software development was that Linux was developed over the internet by a large number of volunteers. Quality was maintained by releasing new versions weekly and gaining feedback from users within a few days. Surprisingly, this method of software development worked quite well and, by 1993, Linux was comparable in reliability to many existing UNIX versions. At that point in time, Linus Torvald's approach to software development was a radical change in even the way free software developers worked on GNU programs in small, focused groups.

This method of releasing early versions of software is also known as "Linus's Law" (Raymond, 2001) – "Given enough eyeballs, all bugs are shallow". Linux is also referred to as GNU/Linux by many in the F/OSS community due to Richard Stallman's insistence that Linux is really a combination of the Linus-developed kernel with the GNU utilities.

*Business frame (1998-2003)*

In January 1998, a group of key free software developers met to discuss ways to attract more business people to F/OSS development. They decided to start a marketing campaign to bring free software to the attention of the world (Raymond, 2001). The term "open source" was coined and this gave rise to the open source definition and the OSI organization (www.opensource.org).

During this period, companies began acquiring and developing free software and the frame of "include proprietary with free software" became popular with the OSI, causing an ideological split with factions supporting the "free-only" approach. Contending discourse evolved with the OSI and FSF forming their own web sites to proselytize their viewpoints and to offer downloadable F/OSS files. During this period, the Sourceforge web site (www.sourceforge.org) was initiated in 2000 by VA Linux (now supported by the Open Source Technology Group, a subsidiary of VA Research, Inc.) and now includes a concurrent versions system (CVS) repository (Fogel, 1999), mailing lists, bug tracking, message forums, tasks management software, web site hosting, permanent file archival, full backups, and total web-based administration. In February 2000, Sourceforge hosted 2,370 projects and 15,060 registered users, and, by September, 2006, the site listed 128,465 projects and 1,383,044 users.

One factor which may have contributed to this exponential growth and F/OSS popularity is the vast amount of skilled programmers who were out of a job due to the demise of many software companies during the dot.com crash. This trend was noted in a report of the attendance at the CodeCon conference in 2002. This conference specialized in open source projects for new methods of distributing encrypted data across the internet (Lemos, 2002). Apparently nearly 200 unemployed programmers attended to discuss work on open source projects. It is possible that many other unemployed programmers searching for work during the dot.com debacle may have joined F/OSS projects contributing to the exponential growth of F/OSS projects during the early 2000s period.

During this time, the occupational community emerges with more opportunities for F/OSS programmers' employment including consultants for businesses incorporating Linux and other F/OSS products, consultants for F/OSS distribution companies, and F/OSS software developers for companies supporting F/OSS projects.

*Occupation frame (2003-2007)*

With companies starting to invest resources in the use of GNU/Linux and F/OSS applications, more career opportunities have opened up for F/OSS developers. The occupation frame conveys the message that F/OSS developers can now make a living from working on F/OSS projects, either by consulting with companies using F/OSS applications (especially GNU/Linux), by working for a GNU/Linux distribution company such as Red Hat (www.redhat.org), or working for a company which is developing F/OSS.

This period even spawned a new type of business which offers risk management of developing and using F/OSS with in-house or proprietary software (see Blackduck (www.blackducksoftware.org)) and risk insurance (see Open Source Risk Management (www.osriskmanagement.com)). The field continues to grow and the avenues of discourse to promote the FSM philosophy have expanded to new forms of web-based communication such as blogs and wikis. In addition, many books and special issues of trade and academic journals have been published in the last decade regarding F/OSS development.

The field continues to grow and the avenues of discourse to promote the FSM philosophy have expanded to new forms of web-based communication such as blogs and wikis. In addition, many books and special issues of trade and academic journals have been published in the last decade regarding F/OSS development. One way that

F/OSS programmers can now find work is through the use of bounty systems (also known as sponsor systems or pledge systems) (Wheeler, 2006) in which a user or company asks for an improvement to a F/OSS program and states a price for what they will pay for that improvement. Some bounties are run by individual F/OSS projects such as Mozilla and GNOME among others, and some are run through a third-party system similar to an auction house such as the Participatory Culture Foundation's Bounty County Web site (www.participatoryculture.org)). The Public Software Foundation (a non-profit organization) also runs a web site that lists free and open source software bounties (www.pubsoft.org)).

As the number of F/OSS users grows, more occupational opportunities will continue to emerge for F/OSS developers especially in the area of installation and maintenance of the GNU/Linux operating system. F/OSS developers' work represents a new form of organizing where vertical lines of authority between workers and managers is shifting to a horizontal form focused on careers of achievement recognized in occupational communities, rather than rewarded by organizational achievements (Barley, 1996). The F/OSS developers are part of an occupational community that revels in the values of freedom of information and choice, and establishes career opportunities through their reputation in virtual F/OSS development communities. The managers of organizations wishing to switch to the use of GNU/Linux will need to rethink their managerial roles when hiring F/OSS developers as employees or short-term consultants.

*F/OSS versus proprietary – who will win?*

With a reputation for speed, reliability, and efficiency, GNU/Linux now has more than 12 million users worldwide and an estimated growth rate of 40 percent per year (www.linux.org). With more than one-half of *Fortune* 500 companies now using GNU/Linux (Parloff, 2007) instead of Microsoft's proprietary software, the market threat of F/OSS to Microsoft is more evident. With the recent surge in the use of GNU/Linux by individuals and companies, is it possible that users of F/OSS could eventually surpass those using Microsoft's proprietary software? Researchers have explored this question by building a model of the relationship between Microsoft and F/OSS development from an economic standpoint.

Casadesus-Masanell and Ghemawat (2006) built a formal economic model of the relationship between Microsoft and F/OSS development. Their findings showed that as long as Windows has a first-mover advantage (a larger installed base at time zero), Linux will never displace Windows of its leadership position no matter how strong demand-side learning of Linux is. Furthermore, with the FSF actively promoting free software and the extent of the Linux infiltration into the Windows market, F/OSS is here to stay (Casadesus-Masanell and Ghemawat, 2006). Casadesus-Masanell and Ghemawat (2006) suggested various ways that Microsoft could remain competitive with Linux such as increasing its own demand-side learning, reducing costs, and lessening Linux's direct and indirect network effects like making it as hard as possible for Windows applications to work on Linux and to instill fear, uncertainty, and doubt (a.k.a. FUD in the F/OSS community) into the Linux user community (Casadesus-Masanell and Ghemawat, 2006). In fact, since 1994, Microsoft has been involved in antitrust litigations over software that makes it difficult for F/OSS to develop server software that will operate with Windows.

More recently, Microsoft has been working on instilling FUD into GNU/Linux users by its 2007 announcement that GNU/Linux software is infringing on some of the Windows software patents.

*Microsoft versus F/OSS*

In May 2007, Microsoft announced that the users of GNU/Linux are infringing on 235 of its patents and hence, businesses and individuals may be liable for Microsoft royalties (Parloff, 2007). If Microsoft were to make claims on businesses who are using GNU/Linux for patent infringements, these companies may discontinue GNU/Linux use. Consequently, F/OSS developers who depend on consulting work with such companies could lose business. Microsoft's CEO Steve Balmer reacted to the outrages from F/OSS proponents by stating: "We live in a world where we honor, and support the honoring of, intellectual property…What's fair is fair." The FSF's legal counsel, Eben Moglen, responded to their claim by contending that software is a mathematical algorithm and thus, is not patentable and that software patents can be invalidated in court for various reasons, or can be proven to not have infringed in a particular circumstance.

After making the announcement on patent infringements, Microsoft began negotiations with Novell, a top Linux distributor, to accept payments in the way of Linux coupons. The F/OSS licensing technique, "copyleft" embodied in the GPL prevents Linux distributors such as Novell from paying royalties for patents because by using the GPL, they assigned their copyright to the FSF. However, Microsoft can still collect royalties from Linux end users. In fact, some companies who are currently using Linux have already entered into direct patent licenses with Microsoft. Novell began making its own negotiation with Microsoft over patent infringements and, in the summer of 2006, they made a unique deal where Novell and Microsoft agreed not to sue each other's customers for patent infringement with Novell paying Microsoft a percentage of all its Linux revenue until the end of 2011. Then Microsoft paid Novell millions for "coupons" that it could resell to customers who were interested in the Linux server software.

The F/OSS community reacted to this development by releasing a modified version of the GPL – GPLv3 (Kerner, 2007) which essentially prevents companies like Novell from entering into patent deals with Microsoft. Eben Moglen announced that if Microsoft was selling coupons that customers could trade in for Novell Linux, then Microsoft was acting as a Linux distributor, and thus, subject to the GPL. In GPLv3, Moglen added a provision that would cause Microsoft (as it continues to distribute coupons for Linux use) to waive its right to bring patent suits against Novell customers. In this way, Novell itself and its users could not be sued by Microsoft for patent infringements for using Linux. However, the FUD factor is reaching F/OSS projects. Microsoft and yet another Linux distributor Red Hat are entering into similar patent deal discussions and several large Linux corporate users have purchased Novell Linux coupons from Microsoft.

In July 2007, Microsoft announced that it was not subject to GPLv3 and would continue to distribute Linux coupons. The FSF responded to Microsoft by stating that they are not exempt from GPLv3 and that the FSF will continue to monitor and fight Microsoft on this issue. The FSF claimed that Microsoft's goal was mainly to fracture the FSM and to promote the conversion of free software into proprietary software –

something that Microsoft has been trying to do for years. This was a reference to the antitrust legislation that Microsoft has been entangled in with the European Commission since 2004.

*F/OSS benefits from Microsoft's European Union defeat*

In March 2004, the European Commission fined Microsoft Corporation $497 million over two antitrust violations:

(1) Microsoft should sell two different versions of Windows – one with Media Player and one without; and

(2) Microsoft needs to share its proprietary Windows' interoperability software that enables servers to communicate with clients over a network.

By bundling the Media Player with Windows, Microsoft maintained a monopoly over other software companies marketing media software, and by not sharing interoperability communication protocols, Microsoft prevented fair competition from other software development organizations that wanted to market server technology. The European and North American FSFs strongly supported these sanctions against Microsoft because F/OSS developers could not create and use F/OSS server technology that would interoperate with Windows.

On 17 September 2007, the Court of First Instance rejected Microsoft's appeal, agreeing with the European Commission's antitrust decision and upholding the $613 million fine which Microsoft has already paid. The Court stated that the Commission was correct in taking action to prevent Microsoft from combining the Windows Media Player into the Windows operating system, and in demanding that Microsoft reveal interoperability information to software developers of server technology (Olson, 2007; Meller, 2007 #94). The decision is recognized as a victory for the FSF Europe whose legal counsel described the ruling as "a milestone for competition...and forces Microsoft back into competing on the grounds of software technology" (Meller, 2007).

*Open source at Microsoft*

An ironic twist in this litigation drama is that, despite Microsoft's apparent attempt at slowing the progress of F/OSS as a competitor, Microsoft now has a page on its web site titled "Open source at Microsoft". It starts with a statement on the subject:

> Microsoft is focused on helping customers and partners succeed in a heterogeneous technology world. This starts with participating and contributing to a broad range of choices for developing and deploying open source approaches and applications.

Surprisingly, it even has submitted some of its own versions of licenses to the Open Source Initiative for approval. It is as though Microsoft has finally recognized the prominence of F/OSS which is not going away as it later states: "Open source is neither an industry fad, nor a magic bullet".

*Impact to F/OSS occupational community*

If Microsoft is attempting to hamper F/OSS progress by refusing to offer interoperability protocols for Windows and by threatening patent infringement sanctions for people who use GNU/Linux, it may be underestimating the strength in the F/OSS idealistic belief system that continues to draw people to the movement. The

frame of "Think free speech, not free beer" still resonates with new and old supporters of the FSM. F/OSS developers who work as consultants to GNU/Linux users will need to be familiar with the recent patent infringement allegations by Microsoft and how that might impact their clients. What started as a "free-spirited" movement to promote the use of "free software" has resulted in the emergence of the F/OSS occupational community with growing opportunities and complex legal issues regarding the melding of F/OSS and proprietary software.

## Discussion

We have shown that the free software movement started out as a computerization movement and then evolved into an occupational community with the free software and open source software communities forming occupational subcultures within the larger F/OSS occupational community. What is unique about this particular occupational community is that it has emerged from an ideological base and exists primarily in "virtual" communities. Like the occupational community of itinerant technical professionals (Barley and Kunda, 2006), F/OSS developers build networks with other members of their occupational community to locate potential job opportunities, to understand software development techniques unique to F/OSS projects, to keep abreast of important technical advancements, and to socialize using Internet chat rooms and mailing lists. We discuss the application of a model of an occupational community to F/OSS communities (Duliba and Baroudi, 1991).

### *Model of F/OSS as an occupational community*

Starting with this definition, Duliba and Baroudi (1991) developed a model for determining the antecedents of an occupational community based on both intrinsic and extrinsic factors and applied this to IS personnel in two large *Fortune* 300 financial companies. They used three antecedents in their occupational community model:

- involvement in and identify with work;
- inclusiveness of work; and
- isolation from the rest of society.

Their results showed little support for the presence of an occupational community among the IS personnel studied due to the lack of isolation of the workers. However, they suggested that in areas where isolation does occur, a stronger occupational community might exist among IS professionals. We show how these antecedents fit the F/OSS community with its sense of isolation in virtual communities.

F/OSS developers show a strong involvement and identity with their work. A recent survey of the F/OSS population (Lakhani and Wolf, 2005) showed that motivation to join F/OSS communities range from work-related activity to the belief in free software to the pleasure of programming. These responses show a high involvement and identity with their work. The high level of volunteerism among F/OSS contributors, who often work on F/OSS in the evenings after working a full day somewhere else, supports the notion of their high level of commitment and involvement with their work on F/OSS software. Duliba and Baroudi (1991) measured the inclusiveness of work construct by the degree to which IS personnel are required to follow a code of conduct and norms, such as dress standards, which are similar or different from the rest of the organization. F/OSS communities may not have dress codes, but they do have customs

and traditional ways of developing and maintaining "free software". Knowledge of these shared routines enables F/OSS to work on more than one project at a time. Depending on the F/OSS project's management style, they use various forms of "concertive control" in managing their software development. People working on F/OSS projects have cultural norms and work practices that contributors are expected to follow (Elliott, 2003, Lakhani and Wolf, 2005). F/OSS contributors work in a "virtual world" chatting with each other on IRC during evening and weekend hours, often needing to adjust the times to accommodate three or four time zones around the world. In this way, F/OSS developers usually work in isolation from the rest of society and often, from each other. GNUe and typical free software projects create micro-worlds that rely on virtual communication with each member in isolation from each other and the rest of the world, yet connected with dedication and verve for a common cause, furthering the strength of the F/OSS occupational community.

## Research implications/limitations

We have illustrated a conceptual diagram and discussed a framework for how the free software movement evolved into the F/OSS occupational community. We have drawn attention to the strong ideological presence of the FSF and its broad reach around the world. There are now sister free software foundations in many areas of the world:, e.g. Europe (www.fsfeurope.org), India (www.org.in), and Latin America (www.fsfla.org). Future research could examine the similarities and differences between the US FSF and those of other countries and how they are jointly contributing to global "free software" efforts.

While F/OSS development techniques differ significantly from a closed source software project in a typical organization, the concept of "opening" software source code for others within a company to study, reuse, and improve is steadily gaining popularity. For example, new software development techniques involving sharing of source code are becoming more popular such as agile programming (Nerur and Balijepally, 2007) where prototypes are built sooner and programmers constantly review each others' code. As more companies invest time and money into the use of F/OSS products, software development managers may begin incorporating more F/OSS techniques into their software development practices. The type of software development accomplished by F/OSS developers signifies a shift in the traditional, hierarchical organization to a vertical way of organizing (Barley, 1996) that coincides with the F/OSS occupational community. This paper points out the importance of such differences to managers who may hire F/OSS developers in the future.

This research is limited to the GNUe free software community and while their software development methods and adherence to the recommended FSF norms are typical for those projects aligned with the FSF, we do not claim that all free software projects manage software development exactly like GNUe.

## Conclusions

We have shown that the free software movement represents a new type of computerization movement where investment of time to learn a new software development process is the motivating frame, not just monetary investment in new technology. We conclude that the ideological foundation of the free software movement mobilizes people to join the movement and contribute to F/OSS development. The legal

and political morass of players in this picture is constantly evolving and the outcome of how close F/OSS will come to overtaking software giants like Microsoft is equivocal at this point in time. We hope this paper encourages future researchers to explore this dynamic phenomenon from an economic or socio-technical perspective.

The combination of the formation of web-based virtual communities developing F/OSS and the split of the free software movement into two factions – FSF and OSI – gave birth to the occupational community of F/OSS developers. Ironically, Richard Stallman started the FSF in 1985 to honor freedom and advocate for all source code to be "free" for anyone to modify and use (Stallman, 2002). His many essays on the FSF web site attest to this belief in freedom. However, much as Barker (1993) found in his study of concertive control in teams where value-based direction and management from within the group proved just as stringent and rule-driven as if from the company management it was trying to replace, many F/OSS projects form hierarchical structures of control to manage and accept changes for final releases of a product much like the bureaucratic "iron cage" of control used in large corporations. The combination of web-based communication and the use of concertive control to manage F/OSS development have resulted in successful free software projects including GNU/Linux and various business applications. In the fluid world of F/OSS development, F/OSS projects may need to exert concertive control over co-workers in order to build reliable, useable, downloadable "free" software. As more businesses become involved in the use and maintenance of F/OSS systems, new business models involving labor contracts such as internet-based bounty systems may emerge with the norms of the F/OSS occupational community evolving in new directions. For businesses and individuals embracing F/OSS systems like GNU/Linux, it is important to understand this emerging global occupational community.

## Note

1. A slang term for a computer enthusiast, i.e. a person who enjoys learning programming languages and computer systems and can often be considered an expert on the subject(s). Among professional programmers, depending on how it used, the term can be either complimentary or derogatory, although it is developing an increasingly derogatory connotation. The pejorative sense of hacker is becoming more prominent largely because the popular press has adopted the term to refer to individuals who gain unauthorized access to computer systems for the purpose of stealing and corrupting data. Hackers, themselves, maintain that the proper term for such individuals is cracker.

## References

Barker, S.R. (1993), "Tightening the iron cage: concertive control in self-managing teams", *Administrative Science Quarterly*, Vol. 38 No. 3, pp. 408-37.

Barley, S.R. (1986), "Technology as an occasion for structuring: evidence from observation of CT scanners and the social order of radiology departments", *Administrative Science Quarterly*, Vol. 31 No. 1, pp. 78-108.

Barley, S.R. (1988), "Technology, power, and the social organization of work: towards a pragmatic theory of skilling and deskilling", in Ditomaso, N. and Bacharach, S. (Eds), *Research in the Sociology of Organizations*, JAI Press, Greenwich, CT, pp. 33-80.

Barley, S.R. (1996), "Technicians in the workplace: ethnographic evidence for bringing work into organizations studies", *Administrative Science Quarterly*, Vol. 41 No. 3, pp. 404-41.

Barley, S.R. and Kunda, G. (2006), "Contracting: a new form of professional practice", *Academy of Management Perspectives*, Vol. 19, February, pp. 1-19.

Bergquist, M. and Ljungberg, J. (2001), "The power of gifts: organizing social relationships in open source communities", *Information Systems Journal*, Vol. 11 No. 4, pp. 305-20.

Bijker, W.E. (1987), *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*, MIT Press, Cambridge, MA.

Blumer, H. (1969), "Social movements", in Mclaughlin, B. (Ed.), *Studies in Social Movements: A Social Psychological Perspective*, Free Press, New York, NY, pp. 8-29.

Bush, V. (1988), "As we may think", in Greif, I. (Ed.), *Computer-Supported Cooperative Work: A Book of Readings*, Morgan-Kaufmann, San Mateo, CA, pp. 17-34, (1949 reprint).

Casadesus-Masanell, R. and Ghemawat, P. (2006), "Dynamic mixed duopoly: a model motivated by Linux vs Windows", *Management Science*, Vol. 52 No. 7, pp. 1072-84.

Crowston, K. and Scozzi, B. (2002), "Exploring strengths and limits on open source software engineering processes; a research agenda", *Proceedings of the 2nd Workshop on Open Source Software Engineering, Orlando, Florida*.

Dedrick, J. and West, J. (2008), "Movement ideology vs. user pragmatism in the organizational adoption of open source software", in Elliott, M. and Kraemer, K.L. (Eds), *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, Information Today, Inc., Medford, NJ.

Duliba, K. and Baroudi, J. (1991), "IS personnel: do they form an occupational community?" *Proceedings of the Special Interest Group (SIG) on Computer Personnel Research (CPR) Annual Conference*, ACM, Athens, GA, pp. 111-18.

Ekbia, H.R. and Gasser, L. (2008), "Seeking reliability in freedom: the case of F/OSS", in Elliott, M. and Kraemer, K. (Eds), *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, Information Today, Inc., Medford, NJ.

Elliott, M. (2000), "Organizational culture and computer-supported cooperative work in a common information space: case processing in the criminal courts", unpublished dissertation, University of California, Irvine, CA.

Elliott, M. (2003), "The virtual organizational culture of a free software development community", *Proceedings of the 3rd Workshop on Open Source Software Engineering, Portland, Oregon*, pp. 40-4.

Elliott, M. (2008), "Examining the success of computerization movements in the ubiquitous computing era: free and open source software movements", in Elliott, M.S. and Kraemer, K.L. (Eds), *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, Information Today, Inc., Medford, NJ.

Elliott, M. and Kraemer, K.L. (Eds) (2008), *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, Information Today, Inc., Medford, NJ.

Elliott, M. and Scacchi, W. (2003a), "Free software developers as an occupational community: resolving conflicts and fostering collaboration", *Proceedings of the GROUP '03.ACM, Sanibel Island, Florida*, pp. 44-66.

Elliott, M. and Scacchi, W. (2003b), *Free Software: A Case Study of Software Development in a Virtual Organizational Culture, Technical Report(UCI-ISR-03-06*, Institute for Software Research, University of California, Irvine, CA, available at: www.isr.uci.edu/tech_reports/UCI-ISR-03-6.pdf

Elliott, M. and Scacchi, W. (2004), "Free software development: cooperation and conflict in a virtual organizational culture", in Koch, S. (Ed.), *Free/Open Source Software Development*, Idea Group Publishing, Hershey, PA, pp. 152-72.

Elliott, M. and Scacchi, W. (2007), "Evolution from computerization movement to occupational community: the free and open source software movements", working paper, Institute for Software Research, University of California, Irvine, Irvine, CA.

Feller, J. and Fitzgerald, B. (2002), *Understanding Open Source Software Development*, Addison-Wesley, New York, NY.

Fogel, K. (1999), *Supporting Open Source Development with CVS*, Coriolis Press, Scottsdale, AZ.

Gregory, K. (1983), "Native-view paradigms: Multiple cultures and culture conflicts in organizations", *Administrative Science Quarterly*, Vol. 28 No. 3, pp. 359-76.

Hars, A. and Ou, S. (2002), "Working for free? Motivations for participating in open-source projects", *International Journal of Electronic Commerce*, Vol. 6 No. 3, pp. 25-39.

Henderson, J.C. and Lee, S. (1992), "Managing I/S design teams: a control theories perspective", *Management Science*, Vol. 38 No. 6, pp. 757-77.

Hine, C. (2000), *Virtual Ethnography*, Sage Publications, London.

Hine, R.V. (1983), *California's Utopian Colonies*, University of California Press, Berkeley, CA.

Holck, J. and Jorgensen, N. (2005), "Do not check in on red: control meets anarchy in two open source projects", in Koch, S. (Ed.), *Free/Open Source Software Development*, Idea Group Publishing, Hershey, PA, pp. 1-26.

Iacono, S. and Kling, R. (2001), "Computerization movements: the rise of the internet and distant forms of work", in Yates, J.A. and Maanen, J.V. (Eds), *Information Technology and Organizational Transformation: History, Rhetoric and Practice*, Sage Publications, Thousand Oaks, CA, pp. 93-136.

Jensen, C. and Scacchi, W. (2007), "Role migration and advancement processes in OSSD projects: a comparative case study", *Proceedings of the 29th International Conference on Software Engineering, Minneapolis, MN*, pp. 364-74.

Kerner, S.M. (2007), "Stallman urges users to upgrade to GPLv3", *Internetnews.com*, 29 June, available at: www.internewnews.cm/devnews/article.php/3686486

Kling, R. and Iacono, S. (1988), "The mobilization of support for computerization: the role of computerization movements", *Social Problems*, Vol. 35 No. 3, pp. 226-42.

Kling, R. and Lamb, R. (1996), *Bits of Cities: Utopian Visions and Social Power in Placed-Based and Electronic Communities*, Center for Social Informatics, Indiana University, Bloomington, IN, available at: scholarworkds.iu.edu/dspace/html/2022/1805.wp96-02B.html

Kollock, P. and Smith, M. (1999), "Communities in cyberspace", in Smith, M. and Kollock, P. (Eds), *Communities in Cyberspace*, Routledge, London, pp. 3-25.

Kozeny, G. (n.d.), *Intentional Communities: Lifestyles Based on Ideals*, Community Catalyst Project, San Fransisco, CA, available at: www.ic.org/pnp/cdir/1995/01kozeny.php

Lakhani, K.R. and Wolf, R.G. (2005), "Why hackers do what they do: understanding motivation and effort in free/open source software projects", in Feller, J.R., Fitzgerald, S., Hissam, S. and Lakhani, R.K. (Eds), *Perspectives on Free and Open Source Software*, MIT Press, Cambridge, MA, pp. 3-21.

Lemos, R. (2002), "Dot-com dropouts share open-source love", *CNET News.com, San Fransisco*, 18 February, available at: www.news.com/2100-1001-839705.html

McCormick, C. (2003), "The big program that never ends: role and task negotiation within an emerging occupational community", unpublished dissertation, University of Albany New York, NY.

MacCormack, A., Rusnak, J. and Baldwin, C.Y. (2006), "Exploring the structure of complex software designs: an empiricial study of open source and proprietary code", *Management Science*, Vol. 52 No. 7, pp. 1015-30.

Marschall, D. (2002), "Internet technologists as an occupational community: ethnographic evidence", *Information, Communication & Society*, Vol. 5 No. 1, pp. 51-69.

Martin, J. (2002), *Organizational Culture: Mapping the Terrain*, Sage Publications, Thousand Oaks, CA.

Meller, P. (2007), "Microsoft loses appeal against EU antitrust ruling", *InforWorld*, 17 September, available at: www.infoworld.com/07/09/17/Microsoft-loses-appeal-against-EU-antitrust-ruling.html

Mitchell, W.J. (1995), *City of Bits: Space, Place and the Infobahn*, The MIT Press, Cambridge, MA.

Mockus, A., Fielding, R.T. and Herbsleb, J. (2002), "A case study of open source software development: the apache server", *ACM Transactions on Software Engineering and Methodology*, Vol. 11 No. 3, pp. 309-46.

Nerur, S. and Balijepally, V. (2007), "Theoretical reflections on agile development methodologies: the traditional goal of optimization and control is making way for learning and innovation", *Communications of the ACM*, Vol. 50 No. 3, pp. 79-83.

Olson, P. (2007), "Microsoft has soft defense in antitrust case", *Forbes*, 17 September, available at: www.forbes.com/markets/2007/09/17/microsoft-eu-closer-market

Orlikowski, W. and Gash, D.C. (1994), "Technological frames: making sense of information technology in organizations", *ACM Transactions on Information Systems*, Vol. 12, pp. 174-207.

Orlikowski, W. and Robey, D. (1991), "Information technology and the structuring of organizations", *Information Systems Research*, Vol. 2, pp. 143-69.

Orlikowski, W.J. (2000), "Using technology and constituting structures: a practice lens for studying technology in organizations", *Organization Science*, Vol. 11 No. 4, pp. 404-28.

Ouchi, W.G. (1979), "A conceptual framework for the design of organizational control mechanisms", *Management Science*, Vol. 25 No. 9, pp. 833-48.

Parloff, R. (2007), "Microsoft takes on the free world", *Fortune Magazine*, 28 May, available at: money.cnn.com/magazines/fortune/fortune_archive/2007/05/28

Pavlicek, R.G. (2000), *Embracing Insanity: Open Source Software Development*, SAMS Publishing, Indianapolis, IN.

Raymond, E.S. (2001), *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates, Sebastopol, CA.

Rheingold, H. (1993), *The Virtual Community: Homesteading on the Electronic Frontier*, Addison-Wesley, Reading, MA.

Scacchi, W. (2002a), *Open EC/B: A Case Study in Electronic Commerce and Open Source Software Development*, Institute for Software Research, UC Irvine, Irvine, CA, available at: http://www.ics.uci.edu/%7Ewscacchi/Papers/EC/OpenEC-July2002.pdf

Scacchi, W. (2002b), "Understanding requirements for developing open source software systems", *IEE Proceedings-Software*, Vol. 149 No. 2, pp. 24-39.

Scacchi, W. (2004), "Free and open source development practices in the game community", *IEEE Software*, Vol. 21 No. 1, pp. 59-67.

Scacchi, W. (2007), "Free/open source software development: recent research results and methods", *Advances in Computers*, Vol. 69, pp. 243-95.

Schein, E.H. (1992), *Organizational Culture and Leadership*, Jossey-Bass, San Francisco, CA.

Service, N.P. (2007), The Amana Colonies: Utopias in America, Institution, City, available at: www.nps.gov/history/nr/travel/amana/utopia.html.

Snow, D. and Benford, R. (2000), "Framing processes and social movements: an overview and assessment", *Annual Review Sociology*, Vol. 26, pp. 611-39.

Sommerville, I. (2006), *Software Engineering*, 7th ed., Addison-Wesley, New York, NY.

Stallman, R.M. (2002), *Free Software, Free Society: Selected Essays of Richard M. Stallman*, GNU Press, Cambridge, MA.

Stewart, K.T. and Gosain, S. (2006), "The impact of ideology on effectiveness in open source software development teams", *MIS Quarterly*, Vol. 30 No. 2, pp. 291-314.

Tompkins, P.K. and Cheney, G. (1985), "Communication and unobtrusive control in contemporary organizations", in Mcphee, R.D. and Tornpkins, P.K. (Eds), *Organizational Communication: Traditional Themes and New Directions*, Sage, Newbury Park, CA, pp. 179-210.

Trice, H.M. and Beyer, J.M. (1993), *The Cultures of Work Organizations*, Prentice Hall, Englewood Cliffs, NJ.

Van Maanen, J.V. and Barley, S.R. (1984), "Occupational communities: culture and control in organizations", *Research in Organizational Behavior*, Vol. 6, pp. 287-365.

Weber, M. (1958), *The Protestant Ethic and the Spirit of Capitalism*, Scribners, New York, NY.

Wheeler, D.A. (2006), "Why open source software/free software (OSS/FS, FLOSS, or FOSS)? Look at the numbers!", available at: www.dwheeler.com.

Williams, S. (2002), *Free as in Freedom: Richard Stallman's Crusade for Free Software*, O'Reilly & Associates, Sebastopol, CA.

Zald, M.N. and Ash, A. (1966), "Social movement organizations: growth, decay, and change", *Social Forces*, Vol. 44 No. 3, pp. 327-41.

## Appendix

Table AI lists the types of data and the methods used to analyze it. The stages of research refer to the following:

(1) Reading and studying of free and open source books, articles, web sites.

(2) Site selection, open coding of case studies.

(3) Axial coding of three case studies looking for commonalities and differences; comparison of GNUe with other free and open source communities; use of GNUe's search engine to locate discussions regarding ideology in IRC and mailing list archives.

The interview protocols were of two types: email and face to face. For the e-mail interviews, two core maintainers who were prominent in the IRC discussions were sent a list of questions via e-mail. One answered quickly and agreed for some follow-up questions:

(1) Who decides when a release will occur? Are decisions made by committee or is there one person in charge making top level decisions?

(2) Are all contributors non-paid? Are the core maintainers the only ones being paid by consulting firms (part-time)? Do people ever meet face-to-face?

(3) In general, what are some of the obstacles and problems that arise in this environment? How are tasks assigned to contributors?

| Type of data | Stage/collection method | Analysis method |
|---|---|---|
| Books and articles on free and open source software development; Websites of free/open source projects | I – Library and web searches | Reading and outlining books and articles. Open coding of outlines looking for conceptual framework to help select a research site |
| Site data (GNUe and FSF)<br>IRC<br>Kernel Cousins<br>Mailing lists<br>Web site articles | I – Study of web site documentation, examined KCs indexed by topic, found topics of conflict and cooperation, selected cases from IRC details. I | Kernel Cousins coded by topics; cases selected from KCs; IRC and Mailing list archives of three cases – open coding; conceptual diagram |
| Interviews<br>E-mail<br>Face to face | I (Email) – set up conference F2F meeting II (face to face) | Analysis along open codes derived from site data |
| Archives of site data IRC Kernel Cousins mailing lists | III Review of case files again, built graphical conceptual diagram; Review similar KCs and IRCs to test theory | Axial coding and comparison of GNUe with other free and open source projects. As of early 2004, GNUe upgraded web site with a Wiki and a sophisticated search engine. Used search to discern similar patterns of work across archives |
| Source code and documentation; web site Wiki for GNUe | III Explore more IRC archives to verify theory from earlier research using GNUe search engine; study wiki | Download software for PC; Attempt to install working GNUe unsuccessful to date |

**Table AI.**
AI Research data and analysis methods

(4) Many of the IRC logs show how conflicts arise and are resolved. Is that the main channel for problem solving instead of say, email or phone calls.

Follow-up questions after attending an Open Source conference:

(1) Is it common practice to accept criticism from relative newcomers to the work?

(2) Are some GNUe programmers motivated by establishing a reputation in the community?

(3) How do people use the Kernel Cousins?

(4) Can you help us by reviewing our GNUe research papers?

Then about a year into the research, we attended a Linux conference where the same core maintainer was interviewed with the following questions:

(1) Open ended at first, interviewee talked for about ten minutes about free software and GNUe's future.

(2) What about IRC? We noticed that you use these to resolve conflicts?

(3) What do you do with people who contribute code that is not up to your standards?

(4) General questions about decision making about GNUe software development.

(5) What do you think of Compiere, the French free ERP software package?

(6) What is your motivation to participate in free software? Is building a reputation important?

(7) How do you schedule activities in free software projects? Do you know who does what and do you have timelines?

(8) Are any other open source projects using IRC as a communication medium?

(9) How many companies are using GNUe?

(10) How can we help you other than contributing code?

**Corresponding author**
Margaret S. Elliott can be contacted at: melliott@ics.uci.edu