Understanding the Development of Free E-Commerce/E-Business Software: A Resource-Based View

Walt Scacchi
Institute for Software Research
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425 USA
Wscacchi@ics.uci.edu

August 2006

Revised version to appear in S.K. Sowe, I. Stamelos, and I. Samoladas (eds.), *Emerging Free/Open Source Software Practices*, IDEA Group Publishing, Hershey, PA.

Abstract

This study seeks to identify and characterize the array of social and technical resources needed to support the development of open source software supporting E-Commerce (EC) or E-Business (EB) capabilities. This entails a case study within a virtual organization that has undertaken an organizational initiative to develop, deploy, and support free/open source software systems for EC or EB services, like those supporting Enterprise Resource Planning. The objective this study is to identify and characterize the resource-based software product development capabilities that lie at the center of the initiative, rather than the software itself, or the effectiveness of its operation in a business enterprise. By learning what these resources are, and how they are arrayed into product development capabilities, we can provide the knowledge needed to understand what resources are required to realize the potential of Free EC and EB software applications. In addition, the resource-based view draws attentions to those resources and capabilities that provide potential competitive advantages and disadvantages of the organization in focus.

Introduction and Background

Many companies face a problem in determining how to best adopt and deploy emerging capabilities for E-Commerce and E-Business services. This study employs a *resource-based view* of the organizational system involved in developing an open source EC/EB software products or application systems. This chapter examines the GNUenterprise.org (hereafter GNUe) project as a case study. The analysis and results of the case study focus attention to characterizing an array of social and technical resource the developers must mobilize and bring

together in order in the course of sustaining their Free EC/EB software development effort. Free EC/EB results from applying free software development concepts, techniques, and tools [Williams 2002] to supplant those for open source software supporting EC and EB [cf. Carbone and Stoddard 2001].

This study does not focus on the software functionality, operation, or development status of the GNUe Free EC/EB software, since these matters are the focus of the GNUe effort, and such details can be found on that project's Web site. Similarly, it does not discuss what EC/EB application packages are being developed or their operational status, though the categories of software packages can be seen in Exhibit 1, presented later. Instead, the resource-based view [Acedo, *et al.* 2006, Barney 2001] that is the analytical lens employed in this chapter helps draw attention to a broader array of resources and institutionalized practices (i.e., resource-based capabilities) [Oliver 1997] that may better characterize the socio-technical investments that provide a more complete picture of the non-monetized costs associated with the development of free/open source software (FOSS), as well as possible competitive advantages and disadvantages [Hoopes, *et al.* 2003]. Such a characterization might then eventually inform other studies that seek to identify and explain the "total costs of operations" involved in developing, deploying and sustaining FOSS, or the commercial services that support them.

Case Study: The development of Free EC/EB software in GNUe

GNUe is an international virtual organization for software development [Crowston and Scozzi 2002, Noll and Scacchi 1999] based in the U.S. and Europe that is developing an Enterprise Resource Planning (ERP) system and related EC/EB packages using only Free software. One of their overarching goals is to put freedom back into "free enterprise", as seen in the overview of

GNUe shown in Exhibit 1, which is taken from the project's Web site. This organization is centered about the GNUenterprise.org Web site/portal that enables remote access and collaboration. Developing the GNUe software occurs through the portal that serves as a global information sharing workplace and collaborative software development environment. Its paid participants are sponsored by one or more of a dozen or so companies spread across the U.S. and Europe. These companies provide salaried personnel, computing resources, and infrastructure that support this organization. However, many project participants support their participation through other means. In addition, there are also dozens of unpaid volunteers who make occasional contributions to the development, review, deployment, and ongoing support of this organization, and its software products and services. Finally, there are untold numbers of "free riders" [Olson 1971] who simply download, browse, use, evaluate, deploy, or modify the GNUe software with little/no effort to contribute back to the GNUe community.

GNUe is a community-oriented project, as are most sustained FOSS development efforts [Scacchi 2002a, Sharman *et al*., 2002, West and O'Mahony 2005]. The project started in earnest in 2000 as the result of the merger of two smaller projects both seeking to develop a free software solution for EC/EB applications. More information on the history of the GNUe project can be found on their Web site.

The target audience for the GNUe software application packages is envisioned primarily as small to mid-size enterprises (SMEs) that are underserved by the industry leaders in ERP software. These SMEs may be underserved due to the high cost or high prices that can be commanded for

commercial ERP system installations. Many of these target SMEs might also be in smaller or developing countries that lack a major IT industry presence.

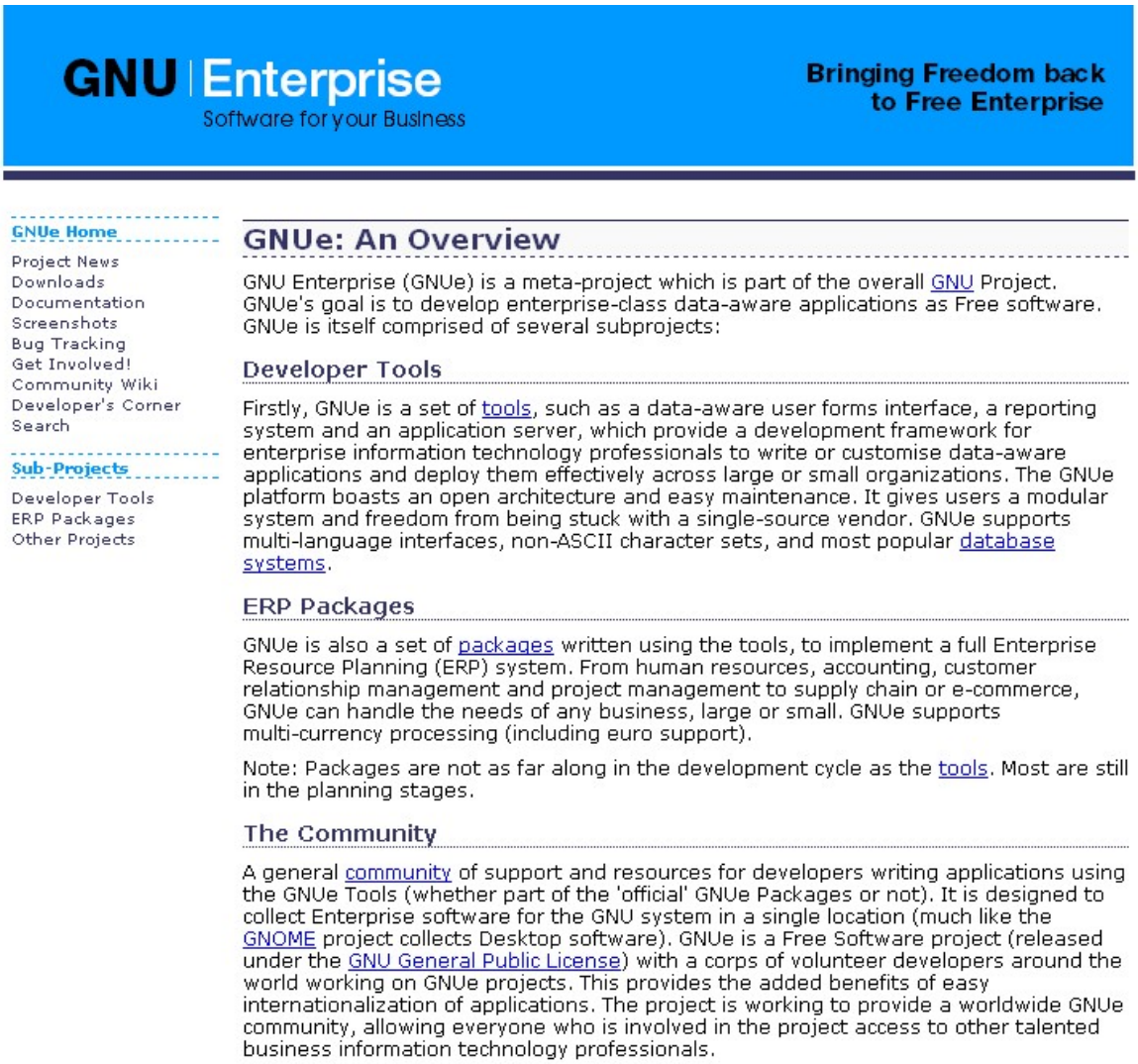

Exhibit1. Overview of the GNUe and its GNUe software (Source: http://www.gnuenterprise.org/, April 2006)

GNUe is a free software project affiliated with the Free Software Foundation and the European FSF. The ERP and EC/EB software modules and overall system architecture are called the GNUe software. All the GNUe software is protected using the GNU Public License (GPL)

[DiBona, Ockman, and Stone 1999, Pavlicek 2000, Williams 2002]. This stands in contrast to the open source ERP software from Compiere[1], which depends on the use of a commercial Oracle DBMS, or other commercially-based OSS ERP project like OpenMFG.com and Openbravo.com. Thus, GNUe is best characterized as a *free software* project [Williams 2002], rather than simply an open source software project [Feller and Fitzgerald 2002]. But many GNUe participants also accept its recognition as an open source software project, since most OSS and all free software projects employ the GPL to insure the FOSS nature of their development activities and products.

GNUe itself is not in business as a commercial enterprise that seeks to build products and/or offer services. It is not a dot-com business, but is a "dot-org" community venture. The “business model” of GNUe is more of a pre-competitive alliance (or a “cooperative”) of software developers and companies that want to both cooperate and participate in the development and evolution of free ERP and EC/EB software modules. As such, it has no direct competitors in the traditional business sense of market share, sales and distribution channels, and revenue streams.

GNUe does not represent a direct competitive threat to ERP vendors like SAP, Oracle, or JD Edwards. This will be true until these companies seek to offer low-cost, entry-level ERP or EC/EB service applications for SME customers. However, it does compete for attention, participation, independent consulting engagements, and mindshare from potential FOSS developers/users with companies like Compiere.com, OpenMFG.com, Openbravo.com, and others that seek to develop and deploy OSS for ERP applications and EC/EB service offerings

[1] Compiere.com is a software product development community that is building an open source software ERP system that requires use of Oracle. It is not however free software, as in "freedom" software [Williams 2002]. Compiere.com however claims more than 500K copies of its software have been downloaded or installed, making it the most widely deployed ERP system in the world, whether as a proprietary or FOSS-based offering.

that may incorporate non-free, closed-source, proprietary software. In addition, since the development of the GNUe software is open for global public review and corporate assessment, it is possible that the efforts and outcomes of GNUe might influence other companies developing ERP or EC/EB software. For example, other non-free, closed source ERP software vendors may perceive competitive pressure of new system features, lower cost software products, better quality, more rapid maintenance, or modular system architectures [CW360, 2002] arising from the globally visible FOSS development efforts of GNUe.

The GNUe virtual organization is informal. There is no lead organization or prime contractor that has brought together the alliance as a network. It is more of an emergent organizational form where participants have in a sense discovered each other, and have brought together their individual competencies and contributions in a way whereby they can be integrated or made to interoperate [Crowston and Scozzi 2002, Crowston and Howison 2005]. In GNUe, no company or corporate executive has administrative authority or resource control to determine: (a) what work will be done; (b) what the schedule will be; (c) who will be assigned to perform specified tasks; (d) whether available resources for the project are adequate, viable, or extraneous; or (e) who will be fired or reassigned for inadequate job performance. As such, there is comparatively little administrative overhead to sustain ongoing going software development and community portal support activities. Instead, there is a group of core developers, secondary contributors, and casual volunteers who review and comment on what has been done [cf. Jensen and Scacchi 2006]. The participants come from different small companies or act as individuals that collectively move the GNUe software and the GNUe community forward. Thus, the participants

self-organize in a manner more like a meritocracy [Fielding 1999, Scacchi 2004], rather than a well-orchestrated community for Web-based commerce or entertainment [Kim 2000].

Certain kinds of software development decisions are made by "logically centralized but physically distributed" core developers [cf. Noll and Scacchi 1999]. These core developers have earned the trust, sustained their commitment of personal time and effort on the project, have been recognized as technical authorities in the project, and have achieved some degree of "geek fame", in the eyes other project participants [cf. Fielding 1999, Pavlicek 2000]. Like other project participants and FOSS developers, the GNUenterprise core developers are expected to uphold and reiterate the freedom of expression, sharing, and learning that free, open source GNUe software represents or offers. So as core developers of GNUe software, they must reflect on how their software development decisions reflect, embody, or otherwise reproduce belief in free, open source software. On the other hand, decisions to contribute gifts of skill, time, effort, and other production resources that give rise to software, online communications, and technical peer reviews, are externalized or decentralized across a virtual organization [Bergquist and Ljundberg 2001, Crowston and Scozzi 2002]. This decentralization of costs reduces the apparent direct cost and administrative overhead (indirect cost) of OSSD by externalization and global distribution, while sustaining something of a centralized decision-making authority. Thus, individual, corporate, and collective self-interest are motivated, sustained and renewed in a manner accountable to the culture and community that is GNUe [cf. Monge, *et al*., 1998].

As such, these conditions make this study unique in comparison to previous case studies of EC or EB initiatives, which generally assume the presence of a centralized administrative authority

and locus of resource control common in large firms [e.g., Scacchi 2001]. But it is similar to prior FOSS case studies [e.g., Scacchi 2002a, German 2003] that focus attention to the array of resources whose value is simultaneously both social and technical (i.e., socio-technical resources). Nonetheless, we still need a better understanding of what resource-based capabilities are brought to bear on the development and deployment of EC/EB and ERP software by GNUe. Subsequently, what follows is a description of key resources being employed throughout GNUe to develop and support the evolution of the GNUe software modules.

Analyzing the GNUe Case

This section presents an interpretive analysis of the case study, as is appropriate for the kinds of data and descriptions that have been presented and in related studies [cf. Scacchi 2001, 2002a, Skok and Legge 2002].

A reasonable question to ask at this point is whether GNUe is an efficient and effective enterprise, and whether its participants realize gains that outweigh their individual investments. As a FOSS development alliance and virtual enterprise, GNUe is not designed to make money or be profitable in the conventional business sense. It is, however, conceived to be able to develop and deploy complex ERP and EC/EB software modules. Companies that provide paid software developers to work on the GNUe software expect to make money from consulting, custom systems integration and deployment, and ongoing system support. These services generally accompany the installation and deployment of this kind of software. They may also just seek to acquire, use, and deploy open ERP or EC/EB applications for their own internal EB operations. Similarly, they may value the opportunity to collaborate with other firms or other highly competent ERP and EC/EB software developers [Crowston and Scozzi 2002, Jensen and Scacchi

2006, Monge, *et al.,* 1998]. Other unpaid contributors and volunteers make also share in these same kinds of values or potential outcomes.

Can an enterprise make money from creating a complex ERP and EC/EB software suite that from the start is distributed as free, open source software? Don't ERP and EC/EB software products whose proprietary closed source alternatives from SAP and others cost upwards of a million dollars or more [Curran and Ladd 2000, Keller and Tuefel 1998]? Yes, closed source ERP and EC/EB systems do entail a substantial acquisition, implementation, deployment, and support costs. But the purchase price of most ERP software packages and EC/EB service application may only represent 5-10% of the total cost of a sustained deployment in a customer enterprise. Subsequently, most of the financial cost of an ERP or EC/EB application deployment is in providing the installation, customization, and maintenance support services. As FOSS in widespread use is subject to continuous improvement, then the opportunity to provide ongoing support services to businesses or government agencies that rely on them will continue and grow. Thus, a FOSS project like GNUe can still serve to generate opportunities for support service providers, without the need to generate revenues from sales of their ERP and EC/EB software. Commercial vendors like IBM, RedHat, JBoss (recently acquired by RedHat), and many others offer many kinds of OSS support services to realize their revenue generation goals, so GNUenterprise's developers have the potential to earn a living or make additional money from their FOSS development efforts.

What kinds of challenges can make the transition from EC/EB to Free EC/EB problematic or motivating, and how might these problems be mitigated via OSSD? Two broad categories of

challenges to Free EC/EB are apparent: those involving economic conditions such as those already noted, and those denoting structural or resource-based capabilities [Acedo, et al. 2006, Barney 2001, Hoopes, et al. 2003, and Oliver 1997]. Here the focus is on the later, and thus start with a description of the research methods employed in this study.

Research Methods

This study of GNUe arises from a longitudinal field study spanning 2002-2006. The study employed grounded theory techniques [Glaser and Strauss 1967, Strauss and Corbin 1980] including axial coding and construction of comparative memoranda based on field data collected through face-to-face and email-interviews, as well as extensive collection and cross-coding of publicly available project documents and software development artifacts posted on the project's Web site. These field study methods are subsequently closely aligned with those characterized as virtual ethnography [cf. Hakken 1999, Hine 2000, Scacchi 2002a] applied to software development projects [cf. Viller and Sommerville 2000] operating over the Internet/Web as a distributed virtual enterprise [Noll and Scacchi 1999]. A diverse set of work practices and socio-technical interaction processes emerged from the codings and their comparative analysis. These include how participants in different roles express their beliefs, norms, and values, as well as how they are enacted in shaping what free software development entails [Elliott and Scacchi 2003]. These in turn guide technical decision-making regarding which tools to employ during development activities, as well as how globally distributed participants act through cooperation and conflict to collectively to form (and re-form) GNUe as a virtual organization [Elliott and Scacchi 2005]. Finally, these practices also serve as a basis for articulating an occupational community of free software developers within the free software movement [Elliott and Scacchi 2003, 2006]. The study presented here extends and complements those just cited through a

reframing of the observed practices through data coding and institutionalized patterns that characterize the socio-technical resources and resource-based capabilities that support free software development work in GNUe. Finally, the analysis employs a variety of representational notations, relational schemes and flow diagrams [Scacchi, *et al.* 2006] to help articulate the results that are described next.

Resources and Capabilities for Developing Free EC/EB Software in GNUe

What kinds of resources or business capabilities are needed to help make Free EC/EB efforts more likely to succeed? How do these resources differ from those recommended in traditional software engineering projects? Based on what was observed in the GNUe case study, the following (unordered) set of socio-technical resources and capabilities enable the development of (a) free ERP and EC/EB software packages, as well as (b) the community that is sustaining its evolution, deployment and refinement, though other kinds of socio-technical processes also play a key role in mobilizing these resources into capabilities supporting work practices, and these are described elsewhere [Elliott and Scacchi, 2003, 2005, 2006, Scacchi 2005].

Personal software development tools and networking support

In GNUe, free software developers provide their own personal computer resources (often in their homes) in order to access or participate in the project. They similarly provide their own access to the Internet, and some even host personal Web sites or information repositories. Furthermore, these free software developers bring their own choice of tools (e.g., source code compliers, diagram editors) and development methods to the GNUe community, thought this seems to be common to many FOSS projects. There are few shared computing resources beyond the project's Web site, though its operation is supported in part by a company that provides a small number of

programmers to work on the GNUe software. Nonetheless, the sustained commitment of personal resources helps *subsidize* the emergence and evolution of the GNUe community, its shared (public) information artifacts, and resulting free software. It also helps create recognizable shares of the free software commons [cf. Benkler 2006, Lessig 2005] that are linked (via hardware, software, and Web) to the community's information infrastructure.

Beliefs supporting FOSS Development

Why do free software developers contribute their skill, time, and effort to the development of free software and related information resources? Though there are probably many diverse answers to such a question, it seems that one such answer must account for the belief in the freedom to access, study, modify, redistribute and share the evolving results from a FOSS development project. Without such belief, it seems unlikely that there could be "free" and "open source" software development projects [DiBona, Ockman and Stone, 1999, Pavlicek 2000, Williams 2002]. However, one important consideration that follows is what the consequences from such belief are, and how these consequences are put into action.

In looking across the case study data, in addition to examination of the online GNUe information resources from which they were taken [cf. Elliott and Scacchi 2003, 2005, 2006], many kinds of actions or choices emerge from the development of free software. Primary among them in the GNUe project (and possibly other FOSS projects) are *freedom of expression* and *freedom of choice*. Neither of these freedoms is explicitly declared, assured, or protected by free software copyright (the GNU Public License, GPL) or community intellectual property rights, or end-user license agreements[2]. However, they are central tenets free or open source modes of production

[2] EULAs associated with probably all software often seek to declare “freedom from liability” from people who want to use licensed software for intended or unintended purposes. But liability freedom is not the focus here.

and culture [Benkler 2006, Lessig 2005]. In particular, in FOSS projects like GNUenterprise and others, these additional freedoms are expressed in choices for what to develop or work on (e.g., choice of work subject or personal interest over work assignment), how to develop it (choice of method to use instead of a corporate standard), and what tools to employ (choice over which personal tools to employ versus only using what is provided). Consider the following excerpt from an online chat provided by someone (here identified with the pseudonym, ByronC) who was an outsider to the day-to-day development activities in the GNUe project seeking to determine if free (appropriate) or non-free (inappropriate) software tools were being used to create diagrams that help document and explain the how the GNUe software is organized:

> <BryonC> *Hello. Several images on the Website seem to be made with non-free Adobe software. I hope I am wrong; it is quite shocking. Does anybody know more on the subject? We should avoid using non-free software at all cost, am I wrong?*

Elsewhere, GNUe developers also expressed in choices for when to release work products (choice of satisfaction of work quality over schedule), determining what to review and when (modulated by community ownership responsibility), and expressing what can be said to whom with or without reservation (modulated by trust and accountability mechanisms). Shared belief and practice in these freedoms of expression and choice are part of the virtual organizational culture that characterizes a community project like GNUe [Elliott and Scacchi 2003, 2005]. Subsequently, putting these beliefs and cultural resources into action continues to build and reproduce socio-technical interactions networks that enabled sustained FOSS project community and the free software movement [Elliott and Scacchi 2006, Scacchi 2005].

Competently skilled and self-organizing software developers

Developing complex software modules for ERP applications requires skill and expertise in the domain of EB and EC. Developing these modules in a way that enables an open architecture

requires a base of prior experience in constructing open systems. The skilled use of project management tools for tracking and resolving open issues, and also for bug reports contribute to the development of such system architecture. These are among the valuable professional skills that are mobilized, brought to, or drawn to FOSS development community projects like GNUe [cf. Crowston and Scozzi 2002, Crowston and Howison 2005]. These skills are resources that FOSS developers bring to their projects, much like any traditional software development project.

FOSS developers organize their work as a virtual organizational form that seems to differ from what is common to in-house, centrally managed software development projects, which are commonly assumed in traditional software engineering textbooks [Sommerville 2004]. Within in-house development projects, software application developers and end-users often are juxtaposed in opposition to one another. Danziger [1979] referred to this concentration of software development skills, and the collective ability of an in-house development organization to control or mitigate the terms and conditions of system development as a "skill bureaucracy". Such software development skill bureaucracy (though still prevalent today) would seem to be mostly concerned with rule-following and rationalized decision-making, perhaps as guided by a "software development methodology" and its corresponding “interactive development environment” for software engineering.

In the decentralized virtual organization of a FOSS development community like GNUe, a "skill meritocracy" [cf. Fielding 1999] appears as an alternative to the skill bureaucracy. In such a meritocracy, there is no proprietary software development methodology or tool suite in use. Similarly, there are few explicit rules about what development tasks should be performed, who

should perform, when, why, or how. However, this is not to say there are no rules that serve to govern the project or collective action within it.

The rules of governance and control in the GNUe project are informally articulated but readily recognized by project participants. These rules serve to control the rights and privileges that developers share or delegate to one another in areas such as who can commit source code to the project's shared repository for release and redistribution [cf. Fogel 1999]. Similarly, rules of control are expressed and incorporated into the open source code itself in terms of how, where, and when to access system-managed data via application program interfaces, end-user interfaces, or other features or depictions of overall system architecture. But these rules may and do get changed through ongoing project development and online discourse carried out in the GNUe project's persistent online chat records.

Subsequently, GNUe project participants self- organize around the expertise, reputation, and accomplishments of core developers, secondary contributors, and tertiary reviewers and other volunteers. This in turn serves to help them create a logical basis for their collective action in developing the GNUe free software [cf. Olson 1971]. Thus, there is no assumption of a communal or egalitarian authority or utopian spirit. Instead what can be seen is a pragmatic, continuously negotiated order that tries to minimize the time and effort expended in mitigating decision-making conflicts while encouraging cooperation through reiterated and shared beliefs, values, and norms [Elliott and Scacchi 2005, Espinosa, *et al*., 2002].

In GNUe, participants nearer the core have greater control and discretionary decision-making authority, compared to those further from the core [cf. Lave and Wenger 1991, Crowston and Howison 2006]. However, realizing such authority comes at the price of higher commitment of personal resources described above. For example, being able to make a decision stick or to convince other community participants as to the viability of a decision, advocacy position, issue or bug report, also requires time, effort, communication, and creation of project content to substantiate such an action. Such articulation can be seen in the daily records of the project's online chat archive. The authority brought about through such articulation also reflects developer experience as an interested end-user of the software modules being developed. Thus, developers possessing and exercising such skill may be intrinsically motivated to sustain the evolutionary development of their free open source ERP and EC/EB software modules, so long as they are active participants in the GNUe project community.

Discretionary time and effort of developers

Are FOSS developers working for "free," or for advancing their career and professional development? Most of the core GNUe software developers have "day jobs" as software developers or consultants in companies, but few of these jobs specifically focus on the development of FOSS. So developing free software in the GNUe project is supported only in part for some of its core developers. Elsewhere, the survey results of Hars and Ou [2002] and others [Lerner and Tirole 2000, Hann, *et al*. 2002] suggest there are many personal and professional career oriented reasons for why participants will contribute their own personal (unpaid) time and effort to the sometimes difficult and demanding tasks of software development. What we have found in GNUe appears consistent with the cited observations. These include not only self-determination, peer recognition, community identification, and self-promotion, but also belief in

the inherent value of free software [cf. DiBona, Ockman, and Stone, 1999, Pavlicek 2000, Williams 2002].

In the practice of self-determination, no one has the administrative authority to tell a project member what to do, when, how, or why. GNUe developers can choose to work on what interests them personally, though their choices are limited to features or functions relevant to the ERP or EC/EB packages (or support libraries) they are developing. GNUe developers, in general, work on what they want, when they want, though the core developers do routinely connect to the project's chat room as a way to show up for work and to be visible to others.. However, they remain somewhat accountable to the inquiries, reviews, and messages of others in the community, particularly with regard to software modules or functions for which they have declared their responsibility to maintain or manage as a core developer.

In the practice of peer recognition, a GNUe developer becomes recognized as an increasingly valued community contributor as a growing number of their contributions make their way into the core software modules [Benkler 2006, Bergquist and Ljundberg 2001]. In addition, nearly two-thirds of FOSS developers work on 1-10 *additional* software projects [Hars and Ou 2002, Madey, *et al*, 2005], which also reflects a growing social network of alliances across multiple software development projects [cf. Monge, *et al.* 1998, Scacchi 2005]. The project contributors who span multiple free or non-free software project communities (identified as "linchpin developers" by Madey, *et al*., 2005) serve as "social gateways" that increase the GNUe community's mass [Marwell and Oliver 1993], as well as affording opportunities for inter-project software composition, bricolage, and interoperation [Jensen and Scacchi 2005]. For example,

some of the core developers choose to import and integrate a free project reporting system Previously in use in other software projects) to help keep track of outstanding GNUe software bugs, as well as how is working on what.

In building community identification, GNUe project participants build shared domain expertise, and identify who is expert in knowing how to do what [cf. Ackerman and Halverson 2000]. Interlinked information on the project's Web site, project development artifacts, and persistent online chat messages help point to who the experts and core contributors are within the projects socio-technical interaction network [Scacchi 2005].

In self-promotion, GNUe project participants communicate and share their experiences, perhaps from other application domains or work situations, about how to accomplish some task, or how to develop and advance through one's career. Being able to move from the project periphery towards the center or core of the development effort requires not only the time and effort of a contributor, but also the ability to communicate, learn from, and convince others as to the value or significance of the contributions [cf. Jensen and Scacchi 2006, Lave and Wegner 1991]. This is necessary when a participant's contribution is being questioned in open project communications, not incorporated (or "committed") within a new build version, or rejected by vote of those already recognized as core developers [cf. Fielding 1999].

The last source of discretionary time and effort observed in GNUe is found in the freedoms and beliefs in FOSSD that are shared, reiterated and put into observable interactions. If a community participant fails to sustain or reiterate the freedoms and beliefs codified in the GPL, then it is

likely the person's technical choice in the project may be called into question [Elliott and Scacchi 2003, 2005], or the person will leave the project and community. But understanding how these freedoms and beliefs are put into action points to another class of resources (sentimental resources) that must be mobilized and brought to bear in order to both develop FOSS systems and the global communities that surround and empower them. Social values that reinforce and sustain the project community, and technical norms regarding which software development tools and techniques to use (e.g., avoid the use of "non-free" software), are among the sentimental resources that are employed when participants seek to influence the choices that others in the project seek to uphold.

Trust and social accountability mechanisms

Developing complex software modules for ERP and EC/EB applications requires trust and accountability among GNUe project participants. Though trust and accountability in a FOSS project may be invisible resources, ongoing software and community development work occur only when these intangible resources and mechanisms for social control are present [cf. Gallivan 2001, Hertzum 2002].

The intangible resources of trust and accountability in GNUe arise in many forms. They include assuming ownership or responsibility of a community software module, voting on the approval of individual action or contribution to community software [Fielding 1999], shared peer reviewing [DiBona, Ockman and Stone 1999, Benkler 2006], and by contributing gifts [Bergquist and Ljundberg 2001] that are reusable and modifiable public goods [Olsen 1971, Samuelson 1954, Lessig 2005]. They also exist through the community's recognition of a core developer's status, reputation, and geek fame [Pavlicek 2000]. Without these attributions, GNUe

developers may lack the credibility they need to bring conflicts over how best to proceed to some accommodating resolution. Finally, as the GNUe project has been sustained (though with turnover) for over five years in terms of the number of contributing developers, end-users, and external sponsors, then GNUe's socio-technical mass (i.e., web of interacting resources) has become sufficient to insure that individual developer trust and accountability to the project community are sustained and evolving [Marwell and Oliver 1993].

Thus, the GNUe participants rely on mechanisms and conditions they have created for gentle but sufficient social control that helps constrain the overall complexity of the project. These constraints act in lieu of an explicit administrative authority or project management regime that would schedule, budget, staff, and control the project's development trajectory with varying degrees of administrative authority and technical competence, as would be found in a traditional software engineering project [cf. Sommerville 2004].

Free open source software development informalisms

Software informalisms [Scacchi 2002a] are the information resources and artifacts that participants use to describe, proscribe, or prescribe what's happening in a FOSSD project. They are informal narrative resources (or online document genres [cf. Kwansik and Crowston 2005]) that are comparatively easy to use, and immediately familiar to those who want to join the community project. However, the contents they embody require extensive review and comprehension by a developer before core contributions can be made. The most common informalisms used in GNUe include (i) community communications and messages within project Email, (ii) threaded message discussion forums or group blogs, (iii) news postings, (iv) community digests, and (v) instant messaging or Internet relay chat. They also include (vi)

scenarios of usage as linked Web pages, (vii) how-to guides, (viii) to-do lists, (ix) FAQs, and other itemized lists, and (x) project Wikis, as well as (xi) traditional system documentation and (xii) external publications. Free software (xiii) community property licenses also help to define what software or related project content are protected resources, so that they can subsequently be shared, examined, modified, and redistributed. Finally, (xiv) open software architecture diagrams, (xv) intra-application functionality realized via scripting languages like Perl and PhP, and the ability to either (xvi) plug-in or (xvii) integrate software modules from other OSSD efforts, are all resources that are used informally, where or when needed according to the interests or actions of project participants.

All of the software informalisms are found or accessed from (xix) project related Web sites or portals (see Exhibit 1). These Web environments where most FOSS software informalisms can be found, accessed, studied, modified, and redistributed [Scacchi 2002a]. A Web presence helps make visible the GNUe community's information infrastructure and the array of information resources that populate it. These include FOSS development community project Web sites (e.g., SourgeForge.net, Savanah.org), community software Web sites (PhP-Nuke.org), as well as (xx) embedded project source code Webs (directories), (xxi) project repositories (CVS [Fogel 1999]), and (xxii) software bug reports and (xxiii) issue tracking data base (called DCL in GNUe).

Together, these two dozen or so types of software informalisms constitute a substantial yet continually evolving web of informal, semi-structured, or processable information resources within GNUe. This web results from the hyperlinking and cross-referencing that interrelate the contents of different informalisms together. Subsequently, these FOSS informalisms are

produced, used, consumed, or reused within GNUe. They also serve to act as both a distributed virtual repository of FOSS project assets, as well as the continually adapted distributed knowledge base through which project participants In GNUe evolve what they know about the software systems they develop and use.

FOSSD capability enabling free, open ERP and EC/EB systems

The array of social, technological, and informational resources that enable a FOSS development project like GNUe is substantial. However, they differ in kind and form from the traditional enterprise resources that are provided to support proprietary, closed source software systems. These traditional software engineering resources are money (budget), time (schedule), skilled (salaried) development staff, project managers (administrative authority), quality assurance (QA) and testing groups, documentation writers, computer hardware and network maintainers, and others [cf. Sommerville 2004]. Free software projects like GNUe seem to get by with comparatively small amounts of money, though subsidies of various kinds and sources are present and necessary. They also get by without explicit schedules, though larger projects may announce target release dates, as well as (partially) order which system functions or features will be included in some upcoming versions, for some target release. Further, they get by without a rule-making and decision-making authority of corporate project managers or enterprise executives, who may or may not be adept at empowering, coaching, or rewarding development staff to achieve corporate software development goals. Instead, in GNUe, participants rely on an implicit but frequently recited regime of beliefs, values, and norms that help organize cooperative activity and rationalize conflict mitigation [Elliott and Scacchi 2003, 2005]. The remaining resources are provided within a free software development effort via subsidies, sponsorship, or volunteer effort.

Thus, the resources for free software development efforts are different in kind, and in how they are arrayed and brought to bear when compared to a traditional software engineering effort. Free software project resources are not mobilized, allocated, or otherwise brought to bear in the manner traditional to the development of proprietary, closed source software systems. Hopefully, it should be clear that the differences being highlighted are not based simply on a comparison of functionality or features visible in the development or use of open vs. close source software products. As such, the resource-based capability for developing free software packages for ERP and EC/EB applications is different, though not necessarily more or less costly.

Discussion and Conclusions

Many questions about free software development remain unanswered or unexamined by this study. For example, it is unclear whether there must be a critical mass of salaried software developers whose job includes development or support of GNUe software, and if so, how many software developers this entails. Over the four years in the study of GNUe, the number and composition of core software developers has changed, in part in response to their changing interests and work situations. The project has not grown to the point where commercialization of the GNUe software has become an imperative or new venture start-up opportunity, as has happened with OSS ERP projects of Compiere, OpenMFG, and Openbravo. Thus, it is unclear whether GNUe will become a viable enterprise capable of hiring a professional or full-time staff, as well as engaging in contracted provision of installation, customization, and support services that normally accompany ERP and EC/EB software packages. Further comparative study of other free software projects and their approach for commercialization are needed. However, it does seem clear that GNUe has managed to sustain itself an a viable ongoing enterprise that

continues to develop and sustain free ERP and EC/EB software packages, which its developers use and deploy in their day jobs or consulting practices.

Beyond this, three conclusions can be drawn from the study, data, and analysis presented in this report. First, this chapter identifies many types of socio-technical resources and resource-based capabilities for Free EC/EB that may explain/predict (a) what's involved, (b) how it works, or (c) what conditions may shape the longer-term success or failure of such efforts. In simple terms, these resources include time, skill, effort, belief, personal and corporate subsidies, and community building on the part of those contributing as developers and users of Free EC/EB systems and techniques. Of these, *belief* in the freedoms that open source system development embraces, including freedom of choice and freedom of expression [Elliott and Scacchi 2003, 2005, 2006] appears central. Such belief in turn enables and affords the ongoing commitment, development, and articulation of a web of social, technological, and information resources that sustain a free software project, without the traditional administrative and financial resources found in traditional software development enterprises. Developers and users who believe in the promise and potential of Free ERP or EC/EB packages are willing to allocate (or volunteer) their time and apply their skills to make the effort of developing or using open source systems a viable and successful course of action. Thus companies seeking to invest in or exploit Free EC/EB techniques or systems must account for how it can most effectively cultivate a Free software culture, belief system, and community of practice, as part of their strategic choice.

Second, this is the first study to employ a resource-based view of a FOSS development project. The resource-based view of organizational capability and competitive advantage is the dominant

analytical lens employed in studies of organizational strategy and strategic management [cf. Acedo, *et al.* 2006, Barney 2001]. Why should people interested in FOSS development practices be concerned or interested in such a strategic perspective? Many reasons might be sighted in support, but attention here can be drawn to determining whether free software systems and development methods offer sustained or differentiated advantages over traditional software engineering approaches applied to the development of close-source, proprietary (non-free) software systems. If there are advantages that can be traced to the resource arrangements found in free software projects like GNUe, then these would be noteworthy findings, as well as a possible basis for further exploration and theorizing. Accordingly, in the GNUe case, resources like personal computing tools that help subsidize the development effort, beliefs that provide a cultural basis for making decisions about technical choices, trust and social accountability, discretionary software development work times, and the preferred use of software informalisms instead of software engineering formalisms like "requirements specifications" and "project management plans" all differentiate the practice of free software development from that advocated in traditional software engineering textbooks [e.g., Sommerville 2004].

Last, this study links free software with ERP and EC/EB. No prior case studies of ERP or EC/EB systems have identified or addressed whether or how free software (or open source software) methods might be used to develop or integrate EC/EB software packages, at least beyond the use of OSS Web servers or Web-site content management systems [Carbone and Stoddard 2001]. Thus, there is an opportunity for firms to begin considering whether these results merit timely consideration or exploratory investments in free software or OSS. For example, companies offering consumer products or high value, information technology based products and services

may begin to consider whether Free EC/EB capabilities that offer lower purchase prices, lower total cost of ownership, and higher quality [Scacchi 2002b] represent new market entry or new product differentiation opportunities. Similarly, companies may find that free/open source software represents a new, highly innovative approach to software product or application system development that marries the best capabilities from both private investment and collective action [von Hippel and von Grogh 2003, Olson 1971].

Acknowledgements: The research described in this report is supported by grants from the NSF Industry/University Research Cooperative CRITO Consortium, National Science Foundation #0083075, #0205679, #0205724, #0350754 and # 0534771, and from the Defense Acquisition University by contract N487650-27803. No endorsement implied.

References

Acedo, F.J., Barroso, C., and Galan, J.L. (2006). The Resource-based Theory: Dissemination and Main Trends, *Strategic Management J.,* 27(7), 621-636.

Ackerman, M. and Halverson, C. (2000). Reexamining Organizational Memory, *Communications ACM,* 43(1), 59-64, January.

Barney, J.B. (2001). Resource-Based Theories of Competitive Advantage: A Ten-Year Retrospective on the Resource-Based View, *J. Management*, 27(6), 643-650.
Benkler, Y. (2006). *The Wealth of Networks*, Yale University Press, New Haven, CT.

Bergquist, M. and Ljungberg, J. (2001). The Power of Gifts: Organizing Social Relationships in Open Source Communities, *Information Systems J.*, 11(4), 305-320.

Carbone, G. and Stoddard, D. (2001). *Open Source Enterprise Solutions: Developing an E-Business Strategy*, John Wiley and Sons, Inc. New York.

Crowston, K. and Howison, J. (2006). Hierarchy and centralization in free and open source software team communications, *Knowledge, Technology and Policy*, 18(4), 65-85, Winter.

Crowston, K. and Scozzi, B. (2002). Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development, *IEE Proceedings--Software*, 149(2), 3-17.

Curran, T.A. and Ladd, A. (2000). *SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management (2nd Edition)*, Prentice-Hall, Upper Saddle River, NJ.

CW360, (2002). JD Edwards Pushes Modular ERP, *ComputerWeekly*, 12 June 2002.

Danziger, J. (1979). The Skill Bureaucracy and Intraorganizational Control: The Case of the Data-Processing Unit, *Sociology of Work and Occupations*, 21(3), 206-218.

DiBona, C., Ockman, S., and Stone, M. (1999). *Open Sources: Voices from the Open Source Revolution*, O'Reilly Press. Sebastopol, CA.

Elliott, M. and Scacchi, W. (2003). Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration, *Proc. ACM Intern. Conf. Supporting Group Work (Group'03),* 21-30, Sanibel Island, FL, November.

Elliott, M. and Scacchi, W. (2005). Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture, in S. Koch (ed.), *Free/Open Source Software Development*, 152-172, Idea Publishing, Pittsburgh, PA.

Elliott, M. and Scacchi, W. (2006). Mobilization of Software Developers: The Free Software Movement, (submitted for publication).

Espinosa, J.A., Kraut, R.E., Slaughter, S.A., Lerch, J.F., Herbsleb, J.D., Mockus, A. (2002). Shared Mental Models, Familiarity, and Coordination: A Multi-method Study of Distributed Software Teams. *Intern. Conf. Information Systems*, 425-433, Barcelona, Spain, December.

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

Fielding, R. (1999). Shared Leadership in the Apache Project, *Communications ACM*, 42(4), 42-43, 1999.

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

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

German, D.(2003). The GNOME Project: A case study of open source, global software development, *Software Process—Improvement and Practice*, 8(4), 201-215.

Hann, I-H., Roberts, J., Slaughter, S.L. and Fielding, R. (2002). Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives, *Proc. 2nd Workshop on Open Source Software Engineering*, Orlando, FL, May.

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

Hakken, D. (1999). *Cyborgs@Cyberspace? An Ethnographer Looks at the Future*, Routledge, London.

Hertzum, M. (2001). The importance of trust in software engineers' assessment and choice of information sources, *Information and Organization*, 12(1), 1-18.

Hine, C.M. (2000). *Virtual Ethnography*, Sage Publications, Newbury Park, CA.

Hoopes, D.G., Madsen, T.L., and Walker, G. (2003). Why is there a Resource-based View? Toward a Theory of Competitive Heterogeneity, *Strategic Management J.,* 24(10), 889-902.

Jensen, C. and Scacchi, W. (2005). Process Modeling Across the Web Information Infrastructure, *Software Process—Improvement and Practice*, 10(4), 255-272.

Jensen, C. and Scacchi, W. (2006). Modeling Recruitment and Role Migration Processes in Open Source Software Development Projects, submitted for publication, April.

Keller, G. and Teufel, T. (1998). *SAP R/3 Process Oriented Implementation: Iterative Process Prototyping*, (translated by A. Weinland), Addison Wesley Longman, Harlow, England.

Kim, A.J. (2000). *Community Building on the Web: Secret Strategies of Successful Online Communities*, Peachpit Press, 2000.

Kwansik, B. and Crowston, K. (2005). Introduction to the special issue: Genres of Digital Documents, *Information, Technology and People,* 18(2).

Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*, Cambridge University Press, Cambridge, UK.

Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source, *Journal of Industrial Economics*, 52.

Lessig, L. (2005). *Free Culture: The Nature and Future of Creativity*, Penguin Press, New York.

Madey, G., Freeh, V., and Tynan, R. (2005). Modeling the F/OSS Community: A Quantative Investigation, in S. Koch (ed.), *Free/Open Source Software Development*, 203-221, Idea Group Publishing, Hershey, PA.

Marwell, G. and Oliver, P. (1993). *The Critical Mass in Collective Action : A Micro-Social Theory*. Cambridge University Press.

Monge, P.R., Fulk, J., Kalman, M.E., Flanagin, A.J., Parnassa, C., and Rumsey, S. (1998). Production of Collective Action in Alliance-Based Interorganizational Communication and Information Systems, *Organization Science*, 9(3): 411-433.

Noll, J. and Scacchi, W. (1999). Supporting Software Development in Virtual Enterprises, *Journal of Digital Information*, 1(4), February.

Oliver, C. (1997). Sustainable Competitive Advantage: Combining Institutional and Resource-Based Views, *Strategic Management J.,* 18(9), 697-713.

Olson, M. (1971). *The Logic of Collective Action*, Harvard University Press, Cambridge, MA.

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

Samuelson, P. (1954). The Pure Theory of Public Expenditure. *Review of Economics and Statistics*, 36:387-390.

Scacchi, W. (2001). Redesigning Contracted Service Procurement for Internet-based Electronic Commerce: A Case Study, *J. Information Technology and Management*, 2(3), 313-334.

Scacchi, W. (2002a). Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings--Software*, 149(2), 24-39.

Scacchi, W. (2002b). Is Open Source Software Development Faster, Better, and Cheaper than Software Engineering?, *Proceedings 2nd Workshop on Open Source Software Engineering,* Orlando, FL.

Scacchi, W. (2004). Free/Open Source Software Development Practices in the Game Community, *IEEE Software*, 21(1), 59-67, January/February.

Scacchi, W., (2005). Socio-Technical Interaction Networks in Free/Open Source Software Development Processes, in S.T. Acuña and N. Juristo (eds.), *Software Process Modeling*, 1-27, Springer Science+Business Media Inc., New York.

Sharman, S., Sugurmaran, V., and Rajagopalan, B. (2002). A Framework for Creating Hybrid-Open Source Software Communities, *Information Systems J.,* 12(1), 7-25.

Skok, W. and Legge, M. (2002). Evaluating Enterprise Resource Planning (ERP) System using an Interpretive Approach, *Knowledge and Process Management*, 9(2), 72-82.

Sommerville, I. (2004). *Software Engineering (7th Edition)*, Addison-Wesley, New York.

Viller, S. and Sommerville, I. (2000). Ethnographically Informed Analysis for Software Engineers, *Intern. J. Human-Computer Studies*, 53, 169-196.

von Hippel, E. and von Krogh, G. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science, *Organization Science*, 14(2), 209-223.

West, J. and O'Mahony, S. (2005). Contrasting Community Building in Sponsored and Community Founded Open Source Projects, *Proc. 38th. Hawaii Intern. Conf. Systems Sciences*, Waikola Village, HI.

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