OpenEC/B: Electronic Commerce and Free/Open Source Software Development

Walt Scacchi
Institute for Software Research
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425 USA
http://www.ics.uci.edu/~wscacchi

ABSTRACT
This report investigates Open Source E-Commerce or E-Business capabilities. This entails a case study within one firm that has undertaken an organizational initiative to develop, deploy, use, and support free/open source software systems for Enterprise Resource Planning (ERP), E-Commerce (EC) or E-Business (EB) services. The objective is to identify and characterize the resource-based software product development capabilities that lie at the center of the initiative.

1. INTRODUCTION
This paper presents and analyzes a case study that examines how a firm can support an E-Commerce or E-Business initiative that builds from free/open source software (FOSS) product development capabilities. Such capabilities may focus, for example, on back office activities associated with corporate financial operations, or on front office activities associated with customer relationship management. Alternatively, the focus may be directed as an organizational system where wireless, mobile, or p2p capabilities are sought.

The study employs a resource-based view of the organizational system involved in developing an open source EC/EB software products or application systems. The analysis and results of the case study focus attention to data that characterizes the organization's resource-based product development capabilities. This case study examines the GNUenterprise.org project. This study serves as a point of departure to explicate the concept of Open EC/B introduced in this paper. Open EC/B results from combining OSSD concepts, techniques, and tools with those for EC and EB.

Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE) May 17, 2005, St Louis, MO, USA.

2. Case Study: GNUenterprise.org and the development of FOSS ERP software
GNUenterprise.org 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 a free, open source Enterprise Resource Planning (ERP) systems and related E-Business capabilities.

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 most large firms. Nonetheless, we still need a better understanding of what resource-based capabilities are brought to bear on the development and deployment of EB and ERP software by GNUenterprise.org. Subsequently, what follows is a description of key resources being employed throughout GNUenterprise.org to develop and support the evolution of the GNUe software modules.

The following sections present 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, 2002, Skok and Legge 2002]. One category of challenges to Open EC/B that is apparent are those denoting resource-based capabilities.

3. Resources and Capabilities for Open EC/B
In this section, the two different contexts in which Plone has been reused are described. Firstly, it has formed the basis for the collaboration and communication infrastructure for the EU FP6 Coordination Action for Libre Software Engineering for Open Development Platforms for Software and Services ("CALIBRE"[1]) project. Secondly, based on this earlier use, it has been adapted as the content management system for the newly-formed British Computer Society Open Source Specialist Group. These two contexts differ greatly in their requirements; however the flexibility of Plone has allowed the same system to be adapted to meet the requirements in both cases.

What kinds of resources or business capabilities are needed to help make Open EC/B efforts more likely to succeed? Based on what was observed in the GNUenterprise.org case study, the following kinds of resources enable the development of both

[1]http://calibre.ie/

FOSS ERP/EB software and community that is sustaining its evolution, application and refinement:

3.1 Personal software development tools and networking support

FOSS developers, end-users, and other volunteers provide their own personal computing resources in order to access or participate in a FOSSD community project. They similarly provide their own access to the Internet, and may even host personal Web sites or information repositories. Furthermore, FOSS developers bring their own choice of tools and development methods to the community. The sustained commitment of personal resources helps subsidize the emergence and evolution of the community, and its shared (public) information resources. It also helps create recognizable shares of the commons that are linked (via hardware, software, and Web) to community infrastructure.

3.2 Beliefs supporting FOSSD

Why do software developers and others contribute their skill, time, and effort to the development of FOSS 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 share, learn, modify, and redistribute the evolving results from a FOSSD project. Without such belief, it seems unlikely that there could be "free" and "open source" software development projects [DiBona, Ockman and Stone, 1999, Williams 2002]. However, one important consideration that follows is what are the consequences from such belief, and how are these consequences put into action.

In looking across the case study data, many kinds of actions or choices emerge from the development of FOSS. Primary among them is freedom of expression and choice. Neither of these freedoms is explicitly declared, assured, or protected by free software copyright or community intellectual property rights. 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 (personal tool choice versus only using what is provided). They also are 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). Shared belief and practice in freedom of expression and choice are part of the organizational culture that characterizes a community project like GNUenterprise.org [Elliott and Scacchi 2004]. Subsequently, putting these beliefs and cultural resources into action builds both community and FOSS.

3.3 Competently skilled and self-organizing FOSS 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 bug reports also contributes to the development of such a system architecture. These are among the valuable professional skills that are mobilized, brought to, or drawn to FOSSD community projects like GNUenterprise.org [cf. Crowston and Scozzi 2002]. These skills are resources that FOSS developers bring to their projects.

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. 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 a software development skill bureaucracy would seem to be mostly concerned with rule-following and rationalized decision-making, perhaps as guided by a "software development methodology" and its corresponding computer-aided software engineering tool suite.

In the decentralized virtual organization of a FOSSD community like GNUenterprise.org, 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. Instead, FOSSD participants organize around the expertise, reputation, and accomplishments of core developers, secondary contributors, and tertiary reviewers and other volunteers.

Participants nearer the core have greater control and discretionary decision-making authority, compared to those further from the core. However, realizing such authority comes at the price of higher commitment of personal resources described above. 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. This authority 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 EB software modules, so long as they are active participants in their community project.

3.4 Discretionary time and effort of developers

Are OSS developers working for "free" or for advancing their career and professional development? Following the survey results of Hars and Ou [2002] and others [Lerner and Tirole 2000, Hann, et al. 2002], there are many personal and professional career oriented reasons for why participants will contribute their time and effort to the sometimes difficult and demanding tasks of software development. What we have found in GNUenterprise.org appears consistent with their observations. These include include not only self-determination, peer recognition, community identification, and self-promotion, but also belief in inherent value of free software [cf. DiBona, Ockman, and Stone, 1999, 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. OSS developers can choose to work on what interests them personally. FOSS developers, in general, work on what they want, when they want. However, they remain somewhat accountable to the inquiries, reviews, and messages of others in the community, particularly with regard to software modules for which they have declared responsibility to maintain or manage as a core developer.

In the practice of peer recognition, a developer becomes recognized as an increasingly valued community contributor as a growing number of their contributions make their way into the core software modules [Bergquist and Ljundberg 2001]. In addition, nearly two-thirds of OSS developers work on 1-10 additional OSSD projects [Hars and Ou 2002], which also reflects a growing social network of alliances across multiple free, OSSD projects [cf. Monge, et al. 1998]. The project contributors who span multiple project communities can serve as "social gateways" that increase the community's mass [Marwell and Oliver 1993] and opportunity for inter-project software composition and bricolage. It also enables and empowers their recognition across multiple communities of FOSSD peers.

In building community identification, project participants build shared domain expertise, and identify who is expert in knowing how to do what [cf. Ackerman and Halverson 2000]. Interlinked contents and persistent communicated messages help point to who the experts and core contributors are.

In self-promotion, 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 towards the center or core of the development effort requires not only the time and effort of a contributor, but also the ability to convince others as to the significance of the contributions. 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 GNUenterprise.org 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 institutionalized in the GPL, then it is likely the person will leave the project and community. But understanding how these freedoms and beliefs are put into action points to another class of (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.

3.5 Trust and social accountability mechanisms

Developing complex software modules for ERP, EB, or EC applications requires trust and accountability among project participants. Though trust and accountability in a FOSSD 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. Hertzum 2002].

The intangible resources 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], and by contributing gifts [Bergquist and Ljundberg 2001] that are reusable and modifiable public goods [Olson 1971]. They also exist through the community's recognition of a core developer's status, reputation, and geek fame [Pavlicek 2000]. Without these attributions, developers may lack the credibility they need to bring conflicts over how best to proceed to some accommodating resolution. Finally, as a FOSSD project grows in terms of the number of contributing developers, end-users, and external sponsors, then community's mass becomes sufficient to insure that individual trust and accountability to the project community are sustained and evolving [Marwell and Oliver 1993].

Thus, FOSSD efforts rely on mechanisms and conditions 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.

3.6 FOSSD informalisms

Software informalisms [Scacchi 2002] are the information resources and artifacts that participants use to describe, proscribe, or prescribe what's happening in a FOSSD project. They are informal resources 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 include community communications and messages within Email, threaded Email discussion forum, news postings, community digests, and instant messaging chat. They also include scenarios of usage as linked Web pages, how-to guides, to-do lists, FAQs, and other itemized lists, as well as traditional system documentation and external publications. FOSS community property licenses also help to define what software or related project content are protected resources that can subsequently be shared, examined, modified, and redistributed. Finally, open software architectural designs, scripting languages like Perl and PhP, and the ability to either plug-in or 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 project related Web sites or portals. These Web environments are also software informalisms [Scacchi 2002]. A project's Web presence helps make visible the community's information infrastructure and the array of information resources that populate it. These include OSSD community project Web sites (e.g., SourgeForge.net, Savanah.org, and Freshment.org), community

software Web sites (PhP-Nuke.org), and project Web site (www.GNUenterprise.org), as well as embedded project source code Webs (directories), project repositories, and software bug reports and issue tracking data base.

Together, these software informalisms constitute a substantial collection of information resources and artifacts that are produced, used, consumed, or reused within and across FOSSD projects.

3.7 FOSSD capability enabling free, open ERP and EB systems

The array of social, technological, and informational resources that enable a FOSSD project 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 resources are money (budget), time (schedule), skilled development staff, project managers (administrative authority), quality assurance (QA) and testing groups, documentation writers, computer hardware and network maintainers, and others. FOSSD projects 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 the rule-making and decision-making authority of project managers, who may or may not be adept at empowering, coaching, or rewarding development staff to achieve corporate software development goals. The remaining resources are provided within a FOSSD effort via subsidies, sponsorship, or volunteer effort.

Thus, the resources for FOSSD efforts are different: they 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 FOSS components or modules for ERP, EB and EC applications is different.

4. CONCLUSIONS

Two main conclusions can be drawn from the study, data, and analysis presented in this report.

First, this study identified and introduced a new concept called OpenEC/B. OpenEC/B denotes the integration of FOSSD resources, products, and processes, with the existing or emerging capabilities for Electronic Commerce/Business. This concept and its consequences are explained in the case study and analysis. No prior case studies of EC/EB have identified or addressed whether or how OSS methods might be applied or integrated with EC/EB, at least beyond the use of OSS Web servers or Web-site content management systems. Thus, there is an opportunity for firms to begin considering whether these results merit timely consideration or exploratory investments. For example, companies offering consumer products or high value, information technology based products and services may begin to consider whether OpenEC/B capabilities that offer lower purchase prices, lower total cost of ownership, and higher quality represent new market entry or new product differentiation opportunities. Similarly, companies may find FOSSD represents a highly innovative approach to software product development that marries the best capabilities from both private investment and collective action [Marwell and Oliver 1993, Olson 1971, von Hippel and von Krogh 2003]

Last, this study identifies resources and resource-based capability for OpenEC/B that may explain or 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 OpenEC/B systems and techniques. Of these, belief in the freedoms that open source system development allows appears central. Developers and users who believe in the promise and potential of OpenEC/B systems 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 OpenEC/B techniques or systems must account for how it can most effectively cultivate an OpenEC/B culture, belief system, and community of practice, as part of their strategic choice.

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, and #0350754 and from the Defense Acquisition University by contract N487650-27803. No endorsement implied..

References

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

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

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

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

[5] C. DiBona, S. Ockman and M. Stone, *Open Sources: Voices from the Open Source Revolution*, O'Reilly Press, 1999.

[6] M. Elliott and W. Scacchi, 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, 2004.

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

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

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

[10] M. Hertzum, The importance of trust in software engineers' assessment and choice of information sources, *Information and Organization*, 12(1), 1-18, 2002.

[11] J.Lerner and J. Tirole, Some Simple Economics of Open Source, *Journal of Industrial Economics*, 52, 2002.

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

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

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

[15] M. Olson, *The Logic of Collective Action*, Harvard University Press, Cambridge, MA, 1971.

[16] R. Pavlicek, *Embracing Insanity: Open Source Software Development*, SAMS Publishing, Indianapolis, IN, 2000.

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

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

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

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

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