A FRAMEWORK ANALYSIS OF THE OPEN SOURCE SOFTWARE DEVELOPMENT PARADIGM

Joseph Feller
Brian Fitzgerald
University College Cork
Ireland

Abstract

Open Source Software (OSS) has become the subject of much commercial interest of late. Certainly, OSS seems to hold much promise in addressing the core issues of the software crisis, namely that of software taking too long to develop, exceeding its budget, and not working very well. Indeed, there have been several examples of significant OSS success stories—the Linux operating system, the Apache web server, the BIND domain name resolution utility, to name but a few. However, little by way of rigorous academic research on OSS has been conducted to date. In this study, a framework was derived from two previous frameworks which have been very influential in the IS field, namely that of Zachman's IS architecture (ISA) and Checkland's CATWOE framework from Soft Systems Methodology (SSM). The resulting framework is used to analyze the OSS approach in detail. The potential future of OSS research is also discussed.

1. INTRODUCTION

The term "software crisis," first coined in 1968, may be summarized as systems taking too long to develop, costing too much, and not working very well when delivered. In the intervening 32 years, many "silver bullet" solutions to this crisis have been proposed (Brooks 1987). While not classifying the Open Source Software (OSS) development approach as a silver bullet, the case can be made that OSS addresses many aspects of the software crisis, in that reliable, high quality software may be produced quickly and inexpensively. Since the OSS term was coined in 1997, little rigorous academic research has been conducted on the topic. The reports that have been published tend to be somewhat evangelical, concentrating on the genuine success stories associated with the OSS approach. In this paper, we draw on research by Zachman into IS development architectures (Zachman 1987; Sowa and Zachman 1992) and work by Checkland (1981) on Soft Systems Methodology (SSM) to derive a framework with which to analyze the OSS development approach in detail.

The remainder of the paper is structured as follows. Section 2 defines the OSS approach and discusses its origins. Section 3 contains the derivation of the analytical framework. Section 4 analyses the OSS approach using the framework. Section 5 briefly compares OSS with Rapid Application Development (RAD). Finally, the potential future of OSS research is discussed.

2. THE OPEN SOURCE SOFTWARE DEVELOPMENT APPROACH

Open Source Software is software released under a license conforming to the Open Source Definition (OSD), as articulated by the Open Source Initiative.[1] Commercial software is generally sold for a fee, with restrictions on use, and in binary form only (e.g., Microsoft Office). By contrast, software conforming to the Open Source Definition meets the conditions outlined in Table 1.

[1]"Open Source" is a certification mark owned by the Open Source Initiative (http://www.opensource.org).

Table 1. Outline of Key Conditions of Open Source Definition[2]

Condition	Commentary
The source code must be available to user.	The software distribution must include the source code (i.e., the original programming language), or else the code must be made available by free, public Internet download.
The software must be redistributable.	The user of an OSS release is given full rights to reproduce and redistribute the software, on any medium, to any party, either gratis or for a fee.
The software must be modifiable, and the creation of derivative works must be permitted.	All users are given the right to modify the software or produce derivative works. There is considerable variation among licenses regarding whether or not modifications must also be released publicly under an OSD compliant license.
The license must not discriminate against any user, group of users, or field of endeavor.	In an attempt to counter overtly ideological content in software licenses, the OSD precludes any limitations on the possible uses of an OSS distribution.
The license must apply to all parties to whom the software is distributed.	While some licenses might allow modifications to be released under a non-compliant license, an OSS distribution cannot be "relicensed" by the user.
The license cannot restrict aggregations of software.	OSD compliant licenses cannot be limited to a particular distribution, nor can they seek to contaminate separately licensed software with which it is aggregated.

Open Source Software should not be confused with "freeware" products, "shareware" products, "use-restricted" products, or "royalty-free" products. Table 2 summarizes the differences between these various categories of software.

While the term Open Source Software was only recently coined, "free" software has a long history. The earliest programmers shared any software that was developed, a good analogy perhaps being that of recipe sharing or cooperative barn raising. However, as hardware vendors began to dominate the software market, proprietary software distributed in binary code form at a significant cost became the norm (McKusick 1999).

There have been many initiatives endeavoring to re-establish the free software norm. The Berkeley System Distribution (BSD) sought to distribute software to interested parties (although with some political idealism—BSD products were not made available to South Africa during the apartheid regime). Another very significant initiative was Richard Stallman's Free Software Foundation (FSF), and the creation of the GNU[3] Public License, in 1985. The FSF policy is strongly ideological and is captured in the recasting of the common copyright phrase to read "*copyleft—all rights reversed*" (Stallman 1999). Given the idealism of such initiatives, they were deemed (generally erroneously) to be anti-commercial and hostile to any profit motive (Perens 1999). There was also some confusion over the term free. The term is intended to indicate free as in *libertas*, e.g., free speech. It is not intended to mean free as in *gratis,* e.g., free beer (DiBona et al. 1999). For the commercial community, the latter connotation was generally assumed. Thus, the Open Source Software term was coined to eliminate this confusion in an attempt to remove potential obstacles to the commercial expansion of the concept. The renaming has not been without conflict, and considerable contestation exists between the FSF and the Open Source Initiative (see Raymond 1999b).

The OSS movement has provided many examples of spectacularly successful projects. The Linux operating system is perhaps the most well-known—indeed, it is assumed by many to be synonymous with the OSS approach. The Linux project was begun in 1991 by Linus Torvalds, a Finnish university student. From these humble beginnings, Linux now holds 25% of the general server market (Mitchell 2000b) and 34% of the Internet/Web server market (Mitchell 2000a). A survey carried out by the IDC group predicts that its market-share will grow faster than all other operating systems combined through 2003 (Raymond 1999a). Another notable OSS product is the Apache web-server, begun in February 1995 by a group of web software developers who decided to pool their expertise rather than duplicate work by reinventing the wheel in isolation. As of March 2000, Apache had achieved an astonishing 60% of the web-server market (Speedie 2000).

[2]The authoritative Open Source Definition can be found at http://www.opensource.org/osd.html.

[3]A recursive acronym, ever popular with the hacking community, for *GNU's Not Unix.*

Table 2. A Software Licensing Taxonomy (Adapted from The Halloween Documents, 1998[4])

License Feature / Software Type	Zero Price	Redistributable	Unlimited Users and Usage	Source Code Available	Source Code Modifiable
Commercial (e.g., typical Microsoft products)					
Trial Software (e.g., time-bombed evaluation products)	X	X			
Use-Restricted (e.g., Netscape Navigator, freely available and re-distributable only to non-profit-making entities)	X	X			
Shareware (e.g., WinZip, which have a license that eventually mandates purchase)	X	X			
Freeware (e.g., Leap Frog, released in binary form only and in the public domain)	X	X	X		
Royalty-free binaries (e.g., Microsoft's Internet Explorer and NetMeeting, distributed in binary form only)	X	X	X		
Royalty-free libraries (e.g., class libraries)	X	X	X	X	
Open Source (e.g., Linux, Apache)	X	X	X	X	X

O'Reilly (2000) points out that although Linux and Apache are high profile, the more significant success of OSS can be seen behind the scenes, in the Internet "category killers" such as BIND, which runs the Domain Name Server (DNS) for the World Wide Web, Sendmail, which is central to the Internet mail backbone, and Perl, still the dominant language for CGI (Common Gateway Interface) programming. Also, it should be recognized that Linux is only an operating system kernel and much of the functionality of a Linux system is derived from the GNU family of utilities which emerged from Stallman's Free Software Foundation.

Mainstream corporate interest in the topic has also increased exponentially, with IBM, Apple, Oracle, Cobalt, Corel, SGI, Intel, and Ericsson all taking various initiatives which support the growth of OSS. This interest may have been stimulated to a significant extent by the example of Netscape's Mozilla OSS project. The Mozilla project stemmed from the predicament Netscape found themselves in as of early 1998: their Navigator browser product was at 13% market-share and losing ground, principally to Microsoft's Internet Explorer. Drastic action was deemed necessary, and Netscape released the code of their browser under a parallel, open development scheme called Mozilla. This had a positive effect: by November 1998, Netscape was regaining market-share, and following their take-over by America-Online (AOL), there has been a public commitment to remain Open Source. The Mozilla example thus raised the profile of the OSS approach in the corporate world.

Having established the origins of the OSS approach and discussed some significant examples, in the next section, a framework is derived to analyze the OSS approach in more detail.

[4]"The Halloween Documents" refers to a set of confidential internal Microsoft memos, leaked to the OSS community at end of October 1998 (hence the Halloween title), which discuss the concerns of Microsoft at the threat posed by OSS.

3. A FRAMEWORK FOR ANALYZING THE OSS DEVELOPMENT APPROACH

Zachman has drawn on the disciplines of architecture and engineering to derive a framework for IS architecture which basically contains the categories *what, how, where, who, when,* and *why* (a set of descriptors which appear to have been borrowed from Kipling). Zachman discusses in some detail the descriptors *what, how,* and *where* to categorize different IS architectures and suggests that these are independent but "inextricably linked," and suggests that, for the sake of logical completeness, these should be complemented by *who, when* and *why* (Zachman 1987; Sowa and Zachman 1992). Interestingly, Zachman concluded the 1987 paper by suggesting that the framework could be used in a number of areas, including to rethink the nature of software development.

Checkland (1981) has proposed the CATWOE technique as part of Soft Systems Method (SSM) and it is described in some detail. Basically, CATWOE is a mnemonic for Client, Actor, Transformation, *Weltanschauung*, Owner and Environment:

- Client being the beneficiary affected by an activity, i.e., the *whom*;
- Actor is the agent of change who carries out the transformation, i.e., the *who;*
- Transformation refers to the change taking place, i.e., the *what;*
- *Weltanschauung* is the world-view or assumptions which underpin the process—broadly speaking, it addresses the *why* question;
- Owner is the sponsor of the activity;
- Environment is the wider system of which the activity is a part—broadly speaking, this addresses the *where* and *when* questions.

Zachman's framework is strong in the area of functions and processes and has separate categories for *how* and *when* issues, which do not appear explicitly in Checkland's CATWOE. However, CATWOE has a greater emphasis on people, perhaps unsurprising given its origins. Thus, Zachman's *who* category is significantly elaborated in the CATWOE elements of *client*, *actor*, and *owner*. In terms of the remaining categories, it can be argued that there is quite a close correspondence between the elements. Thus, Checkland's *environment* aligns with Zachman's *where* and *when* categories. Similarly, *transformation* and *what* can be paired, as can *weltanschauung* and *why*. The resulting framework categories and their operationalization for this research are depicted in Table 3.

Table 3. A Framework for Analyzing the OSS Approach

What (Transformation) • What defines a software project as OSS? • What types of projects tend to be OSS?
Why (Weltanshauung, or World View) • What are the technological motivations for OSS development? • What are the economic motivations for OSS development? • What are the socio-political motivations for OSS development?
When and Where (Environment) • What are the temporal dimensions of OSS development? • What are the spatial/geographic dimensions of OSS development?
How • How is the OSS development process organized? • What tools are used to support the OSS model?
Who (Client, Actor, Owner) • What are the characteristics of the individual developers contributing to OSS projects? • What are the characteristics of the companies distributing OSS products? • What are the characteristics of the users of OSS products?

4. ANALYSIS OF THE OSS APPROACH

In this section, the OSS approach is analyzed in detail using each of the categories of the derived framework. The overall analysis is summarized in Table 4, and the analysis of each of the categories is discussed in the following sub-sections.

Table 4. Framework Analysis of the OSS Approach

What (Transformation)	
What defines a software product/project as OSS?	Open Source Software is strictly defined by the license under which it is distributed (i.e., compliance with the OSI Open Source Definition). However, as with any emerging concept, there is some fluidity. Thus, OSS is further characterized by the dynamics described subsequently in this framework analysis.
What types of products/projects tend to be OSS?	OSS has in the past been dominated by operating and networking systems software, utilities, development tools, and infrastructural components. Currently, an increasing number of productivity and entertainment applications are being developed.
Why (Weltanschauung, or World View)	
What are the technological motivations for OSS development?	The primary technological drivers for OSS include the need for more robust code, faster development cycles, higher standards of quality, reliability and stability, and more open standards/platforms.
What are the economic motivations for OSS development?	Business drivers for OSS include the corporate need for shared cost and shared risk, and the redefinition of software industry as a commodity and service industry.
What are the socio-political motivations for OSS development?	"Human" motivations for OSS include scratching a developer's "personal itch," the desire for advancement through mentorship, peer reputation, the desire for "meaningful" work, and community oriented idealism.
When and Where (Environment)	
What are the temporal dimensions of OSS development?	OSS is characterized by the rapid development and rapid evolution of software, by frequent, incremental release, and by interaction in "Internet time."
What are the spatial/geographic dimensions of OSS development?	OSS is characterized by distributed developer teams, bounded by "cyberspace" rather than physical geography.
How	
How is the OSS development process organized?	The core methodology of OSS is massive parallel development and debugging. This has traditionally involved loosely-centralized, cooperative, and *gratis* contribution from individual developers (although there is a recent increase in paid, coordinated development).
What tools are used to support the OSS model?	Massive parallel development methods are supported by the Internet as a communication, collaboration, and distribution platform, and by concurrent versioning software. Other support tools include academic, non-profit, and commercial patrons of OSS projects, "reverse auctions" and online agency services.
Who (Client, Actor, Owner)	
What are the characteristics of the individual developers contributing to OSS projects?	OSS developers have traditionally been self-identified hackers (not "crackers"), professional developers (not amateurs), self-selected, and highly-motivated. Also, paradoxically, given the reputation-based culture, developers tend to be (publicly) modest and self-deprecating. This has important implications for stimulating cooperative development.
What are the characteristics of the companies distributing OSS products?	OSS companies have had a fantastic IPO stock market history. Generally, these companies provide patronage for OSS developers akin to a Renaissance model. Profitability hinges on a paradigm shift in software development. Customer service, brand management, and peer reputation are critical. Parallels exist with the automobile and legal industries and with academe.
What are the characteristics of the users of OSS products?	To date, OSS users have primarily been expert users and early adopters. Traditionally there has been considerable overlap between the developer and user pool. As more OSS projects focus on usability and interface issues, this profile will change.

4.1 Analysis of the What of OSS

The *what* of OSS, and its link to the Open Source Definition, has already been discussed to some extent in section 2, where the concept was defined and its origins discussed. Reiterating briefly, the OSS approach makes the source code of software available and protects the unconditional right of collaborating developers to modify and redistribute Open Source Software. The approach involves the massively parallel, often decentralized and *gratis* contribution of literally thousands of globally-located developers. Some of the more notable OSS examples are listed in Table 5.

Table 5. Examples of OSS Products

Product	Description
• *Linux*	• Operating System
• *Apache*	• Web server
• *Sendmail*	• Internet mail utility
• *BIND*	• Berkeley Internet Name Daemon, the software that runs the Domain Name Server (DNS) for the Web
• *PERL, Python*	• Programming languages
• *GNU Software*	• A variety of compilers, utilities, and notably, the EMACS Editor
• *GNOME, KDE*	• GUI interfaces for Linux
• *Mozilla*	• OSS version of Netscape Navigator
• *Jikes*	• Java compiler from IBM
• *GIMP*	• Image workshop
• *Darwin*	• Mac OSX server system
• *SAMBA*	• Allows integration with Microsoft's SMB protocol
• *Ghostscript*	• Aladdin enterprises PDF utility
• *Doom*	• First-person shooter game

As can be seen from Table 5, OSS products are typically development tools, back-office services/applications, and infrastructural and networking utilities. Performance and reliability are critical factors and these products score very highly on these criteria. They have been chosen by technically-aware IT personnel who are not as susceptible to a glossy marketing campaign as the wider market. Also, it is interesting that a lot of the software listed in Table 5 is Internet or Web-related. This reflects the OSS-like nature of the evolution of the Internet, e.g., the TCP/IP protocol, DNS, and electronic mail utilities. Also, the initial implementation of the Web was characterized by an OSS style; thus, it is appropriate that utilities such as BIND and the Apache web server should be OSS. It should be noted that as OSS-related companies seek to push the movement into the commercial mainstream, entertainment and end-user productivity tools are becoming more popular.

4.2 Analysis of the Why of OSS

There are three "world views" important to understanding why, on a macro level, industry might choose to develop software according to the OSS paradigm, and, on a micro level, why individual developers would choose to participate. These are discussed in turn.

The technological motivation for OSS development directly relates to the software crisis, which clearly illustrates that traditional modes of development do not work very well, specifically in the areas of speed, quality, and cost of development. The OSS approach counters these tripartite aspects. The Linux operating system and other OSS products mentioned above are characterized by a very rapid development time-scale. For example, new releases of Linux were produced more than once per day in the early days of its development (Raymond 1998a). Part of the conventional wisdom of software development is captured in Brooks' fundamental law, viz., "adding manpower to a late software project makes it later" (Brooks 1975). Based on this, it was thought that complex software, which all software pretty much is (Brooks 1987), required a disciplined and orderly approach, and that

the communication overhead of adding extra developers negated their potential contribution to development productivity. However, the OSS community has recast this law as "given enough eye-balls, every bug looks shallow" (Raymond 1998a) to reflect the manner in which the literally global community of OSS co-developers, operating in a decentralized cooperative manner according to the principle of prompt feedback, are able to solve the various problems that arise. Linux has had more than 1,000 developers working on the kernel alone, while Fetchmail has had more than 600 globally-distributed co-developers working asynchronously in different time-zones (Raymond 1998a).

The second aspect of the software crisis, software quality, is also addressed by the OSS approach, in that OSS developers are reckoned to be the most-talented and highly motivated 5% of software developers (Raymond 1998a). Also, peer review of any development product is truly independent, in that the global community of co-developers have no vested interest, consciously or subconsciously, in turning a blind eye to deficiencies in the product. Evidence of this quality and inherent reliability of OSS output is amply demonstrated by the fact that these products have achieved such a significant market share without any conventional marketing or advertising campaigns—there has not been any $100 million *Start me up* campaign for any OSS products!

Raymond (1999a) has considered the specific issue of when OSS development is appropriate in some detail and suggests the following (mainly technological criteria) as predisposing toward OSS development:

- When reliability and stability of the software are critical,
- When correctness is only established through independent peer review,
- When software is critical to the business,
- When the software establishes a communications infrastructure,
- When the key algorithms are part of common software engineering knowledge.

The final aspect of the software crisis, cost, is also addressed by OSS, and sets the stage for an understanding of the economic motivations of OSS development. Under the OSS model, software may be downloaded freely by *ftp*, or more typically, can be purchased in CD-ROM format for a very nominal fee, an anathema to traditional vendors of proprietary software. For example, Red Hat, one of the leading OSS product distributors, package 435 fully-tested OSS products into their distribution for a fee of about $50. Competitors can (and do) download free copies of the same products and offer them at a lower price. However, the rules of competition are not fully determined by price (which is nominal anyway); rather, companies compete on the basis of the service provided to the consumer, which can only be good news for the latter.

More important than sticker-price, OSS allows companies developing and implementing systems to share both the risks and long-term costs associated with a system. By shifting the locus of value from protecting "bits" of code to maximizing the gain from software use and platform development, OSS redefines software as an industry. Raymond (1999a) identifies the mistaken business and financial models underpinning conventional software development, terming it a service industry operating under the delusion of being a manufacturing one. Certainly, it is a well established fact that the vast proportion of the total cost of software development is incurred in the maintenance phase—with reliable estimates varying from 70% (Boehm 1976) to 80% (Flaatten et al. 1989). This suggests that the model for proprietary software, which operates a high purchase price with a low support fee, does not reflect the reality of the cost distribution in practice. This is recognized in the OSS model where the software is distributed for a nominal fee and companies then compete on service to the consumer. The model views software as a commodity product where the ingredients are free (Young 1999). Brand management becomes critical and customers learn to value a brand they can trust in terms of quality, reliability, and consistency. In these business conditions, OSS companies can learn from the experiences of companies such as Perrier, Ballygowan, and Heinz, where brand image has been successfully exploited in a highly-competitive consumer-driven market.

While technological and economic factors may be sufficient to understand industrial support for OSS, the motivations of individual developers are often socio-political. The nature of the software development craft and those who practice it needs to be considered. These issues will be discussed in detail in the analysis of the *who* category of the framework below. OSS developers tend to be self-selected and highly-motivated professionals. They may be working on typically long, drawn-out development projects in their own organizations; thus, any of their software output is not usually subject to the prompt feedback of positive reinforcement. When they contribute to OSS development, however, they get a very real "rush" from seeing their code being used and tested immediately (DiBona et al. 1999). Also, the OSS community norms ensure a strict meritocracy where quality speaks for itself. Contributors cannot confer expert status on themselves; rather, it arises through peer recognition. Although now viewed somewhat controversially (Wahba and Bridwell 1976), Maslow's hierarchy of needs, which posits a category of self-actualisation needs (Maslow 1970), has been drawn upon to help explain the motivation behind the committed contribution of OSS developers (Raymond 1998b). Furthermore, the possibility of learning and skill advancement and overt social and political agenda serve as powerful motivators.

A number of additional criteria that seem to be relevant in choosing OSS development can be derived. Project size would appear to be relevant, in that the benefits of OSS may perhaps be best realized on large projects where management overhead is a significant issue. Also, it appears that OSS works best when software should follow an evolutionary development pattern, never reaching completion but continually evolving as market needs change. Interestingly, researchers have argued that such a development pattern is the most appropriate given the emergent nature of requirements in the current organizational environment (Truex et al. 1998).

4.3 Analysis of the When and Where of OSS

The developmental environment of OSS is one that is shaped by "Internet time" and virtual geography. As noted above, OSS projects are often characterized by rapid development. Perhaps more importantly, the platform-building ethos of OSS allows for rapid evolution of software, addressing what Young and Rohm (2000) identify as a sharp disparity between hardware and software innovation, in that hardware advances run at a rate of more than twice that of software. OSS, as its history to date has shown, offers some promise that this can be redressed. Again, the huge expansion in the pool of potential co-developers has parallels with what took place in the automobile and telephone industries in the past. Fogel (1999) also discusses the parallels between OSS and biological evolution and identifies features such as natural selection and survival of the fittest from actual examples of OSS projects. The *modus operandi* of frequent, incremental releases encourages adaptation and mutation, and the asynchronous collaboration of developers means that OSS projects achieve an agility of which corporations often only dream. Geographically OSS is characterized by massive distribution, with teams, community, and peer groups defined by virtual, rather than physical, boundaries.

4.4 Analysis of the How of OSS

Briefly stated, the "how" of OSS is massive parallel development. The classic model seems to be that described by Raymond (1998a) in contrasting the cathedral and bazaar modes of development. Raymond's description of the Fetchmail project would appear to be typical of early (and many current) OSS projects. A single lead developer (or a small core of developers as in the Apache case) takes the lead in establishing the project direction. Co-developers, having full access to the source code, submit code patches to solve various problems that arise and to add to the functionality of the software, "scratching a developer's personal itch" as Raymond (1998a) so aptly puts it.

Within this framework, there are no formally documented norms and customs for OSS development; rather, there are implied customs and taboos which are learned by experience. For example, given the global distribution of developers, who do not tend to meet face-to-face, and the absence of any formal project plans, it is vital that all developers contribute to the same purpose. There is a strong taboo against forking, a phenomenon whereby the software evolves in different strands until the different strands eventually become incompatible. The taboo against forking is understandable when the stakes for OSS development are so strongly intertwined with reputation, as it is not practical for developers to be involved in both forks; thus, there is a reputation risk.

Increasingly, companies are acting as patrons, coordinators, and mediators for more centralized OSS projects. Developer "auction" sites, like SourceXchange (http://www.sourcexchange.com/) and CoSource (http://www.cosource.com/) unite clients with "itches to scratch" with developers. Competing distributors like Red Hat and Caldera each hire top-talent to build improvements to Linux, which are then freely available to the other one, in the belief that it is the market penetration of Linux, not their own distribution, that will best serve their long-term goals.

The OSS method of collaboration, whether "in the wild" or as part of an emerging business form, is supported by the infrastructural tools that it helped to create. Given that face-to-face meetings are not the norm for OSS co-developers, the Internet, and concurrent versioning software, form a critical enabling tool kit, allowing the submission and prompt testing of code patches, and facilitating the mode of parallel development in general, which makes for the frequent releases that characterize Open Source Software. Also, there is a low tolerance of "noise" as the communication channel is limited; thus, only high quality contributions tend to be surfaced.

4.5 Analysis of the Who of OSS

The *who* category also lends itself to three levels of analysis, namely that of the developers, companies, and users. In considering the individual OSS developer it is useful to draw a distinction between "hackers" and "crackers." Crackers are software

developers who specialize in compromising computer security, striving to sabotage commercial ventures, for example, by releasing alternative pirated offerings of new commercial products on their launch day, termed "zero day warez" in cracker parlance. A typical example of a cracker group is the *Cult of the Dead Cow*, the group responsible for *Back Orifice 2000*, a set of utilities for breaking Microsoft security tools. Despite the fact that a primary motivation for crackers is also peer reputation, there are important differences between crackers and the hackers who comprise the OSS development community. Crackers tend to hoard secrets and distribute binary code rather than source code. Also, while some early hackers were also driven by zealous anti-commercialism, the OSS community intends to exploit commercial opportunities. Interestingly, there is some evidence that the cracker community is moving toward OSS (Raymond 1998b).

Young (1999) suggests that it is a common but mistaken perception that OSS developers are lone 14-year-old amateurs operating from their bedrooms, and suggests that this perception may be encouraged by traditional closed proprietary software vendors as part of a FUD[5] strategy to discredit the OSS model. However, in reality OSS developers are typically professionals working in traditional software environments. As already mentioned, they are self-selected and highly motivated, and reckoned to be among the most talented 5% of developers (Raymond 1998b).

Given the reputation-based culture of OSS, it is somewhat paradoxical that the leading OSS developers tend to be modest and self-deprecating. The advantages of this trait in terms of stimulating initial development and ensuring the motivation to continue development were discussed earlier. Thus, it seems to be an important factor in the success of OSS.

At the company level, OSS distributors such as Red Hat and VA Linux have been the subject of much commercial investment. As noted above, these companies contribute a great deal of support to the OSS development community in a form of patronage akin to artists during the Italian Renaissance. They realize that even if the software is free, customers are willing to pay for convenience and a brand they can trust. For these OSS companies, market size is more important than market-share, and the rules of competition have to do with brand management and customer service. While distribution companies like Red Hat came on the scene after Linux, a number of "pure play" OSS companies have emerged, such as OpenSales (http://www.opensales.com), who make an open e-commerce engine and seek to generate profit from support services. Likewise, new business models, such as the auction services discussed earlier, are entering the market.

Traditionally, the users of OSS have been experts, early adopters, and nearly synonymous with the development pool. As OSS enters the commercial mainstream, a new emphasis is being placed on usability and interface design, with Caldera and Corel explicitly targeting the general desktop market with their Linux distributions. Non-expert users are unlikely to be attracted by the available source code and more likely to choose OSS products on the basis of cost, quality, brand, and support.

5. COMPARISON OF OSS AND RAPID APPLICATION DEVELOPMENT (RAD)

It is desirable that OSS be contextualized vis-à-vis other developmental methodologies. While a full contextualization is beyond the scope of this paper, a brief comparison to one other methodology may be useful. Based on the importance of the end user and the practice of incremental release, OSS is often compared to Rapid Application Development. Certainly, RAD is also often touted as an appropriate silver bullet to solve the software crisis. Table 6 provides a comparison of the central features of both approaches and illustrates where they are similar and different.

Table 6 reveals that there are certain similarities between OSS and RAD, but these are quite superficial and limited. Users are more heavily involved in both methodologies than in traditional development. Also, the frequency of product release is a point on which there are similarities. However, they differ radically in terms of application area, numbers of users, and their view of requirements. OSS products are typically complex back-office infrastructural support systems with many users while RAD is typically used in the development of standalone, single user, small projects (Fitzgerald 1999). Also, RAD is often used to help elaborate requirements that are not easily definable. However, the basic philosophy of RAD still assumes an end-point of a finalized system deliverable. In the case of OSS, the requirements are constantly being elaborated, and the product is never in a finalized state, until all development work has ceased on the product.

[5]FUD is an acronym for Fear, Uncertainty, Doubt, and it represents a "dirty tricks" strategy that has been used particularly in the computer industry to help undermine competitors (Irwin 1998).

Table 6. Comparison of OSS and RAD Approaches

Characteristic/Feature	OSS	RAD
Source code availability	Critical feature.	Not a critical feature.
Scope of system development	Large projects. Many developers. Large user base.	Small projects. Few developers. Small user base.
Nature of system deployment	Mission critical.	Interim or temporary solution. Not mission critical.
Type of system	Infra-structural, back-office support, multi-user.	Standalone, single or few users.
Development tools and emphasis	3GLs. Application area more algorithmically complex.	4GLs used. Emphasis on user interface, information presentation.
Release frequency	Very frequent—even more than once per day.	Quite frequent—new releases of software in 90-day time-boxes.
Role of users	Users are a critical feature, serving as coders, testers, documenters, and also providing prompt notification of new requirements.	Users are quite heavily involved doing requirement specification, testing, and documentation, but not actual coding.
View of system requirements	System is recognized as continually evolving incrementally—never finalized.	Prototypes produced to help with identification of requirements. Prototypes may become part of finalized system.

6. CONCLUSION

As noted above, we believe that rigorous academic inquiry into OSS is sorely needed. To that end, it is hoped that this paper will provide a foundational framework for further research by the wider academic community. There are three general areas which we believe offer the most promise in this regard.

6.1 Cross-Methodological Comparative Studies

If OSS is to become anything more than an interesting phenomenon observed from afar, efforts must be made to situate it within the context of other systems development methodologies. In particular, we believe four questions are critical:

- What life-cycle (or, indeed, life-cycles) underpins the OSS model?
- What insight can be derived from the dynamics of OSS's parallel, modular, collaborative development?
- What toolkit supports the OSS methodology? How does it differ from traditional development methodologies?
- What types of projects are and are not suited to OSS development?

6.2 Psychological and Sociological Inquiries

OSS offers an opportunity to significantly leverage previous work done in the areas of Computer-Supported Cooperative Work (CSCW), Computer-Mediated Communication (CMC) and the psychology of programming. While most reports on OSS stress the Utopian aspects of collective public good, spontaneous order, selfless behavior, absolute democracy, etc., the fact remains that the culture of OSS is highly individualistic and reputation-based, and that the opportunities for personal gain, both financial and otherwise, are enormous. There is a need to investigate deeply the dynamics of the OSS development process in practice so as to gain fresh insights into programming as a human activity. We consider two questions to be particularly important:

- How are OSS development communities created and maintained, and what implications does this have for other forms of virtual communities and work groups?

- How will the core values of the OSS community be significantly changed as the financial context of OSS becomes more lucrative, and what do businesses need to do to maintain the OSS community (e.g., all royalties earned by Young and Rohm's *Under the Radar* go to the Free Software Foundation)?

6.3 Investigation of Economic Models and Business Forms

While research has been conducted in relation to identifying the different business models underpinning OSS, more is required given the IPO history of OSS players, which has made this a multi-billion dollar industry. The future forms of OSS are open to speculation, and given that the Open Source term was specifically chosen to be more acceptable to commercial interests, it seems likely that "pure" OSS will give way to more hybrid business models. It also remains to be seen how OSS deals with the constant threat posed by the proprietary software industry. We believe the following issues are the most pressing:

- What are the implications of OSS for the nature of systems investment and ownership within organizations?

- Are OSS business models sustainable in the long-term? Or will, as some have suggested, OSS meet the fate of the Medieval "commons," which was over-farmed and became barren and sterile (Hardin 1993)?

- How will OSS businesses face the tremendous developmental and marketing difficulties associated with what Moore (1999) calls "crossing the chasm" from early adopters to the mainstream?

References[6]

Boehm, B. "Software Engineering," *IEEE Transactions on Computers* (25:12), 1976, pp. 1226-1241.

Brooks, F. *The Mythical Man-Month*, Reading, MA: Addison-Wesley, 1975.

Brooks, F. "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer Magazine*, April 1987, pp. 10-19.

Checkland, P. *Systems Thinking, Systems Practice*, Chichester, UK: Wiley, 1981.

Davenport, T., and Hansen, M. *Knowledge Management at Andersen Consulting*, Harvard Business School Case Series No. 9-499-032, Boston, 1998.

DiBona, C., Ockman, S., and Stone, M. "Introduction," *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, 1999.

Fitzgerald, B. "An Investigation of Rapid Application Development in Practice," in *Methodologies for Developing and Managing Emerging Technology Based Information Systems*, T. Wood-Harper, N. Jayaratna, and J. Wood (eds.), London: Springer-Verlag, 1999, pp. 77-87.

Flaatten, P., McCubbrey, D., O'Riordan, P., and Burgess, K. *Foundations of Business Systems*, Chicago: Dryden Press, 1989.

Fogel, K. *Open Source Development with CVS*, Scottsdale, AZ: Coriolis Open Press, 1999.

Halloween Documents "The Halloween Documents," 1998 (http://www.opensource.org/halloween/; accessed May 1, 2000).

Hardin, G. *Living Within Limits: Ecology, Economic and Population Taboos*, Oxford: Oxford University Press, 1993.

Irwin, R. "What is FUD?," 1998 (http://www.geocities.com/SiliconValley/Hills/9267/fuddef.html; accessed (May 1, 2000).

Maslow, A. H. *Motivation and Personality* (2nd ed.), New York: Harper and Row, 1970.

McKusick, M. "Twenty Years of Berkeley UNIX: From AT&T-Owned to Freely Redistributable," in *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, 1999.

Mitchell, R. "Linux Widens Lead in Web Market Share," *Wide Open News*, April 17, 2000a (http://wideopen.com/story/739.html; accessed May 1, 2000).

Mitchell, R. "New Numbers, Historic Ramifications," *Wide Open News,* February 10, 2000b (http://wideopen.com/story/499.html; accessed May 1, 2000).

[6]This paper and the following reference list contain URLs for World Wide Web pages. These links existed as of the date of submission but are not guaranteed to be working thereafter. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced. The author(s) of the Web pages, not ICIS, is (are) responsible for the accuracy of their content. The author(s) of this article, not ICIS, is (are) responsible for the accuracy of the URL and version information.

Moore, G. A. *Crossing the Chasm,* New York: Harper-Perennial, 1999.

O'Reilly, T. "Open Source: The Model for Collaboration in the Age of the Internet," *Wide Open News*, 2000 (http://www.wideopen.com/reprint/740.html; accessed May 1, 2000).

Perens, B. "The Open Source Definition," in *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, 1999.

Raymond, E. "The Cathedral and the Bazaar," 1998a (http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazar.html; accessed May 1, 2000).

Raymond, E. "Homesteading the Noosphere," 1998b (http://www.tuxedo.org/~esr/writings/homesteading/ homesteading.html; accessed May 1, 2000).

Raymond, E. "The Magic Cauldron," 1999a (http://www.tuxedo.org/~esr/writings/magic-cauldron/magic-cauldron.html; accessed May 1, 2000).

Raymond, E. "Shut Up and Show Them the Code," *Linux Today*, June 28, 1999b (http://linuxtoday.com/stories/7196.html; accessed May 1, 2000).

Sowa, J. and Zachman, J. "Extending and Formalizing the Framework for IS Architecture," *IBM Systems Journal* (31:3), 1992, pp. 590-616.

Speedie, A. "Apache Grabs Two-Thirds of New Websites," *Wide Open News,* March 22, 2000 (http://wideopen.com/ story/670.html; accessed May 1, 2000).

Stallman, R. "The GNU Operating Systems and the Free Software Movement," in *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, 1999.

Truex, D., Baskerville, R. and Klein, H. "Growing Systems in an Emergent Organization," *Communications of the ACM* (42:8), 1999, pp. 117-123.

Wahba, M. A., and Bridwell, L. G. "Maslow Reconsidered: A Review of Research on the Need Hierarchy Theory," *Organizational Behavior and Human Performance* (15:2), 1976, pp. 56-95.

Young, R. "Giving it Away: How Red Hat Stumbled Across a New Economic Model and Helped Improve an Industry," in *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, 1999.

Young, R. and Rohm, W. G. *Under the Radar,* Scottsdale, AZ: Coriolis Group, 2000.

Zachman, J. "A Framework for IS Architecture," *IBM Systems Journal* (26:3), 1987, pp. 276-292.