The Transformation of Open Source Software

Brian Fitzgerald
Lero – the Irish Software Engineering Research Centre,
University of Limerick, IRELAND

(**Forthcoming in *MIS Quarterly*, Vol. 30, No. 3, 2006)**

Abstract
A frequent characterization of open source software is the somewhat outdated, mythical one of a collective of supremely talented software hackers freely volunteering their services to produce uniformly high-quality software. I contend that the open source software phenomenon has metamorphosed into a more mainstream and commercially viable form, which I label as OSS 2.0. I illustrate this transformation using a framework of process and product factors, and discuss the shift in the application of the bazaar metaphor from the development process to the product delivery and support process. Overall the OSS 2.0 phenomenon is significantly different from its free software antecedent. Its emergence accentuates the fundamental alteration of the basic ground-rules in the software landscape, signifying the end of the proprietary-driven model that has prevailed for the past 20 years or so. Thus, a clear understanding of the characteristics of the emergent OSS 2.0 phenomenon is required to address key challenges for research and practice.

Keywords: Open source software, free software, IS development

Introduction

Just a few years ago, it would have seemed preposterous to suggest that the might of the proprietary software industry, as exemplified by Microsoft, could be threatened by the largely volunteer open source software movement. This movement, however, has altered the basic nature of the software industry. On the supply side, fundamental changes have occurred to the development process, reward mechanisms, the distribution of development work, and business models that govern how profit can be achieved. On the demand side, the alternatives traditionally available to organizations for software acquisition — buy or build — have been supplemented with another credible alternative — namely, open source. A range of issues arises also in relation

to the altered nature of software support, the need for new models for total cost of ownership (TCO) of software, and perceptions of exposure to the possibility of IP infringement.

Radical movements often mature to be accommodated into the mainstream. French Impressionist art in the 19th Century is a good example. I contend that the open source phenomenon has undergone a significant transformation from its free software origins to a more mainstream, commercially viable form—OSS 2.0, as I term it[1]. This accommodation with the mainstream ensures that the emergent OSS 2.0 phenomenon will continue to thrive as a significant force in the future software landscape. Indeed, it is a harbinger of an end to the current dominance of a proprietary, closed source software model. I illustrate how the quintessential proprietary software company, Microsoft, can appear to satisfy the definition of an open source company, while a quintessential open source company, Red Hat, can appear to resemble a proprietary software company. I identify how OSS 2.0 can accommodate these apparent transformations through achieving a balance between a commercial profit value-for-money proposition while still adhering to acceptable open source community values.

Compounding the fact that the open source phenomenon represents a radical change in the software landscape, it is often mistakenly and paradoxically characterized as a collective of supremely talented developers who volunteer their services to develop very high-quality software by means of a revolutionary new approach. This characterization is a myth as almost every aspect of it can be questioned (Fitzgerald 2005; Michlmayr et al. 2005; Rusovan et al. 2005; Schach et al. 2002). One effect of this outdated characterization is that research to date has focused i*nward* on the phenomenon, studying the motivations of individual developers to contribute to OSS projects, or investigating the characteristics of specific OSS products and projects. Such research has been facilitated by the availability of a vast amount of data on mailing lists and portals such

[1] Because this article focuses on the evolution and transformation of the open source phenomenon, terminology is an issue. I use the term FOSS to refer to the initial era of Free and Open Source Software. I use OSS 2.0 to refer to the phenomenon which I see emerging now, and I use 'open source software' to refer to the phenomenon in general.

as Sourceforge. In the case of the latter, however, it is important to bear in mind that only a small percentage of the 100,000 or so projects are stable and mature.

While some disagreement exists between the free and open source software community as to the definitions of free software versus open source software (www.fsf.org/philosophy/free-software-for-freedom.html), I will not dwell on that here. I first propose a framework to characterize the initial free and open source software (FOSS) phenomenon. While the shift to OSS 2.0 may seem incremental, I use this framework to illustrate the deep nature of the transformation. I also identify key challenges for research and practice that arise as a result of the emergence of OSS 2.0.

Characterizing FOSS

Tushman and Andersen (1986) propose a framework for technological transformation based on two sets of technological factors – namely, process and product. I propose a similar framework to characterize the initial FOSS phenomenon (Table 1 - which also presents a characterization of OSS 2.0, discussed in the next section).

FOSS Development Process

In conventional software development, the development lifecycle in its most generic form comprises four broad phases: planning, analysis, design, and implementation. In FOSS development, these stages tended to be configured differently. The first three phases of planning, analysis and design are concatenated and performed typically by a single developer or small core group. The planning phase is probably best summarized by Raymond's (1999) phrase of a single developer perceiving " an itch worth scratching." This leads to construction of an initial prototype. Given the ideal that a large number of globally distributed developers of different levels of ability and domain expertise should be able to contribute subsequently, the requirements analysis phase was largely superseded. Requirements were taken as generally understood and not needing interaction among developers and end-users. In this regard, FOSS developers were

invariably users of the software being developed. This model is perhaps best suited to infrastructure software in horizontal domains. Design decisions also tended to be made in advance before the larger pool of developers starts to contribute. Systems are highly modularized to allow distribution of work and reduce the learning curve for new developers to participate (they can focus on particular subsystems without needing to consider the system in its totality).

In the FOSS development lifecycle, the implementation phase consists of several sub-phases (Feller and Fitzgerald 2002):

- Code – writing code and submitting to the FOSS community for review
- Review – a strength of FOSS is the independent, prompt peer review
- Pre-commit test – the negative implications of breaking the build ensure that contributions are tested carefully before being committed
- Development release – code contributions may be included in the development release within a short time of having been submitted – this rapid implementation being a significant motivator for developers
- Parallel debugging – the so-called Linus's Law ('given enough eyeballs every bug is shallow') as the large number of potential debuggers on different platforms and system configuration ensures bugs are found and fixed quickly
- Production release – a relatively-stable debugged production version of the system is released

The management of this process varies a great deal. Different projects have varying degrees of formalism as to how decisions are made, but the principle of "having a tail-light to follow" (Bezroukov 1999) captures the spirit well. Often, the initial project founder or small core group make the key decisions in accordance with the process outlined in the lifecycle above.

Table 1 Characterizing FOSS and OSS 2.0

Process	FOSS	OSS 2.0
Development Lifecycle	• Planning – "an itch worth scratching" • Analysis – part of conventional agreed-upon knowledge in software development • Design – firmly based on principles of modularity to accomplish separation of concerns • Implementation o Code o Review o Pre-commit test o Development release o Parallel Debugging o Production Release (often the planning, analysis, and design phases are done by one person/core group who serve as '*a tail-light to follow*' in the bazaar)	• Planning – purposive strategies by major players trying to gain competitive advantage • Analysis and design – more complex in spread to vertical domains where business requirements not universally understood • Implementation sub-phases as with FOSS, but the overall development process becomes *less* bazaar-like • Increasingly, developers being paid to work on open source
Product	**FOSS**	**OSS 2.0**
Product Domains	• Horizontal infrastructure (operating systems, utilities, compilers, DBMS, web and print servers)	• More visible IS applications in vertical domains
Primary Business Strategies	• Value-added service-enabling • Loss-leader/market-creating	• Value-added service enabling o Bootstrapping • Market-creating o Loss-leader o Dual product/licensing o Cost reduction o Accessorizing • Leveraging community development • Leveraging the open source brand
Product Support	• Fairly haphazard – much reliance on email lists/bulletin boards, or on support provided by specialized software firms	• Customers willing to pay for a professional 'whole-product' approach
Licensing	• GPL, LGPL, Artistic License, BSD, and emergence of commercially-oriented MPL • 'Viral' term used in relation to licenses	• Plethora of licenses (85 to date validated by OSI or FSF) • 'Reciprocal' term used in relation to licenses

FOSS Product Domains

Due to the globally distributed nature of the development community – most members never meet face-to-face – FOSS products have tended to be infrastructural systems in horizontal domains. Their requirements are part of the general taken-for-granted wisdom of the software development community. Thus, the most successful FOSS products – the Linux operating system, the Apache web server, the Mozilla browser, the GNU C compiler, the Perl scripting language, and MySQL database management system – are all examples of horizontal infrastructure software.

Primary FOSS Business Strategies

Several FOSS business strategies have been proposed (Hecker 2000; Raymond 1999). Two have been most significant – namely, value-added service-enabling and loss-leader/market-creating.

An early example of the value-added service-enabling model was Cygnus Solutions, which integrated a suite of GNU tools and sold support services and other complementary software products. Red Hat is probably the most well-known proponent of this strategy. Effectively, Red Hat simplifies the task facing end-users in deploying an overall open source solution, such as Linux, that requires complex configuration of different components.

In the loss-leader/market-creating model, the open source product is distributed for free, but with the end goal of enlarging the market for alternative, closed source products and services. For example, the open source Sendmail product enlarges the subsequent market for Sendmail Pro, a product with extra functionality that is distributed for a fee.

FOSS Product Support

The nature of product support in FOSS has been haphazard and bazaar-like and is different from the proprietary model. Requests for support and solutions are commonly sent to forums such as bulletin boards and mailing lists. In some cases, support may be purchased from a competent

third-party provider. For example, Linux support is available from HP or IBM, or a specialized (often local) software firm may offer support and consultancy services. While many organizations are reluctant to rely on bulletin boards for support, they may be equally reluctant to purchase consultancy support to deploy a solution effectively (Fitzgerald and Kenny 2003).

FOSS Licensing

Ironically, given the perceptions that FOSS is collectivist and anti-intellectual property, the success of the open source model is due largely to the use of licensing, albeit in a form that counters the normal restrictive sense. Property rights are vested in the author through copyright, with liberal rights granted to others under license. In the FOSS era, the principal licenses have been the GNU Public License (GPL), the Lesser GPL (LGPL), the Artistic License, and the Berkeley System Distribution (BSD). This era also saw the emergence of the commercially oriented Mozilla Public License (MPL), which has been quite influential.

The earliest open source license, the GPL, was created in the mid 1980s to distribute the GNU project software. Most open source software to date has been distributed under the GPL, Linux being one high-profile example. The GPL subverts the traditional concept of restricted access through copyright by ensuring complete, unrestricted access to all open source software and any derivatives. These must also be licensed under the same terms, referred to as 'copyleft – all rights reversed'. This latter guarantee of the same rights to subsequent users caused such licenses to be termed 'viral.'

The GPL is controversial, because it requires that all applications that contain GPL software are also released under a GPL license. A modified version, the Lesser GPL (LGPL) was created when this proved impractical. The LGPL differs from the GPL in two main ways. First, it is intended for use with software libraries (it was initially known as the Library GPL). Second, the software may be linked with proprietary code, which is precluded by the GPL.

Another early license that achieved fairly widespread use is the BSD license, which imposes few restrictions. Its main requirement is the retention and acknowledgment of previous contributors' work.

The FOSS era also saw the creation of the commercially oriented Mozilla Public License (MPL) by Netscape. The MPL was significant because it focused on the conversion of a commercial software product to open source. This process raised significant challenges. It rendered the GPL problematic, because each licensor whose software was incorporated into the Netscape browser would have had to use the same open source license. Netscape was also concerned that an academic-style license would not guarantee that developers would contribute back to the community. It created a new license, the MPL, to address these specific concerns.

Characterizing OSS 2.0

The term 'open source' was coined in 1998 to place the phenomenon on a more business-friendly footing than that associated with the ambiguous 'free software.' The latter led to the common misperception was that individuals or organizations could not make money with free software. The open source initiative succeeded spectacularly well, and the emergent OSS 2.0 has a very strong commercial orientation. Table 1 above summarizes how OSS2.0 differs from its FOSS antecedent.

OSS 2.0 Development Process

The largely voluntary nature of FOSS led to a vacuum in relation to strategic planning (competing with Microsoft on the desktop being one example of a questionable strategy). In the OSS 2.0 development lifecycle, in contrast, strategic planning moves to the fore. The haphazard principle of individual developers perceiving 'an itch worth scratching' is superseded by corporate firms considering how best to gain competitive advantage from open source. For example, Red Hat has published an architecture roadmap that details its plans to move open source up the software stack towards middleware and management tools. Other proprietary

companies have also seen the strategic potential of open source to alter the competitive forces at play in their industry, perhaps to grow market share or undermine competition. For example, IBM is a strong supporter of Linux, because it erodes the profitability of the operating system market and adversely affects competitors like Sun and Microsoft.

Analysis and Design

As already discussed, FOSS products were targeted primarily at horizontal infrastructure where requirements and design issues were largely part of the established wisdom, thus facilitating a global developer base. Most business software, however, exists in vertical domains where effective requirements analysis poses real problems. Students and developers without any experience in the application area lack the necessary knowledge to derive the accurate requirements that are a precursor to successful development. In OSS 2.0, therefore, the analysis and design phases have become more deliberate. In many cases, based on the earlier phase of strategic planning, paid developers will be assigned to work on open source products in vertical domains.

Given the increasingly commercial nature of OSS 2.0, more rigorous project management is required to achieve a professional product. As a consequence, a shift is occurring whereby the management of the development process is becoming *less* bazaar-like. This outcome is already evident in the formalized meetings for a number of popular open source products (for example, the Apache conferences in the US and Europe, the regular Zope/Plone development project meetings, and the GNOME annual project conferences) (German 2003). These meetings bring together developers to coordinate and plan further development. The legal incorporation of several open source projects ostensibly reduces the risk of litigation for individual developers (O'Mahony 2005), while allowing these projects to accept donations, perhaps to implement requested functionality.

OSS 2.0 Product Domains

Interestingly, in the highly competitive software world, several open source products have nudged out proprietary alternatives to emerge as 'category killers'– that is, products of sufficiently high quality and popularity that they obviate the need for development of competitive products. Also, OSS 2.0 is moving from deployment as back-office, invisible infrastructure to front-office, highly visible deployment of IS applications in vertical domains. An example is the Beaumont Hospital case study (Fitzgerald and Kenny 2003) where a number of in-house developed applications are being made available on an open source basis to other healthcare agencies. In the context of open source, this development is significant. To date, it has often been assumed that open source products will not affect many vertical domains, because developers will not perceive an 'itch worth scratching' there. If, however, organizations in these areas subscribe to the open source philosophy and contribute specialist expertise to open source projects, the model will spread to more vertical applications.

OSS 2.0 Business Strategies

The FOSS era had two overarching 'families' of revenue models – value-added service-enabling and loss-leader market-creating. These models are still applicable in OSS 2.0, but they are further nuanced. Other strategies have also emerged, including leveraging community software development and leveraging the open source brand. Moreover, companies may not stick solely to one of these models and may employ pragmatic hybrids instead.

Value-Added Service-Enabling in OSS 2.0

Building a lucrative service and support business on top of open source was discussed earlier. Companies like Red Hat and Novell have realized large revenues through annual subscriptions. This 'bootstrapping model' is taken to a higher level in OSS 2.0, where open source products are treated as a platform – somewhat similar to a highway or a telecommunications infrastructure. A company bootstraps its own value-added specialty on top of this infrastructure. Small software

companies can become part of an ecosystem offering consultancy, service and support of open source products. One example would be to purchase a single support license from MySQL and sell local support to a number of customers. Roughly 90% of customer support requests are probably easily dealt with directly, and the 10% of complicated issues could be passed to MySQL for resolution, and then the solution passed back to the local customers.

High-profile organizations like Amazon, Google and Salesforce.com take advantage of the reliability and low cost of open source to create a platform on which they can offer value-added services in their own business domains. For the most part, the use of open source is invisible to their customers. These companies also customize open source products to suit their internal needs. Moreover, because they are not redistributing software, they are not faced with any problems of non-compliance with the GPL.

Market Creation Strategies in OSS 2.0

The other FOSS era business strategy discussed above is the loss-leader market-creating strategy. In OSS 2.0, the emphasis is firmly focused on market creation through a loss-leader approach and involves products with dual licensing, cost reduction, and accessorizing.

Integrated Development Environments (IDEs) have illustrated this approach. Traditionally, IDEs were expensive proprietary applications, which were especially lucrative if they attracted a large license-paying user base. When IBM chose to move its Eclipse IDE to open source, the decision seemed surprising because the source code was valued at $40 million. IBM has had massive compensations, however. It substantially increased its popularity as a development platform and expanded the market for its complementary products. Several other companies have now also moved their proprietary IDEs to open source, including Sun with NetBeans and BEA with Beehive.

Several examples of dual product/licensing exist. MySQL provides a high-profile example of such a strategy. Millions of free copies of MySQL have been downloaded. Of these, about one customer in every thousand has purchased a commercial license from MySQL. This proportion seems small, but it amounts to thousands of fee-paying customers. Other dual-product strategies include Red Hat with Fedora and Enterprise Linux, Sun with StarOffice and OpenOffice, and Iona Technology with Celtix and Artix.

Companies can also leverage the commodification effect that has occurred with open source. They take advantage of open source in terms of its low cost, reliability, and portability across platforms. For example, Oracle can reduce the overall cost of database implementation for its customers, and IBM can reduce the overall cost of servers. In the area of embedded systems, open source is fast becoming dominant. Here, companies are concerned with open standards, stability, high performance, small footprint, and the ability to run on generic hardware. A vibrant, responsive development community exists and is willing to port to other platforms and write extra utilities.

Several companies leverage open source as a base upon which they offer products other than software. For example, HP promotes open source in areas that facilitate the deployment of its hardware, while the O'Reilly publishing house has earned significant revenue from books related to the open source concept.

Leveraging Community Software Development

Leveraging the talents of the open source community allows companies to increase development productivity, with the added benefit that much work may be done for free. Thus, hundreds of Eclipse plug-ins have been developed. Also, Apple's initiative in starting the Darwin open source project to develop part of its operating system facilitates extra development contributions that for the most part are free. In addition, Apple's reputation in the open source community has

improved. The phenomenon becomes circular, as the extra functionality increases the software's attractiveness to other developers. These, in turn, contribute additional functionality.

Leveraging the Open Source Brand

While patents and copyrights are key issues with respect to free software, another IP mechanism, the trademark or brand, could become significant with OSS 2.0. For example, Oracle promotes the "unbreakable Linux" slogan. Also, an increasing number of government agencies and public administrations (traditionally the largest consumers of software) are mandating that open source be a priority option, even to the extent of requiring formal justification for not choosing an open source solution if one is available. This will ensure the 'open source' brand becomes even more important in the future.

OSS 2.0 Product Support

In the past, developers have referred to the "exhilarating succession of problem-solving challenges" in installing open source products (Sanders 1998). As the OSS 2.0 model becomes more mainstream, however, time-impoverished professionals are unlikely to seek exhilaration in this manner. Further, many organizations have difficulty relying on bulletin boards for their support. As OSS 2.0 evolves, customers will want a professional service – support, training, and certification – and will be prepared to pay for it.

The 'Whole-Product' Approach – From Bazaar Process to Bazaar Product

The particular characteristics of OSS 2.0 position it as a good exemplar of the 'whole-product' concept of a market-driven business approach that seeks to deliver a complete solution to the customer in terms of products and services (Moore 1999). The open source phenomenon is market-driven and, as discussed above, places a great deal of emphasis on services. It adopts a professional approach to achieving value by establishing a profitable business venture for which customers are willing to pay the going rate. In this scenario, developers do the coding. Others complete the business model by adding sales and marketing services – necessary activities but

ones in which developers may not be interested. The OSS 2.0 'whole-product' approach is also larger than a single company or software product or service. Indeed, the network benefits of open source arise as a result of the size of the overall community and ecosystem. Thus, a network of interested parties with complementary capabilities can form an ecosystem to offer a professional product and service in an agile, bazaar-friendly manner. Customer service requests can be routed to the most appropriate expert partner in the network, perhaps even to the developer who wrote the actual code. In this manner, the OSS 2.0 brand increases trustworthiness to achieve market-leader status. Such convenience networks exist already in conventional business circles. The LVMH (Louis Vuitton Moet Hennessy) brand is an international network of almost 50 luxury brand leaders in fashion, wines and spirits, watches, jewelry, and cosmetics (www.lvmh.com). From a business perspective, this network of well-known brands creates the ultimate luxury brand status, LVMH. Nonetheless, individual businesses can still pursue their own interests independently.

In OSS 2.0, the bazaar metaphor therefore shifts from just being associated with the development process (Raymond, 1999), which becomes *less* bazaar-like, to product delivery and support, which becomes *more* bazaar-like. Many companies will find profitable opportunities in customer support. The claim by large proprietary software companies that open source would stifle local software industries is proving unfounded. A more-likely scenario is that small service-centric software companies will thrive by providing training, technical support, and consultancy for local organizations that deploy open source products.

OSS 2.0 Licensing

In OSS 2.0, a plethora of license types has emerged. The Open Source Initiative (OSI) or the Free Software Foundation (FSF) has approved almost 100 distinct licenses overall to date between them (but with little general agreement as only about one-third of these licenses are approved by both) (Lyddy-Collins 2005). The licenses can be grouped into four broad categories: reciprocal

licenses (as per the FOSS era), academic-style licenses, corporate licenses, and non-approved (by FSF or OSI) licenses such as Microsoft's Shared Source family of licenses (Table 2).

Table 2 A Typology of OSS 2.0 Licenses

Reciprocal	GPL, LGPL, Open Source License (OSL)
Academic Style	Academic Free License, Apache License, BSD, MIT
Corporate Type	MPL, Qt Public License, Sun Public License, IBM Public License, Apple Public License, Eclipse Public License
Non-Approved (e.g., Shared Source family)	Microsoft Shared Source Initiative Licenses: (Microsoft Community License, Microsoft Permissive License), Sun Community Source License (SCSL)

In OSS 2.0, the term 'viral' has been adjudged to have negative connotations. The preferred term is 'reciprocal.' Reciprocal licenses such as the GPL and LGPL have already been discussed above in relation to FOSS licensing. Another variation, the Open Software License (OSL), was created in 2002 as an alternative to the GPL that would be more acceptable to corporate users and developers. Again, this emphasizes the continued progression towards corporate and commercial compatibility, which is at the heart of OSS 2.0. Interestingly, while the FSF has issued warnings against this license, Linus Torvalds has adopted it for open source development other than the Linux kernel.

Corporate-style licenses are central to OSS 2.0. They reveal a potential friction point in OSS 2.0, because they seek to benefit corporate interests rather than the open source development community. As mentioned already, these are generally based on the Mozilla Public License (MPL). Typically, they seek to allow open source code to be mixed with proprietary code and to ensure that corporate sponsors retain control of derivative works.

The non-approved category of licenses for OSS 2.0 is perhaps the most interesting. It pushes the boundaries of proprietary software as this sector seeks to accommodate the open source model.

Two significant exemplars are the Sun Community Source License (SCSL) and the Microsoft Shared Source Initiative family of licenses.

The realization that "transparency increases trust" (Matusow 2005) has led to Microsoft's Shared Source Initiative. Microsoft has recently converged on three core shared source licenses:

- *Microsoft Reference License* allows licensees to merely view source code. This practice is regarded with deep suspicion by the open source community, which foresees potential transgressions of patents by developers who copy the code.
- *Microsoft Community License* is based on the Mozilla Public License and is intended for collaborative projects.
- *Microsoft Permissive License* is similar to the BSD license and effectively allows "licensees to review, modify, redistribute, and sell works with no royalties paid to Microsoft" (Matusow 2005).

Both the FSF and OSI originally agreed that the Shared Source initiative was neither free nor open. Nonetheless, it will be increasingly difficult to exclude licenses, such as the Microsoft Permissive License, that comply with the hybrid model of OSS 2.0.

Microsoft is therefore likely to be a major player in OSS 2.0. It has already distributed an open source product for some time – Windows Services for Unix – and has publicly acknowledged that Windows 2000 and Windows XP use open source BSD code. Also, it has a number of high-profile open source projects on SourceForge. Microsoft has abstracted some of the key ideas from open source. Recognizing the power of the social and community identification aspects of open source, it has introduced the Most Valued Professionals (MVP) initiative. It has extended access to source code to this select group. The Open Value policy permits sales representatives to offer extreme discounts and zero percent financing to small businesses that might switch to zero-cost open source (Roy 2003).

Summary

The above discussion illustrates how the OSS 2.0 phenomenon is significantly different from its FOSS antecedent. The development process becomes less bazaar-like as strategic planning becomes paramount. Analysis and design are more deliberate as the model spreads to vertical product domains. Developers are increasingly being paid to work on open source. More sophisticated business models are emerging and employed in a hybrid fashion. Customers are willing to pay the going rate for the 'whole product' in terms of support, which in turn can be delivered by a bazaar network of interested parties that provide varied but complementary services. Licensing also moves to the fore as proprietary companies produce licenses that comply with the open source definition, while open source companies seek to raise money through licensing. Given this complex melting pot, a number of significant implications and challenges for research and practice emerge.

Implications and Challenges for Research and Practice

The discussion above indicates several issues that provide key challenges for research and practice (Table 3). As is appropriate in an applied discipline, these are not completely distinct. Challenges for practice have a research angle and vice versa. Some research initiatives that appear relevant to addressing these challenges are also identified.

Implications and Challenges for Research

Transferring lessons from open source to conventional development

While open source may not represent a real paradigm shift in software development (Fitzgerald 2005), the model is an extremely successful exemplar of globally distributed development. It is attracting considerable attention in the current climate of outsourcing and off-shoring. Organizations are seeking to emulate open source success on traditional development projects, through initiatives variously labeled as 'inner source,' 'corporate source,' or 'community source' (Dinkelacker and Garg 2001; Gurbani et al. 2005).

Table 3 Key Issues for Research and Practice

Research
Transferring lessons from open source development to conventional development (inner source)
Offshoring – globally-distributed software development
Open code-sharing, large-scale peer-review, community development model
Expanded role of users and altered user-developer relationship
Elaboration of business models
Derivation of appropriate TCO models
Practice
Achieving balance between 'value-for-money' versus acceptable community values
Implementing the whole-product approach
Stimulating development in vertical domains
Safeguarding against IPR infringement

Other open source principles – such as open sharing of source code, large-scale independent peer review, the community development model, and the expanded role of users – also have important implications. In the traditional model of software development, users and developers are often located in separate departments. They sometimes have little mutual respect or voluntary interaction. The user-developer relationship in open source has typically been different. Early open source developers were often users of the products. As OSS 2.0 has emerged, this situation has changed. In the absence of a traditional vendor, users need to become involved more intimately in the development process, as technical staff cannot simply send a checklist of requirements to the vendor to ascertain if needs will be met. Deploying open source can lead to a sense of shared adventure that is not common in the proprietary software arena. Furthermore,

users may be more willing to sacrifice certain desired functionality if open source products could not easily provide it (Norris 2004).

Elaboration of business models

Much research is already being undertaken to refine and elaborate the business strategies discussed earlier (e.g., Koenig 2004; Krishnamurthy 2005; O'Neill 2005; Onetti and Capobianco 2005). Nonetheless, a more-careful definition of the concept is needed. For example, the term 'business model' is frequently used loosely in the context of open source. A useful definition of business model suggests it comprises three components: value, revenue, and logistics (Mahadevan 2000). The value component represents the value proposition for customers and vendors, the revenue component focuses on how organizations can earn revenue, and the logistics component focuses on supply chain issues. Revenue generation has been the primary focus for most of the research on open source business models. As the earlier discussion of OSS 2.0 business strategies illustrates, however, the value proposition and the logistics of the whole product across the overall supply chain are paramount in OSS 2.0. More analysis is needed in these areas.

Deriving appropriate Total Cost of Ownership (TCO) measures for OSS 2.0

Calculating the total cost of ownership (TCO) of software is a complex, multi-faceted issue. It requires consideration of many factors, including software purchase, maintenance and upgrade costs, hardware purchase and maintenance costs, personnel training, and legal and administrative costs (Russo et al. 2005). Given this complexity, proprietary and open source advocates predictably have each claimed a lower TCO (Wheeler 2005). Nonetheless, conventional TCO measures may not be suited to the open source phenomenon. Less-obvious benefits accrue due to network externality effects and a more cooperative developer-user relationship.

A promising strand of research that could suit the dynamics of open source is based on the theory of real options investment analysis (Fichman 2004). Real options analysis is appropriate where

high levels of flexibility and uncertainty exist (characteristics of open source environments). In terms of flexibility, considerable scope surrounds which products or functions may be implemented and how the software might be customized. Also, the zero-cost aspect offers considerable flexibility in terms of choosing when implementation occurs. Uncertainty arises because no 'royal road' to fail-safe open source implementation exists.

Implications and Challenges for Practice

Value for Money v. Adhering to Acceptable Community Values

The ambiguous term 'free' may have been the key word for FOSS, where both its meanings (i.e., free as in 'zero cost,' and free as in 'unrestricted access') were significant. 'Value,' an even more ambiguous term, will be the key word for OSS 2.0. Two of the term's connotations are especially significant – 'value for money' and 'acceptable community values.' The integration of open source into the commercial arena and the associated desire to create profit represents a critical source of tension, given the concomitant need to achieve a balance with collectivist, public-good community values – an inevitable legacy from the more ideologically-driven Free Software community. Both connotations of value are discussed here.

Value for Money

OSS 2.0 can dramatically alter the economic dynamics of a marketplace. Despite the vast sums of money involved and the enormous economic potential of OSS 2.0, it erodes certain hitherto profitable markets (e.g., the multi-billion dollar operating system market). As OSS 2.0 emerges, those involved are neither driven primarily by ideology nor seeking to make vast fortunes. They simply wish to earn a reasonable livelihood from their efforts (Everitt 2004). Both customers and developers need to perceive value for money in OSS 2.0. Free as in zero cost is replaced by a value-for-money concern, and OSS 2.0 customers are prepared to pay for a professional service. For instance, many companies are prepared to pay a fee for StarOffice with associated support and warranty, in preference to adopting the zero-cost, OpenOffice alternative.

Acceptable Community Values

OSS 2.0 blurs the distinction between open source and proprietary software. Key open source players such as Red Hat and Novell's SUSE Linux business unit position their Linux distributions to be more similar to a proprietary model. Traditional proprietary companies, such as HP, IBM and Microsoft, move more towards open source. Nevertheless, in the OSS 2.0 model, these companies must still satisfy certain criteria in relation to acceptable community values (a significant challenge for OSS 2.0). Large commercial organizations are not always well perceived within the open source community. Companies such as IBM, Sun, and HP support open source initiatives, but their support for patents is clearly at odds with the open source philosophy. Also, the quintessential patron of open source, Red Hat, could struggle in future as its policies increasingly conflict with community spirit and values. Use of subscription agreements and effective customer lock-ins through confidential service bulletins are close to the boundary of acceptable community values. Also, MySQL's decision to port to SCO's OpenServer platform, although a profitable venture, has met with strong criticism because of the negative feelings towards the SCO group within the open source community. The power of community should not be underestimated. A telling example was the attempt by Caldera to sell its Linux distribution, which failed due to the extremely negative reaction of the open source community.

Within the overall community, the spirit of OSS 2.0 can lead to positive network externalities. In the Beaumont Hospital case (Fitzgerald and Kenny 2003), users of the same open source products in Finland traveled to Ireland to volunteer support and offer extra functionality that they had developed. The expectation was that Beaumont would reciprocate by making available any extra functionality they developed. Cooperation of this nature is rare in the proprietary marketplace, but it is symptomatic of the strong community value orientation of open source.

Implementing the whole-product approach

I have already discussed how a bazaar network of companies can collaborate to offer a whole-product approach to customers. In addition to providing a customized professional support service, the whole-product bazaar network can satisfy other emerging business needs. For instance, the plethora of open source products currently available and the lack of vendors to provide marketing information cause a large knowledge gap. An up-to-date catalog of high-quality open source products is needed which could provide details on the functionality offered by various products, the types of support available, training needs, reference sites of deployment, and companies offering support.

Stimulating open source in vertical domains

Early open source products tended to involve horizontal infrastructure where requirements are part of conventional wisdom. Developers with different backgrounds, or even students, could contribute. In vertical domains, however, business requirements are more complex and demand more-specialized knowledge. A significant challenge is to stimulate open source development in these domains. An example in the healthcare sector is the Beaumont Hospital case mentioned earlier. As more purposeful strategic planning takes place in OSS 2.0, complementary development strategies will be enacted to provide a complete portfolio of open source products. Likewise, in the education sector, some cooperative initiatives have emerged to promote use of open source (e.g., www.osef.org; www.ossite.org; www.schoolforge.net).

Safeguarding against IPR Infringement

While open source was a fringe phenomenon, its relative obscurity offered some safety from litigation. Once it entered the mainstream, the threat of litigation arising from IPR infringements became real – for example, the SCO Group's lawsuit against IBM over alleged patent infringement in Linux. The ultimate goal of IP protection mechanisms such as patents could be summarized as *the publication of non-trivial ideas, with an explicit guarantee of continued*

availability, which seeks to protect the interest of the small players, for the overall betterment of society, as others learn from and improve on the original ideas. Interestingly, this definition also captures the open source phenomenon well. However, the stimulation of innovation and creativity, which should be the fundamental rationale behind IP protection, has failed abjectly in the software area (Bessen and Hunt 2003). Ironically, even though open source has often been about replicating proprietary products, the ingenuity of the global development community has allowed innovative new functionality to emerge – for example, the OpenOffice suite and the Mozilla Firefox browser.

Warranties and indemnification against IP infringements are key issues for OSS 2.0. A number of initiatives exist, but all are limited in scope. For example, Red Hat offers a warranty against any infringement in its Red Hat Enterprise Linux distribution (although this warranty just promises that Red Hat will replace any infringing code). Similarly, Novell has offered customers of its SUSE Linux an indemnification against copyright (but not patent) infringements. HP also offers its customers an indemnification, but only for claims made by SCO. JBoss offers indemnification to its customers, but limited to the value of the customer's contract . Meanwhile, some third parties, such as Open Source Risk Management, are selling indemnification protection.

Concluding Remarks

The open source field today and the decision support systems (DSS) field in the past have interesting similarities. Both have drawn together a wide range of researchers from disparate disciplines. For DSS, however, the consequence has not been benign. For instance, Keen lamented DSS research having been "co-opted and trivialized…by lab-experiment-academics", and concluded that "identity is easily blurred and eroded when the purposive focus of the research is lost and the topic area then dominates" (Keen, 1991, pp.37-38). I believe a similar situation could occur in open source research – indeed, the problems could be exacerbated as researchers take advantage of the ready availability of large online data repositories (where much

of the data may be of little real value), and continue to focus their research efforts inwards on the phenomenon to repeatedly study project characteristics and developer motivation, for example. Such research has been valuable, but a more purposive agenda is needed – one that also looks outward at the open source phenomenon in general and at the emergent OSS 2.0 phenomenon in particular.

REFERENCES

Bessen, J and Hunt, R (2003) An empirical look at software patents, Working paper 03-17/R, http://www.researchoninnovation.org/swpat.pdf

Bezroukov, N (1999) OSS research as a special type of research, Vol. 4, No 10, http://www.firstmonday.org/issues/issue4_10/bezroukov/

Dinkelacker, J and Garg, P (2001) Applying Open Source Concepts to a Corporate Environment, in *Proceedings of 1st Workshop on Open Source Software Engineering*, ICSE2001, Toronto, available at http://opensource.ucc.ie/icse2001.

Everitt, P (2004) Zope: open source, revisited, First CALIBRE International Conference, Hague, Nov 2004 (available at www.calibre.ie).

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

Fichman, R (2004) Real Options and IT Platform Adoption: Implications for Theory and Practice, *Information Systems Research*, Vol 15, No 3

Fitzgerald, B. (2005) Has open source a future? In Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2004) (Eds) *Perspectives on Free and Open Source Softwar*e, MIT Press, Cambridge, pp. 121-140.

Fitzgerald, B. and Kenny, T. (2003) Open Source Software in the Trenches: Lessons from a Large Scale Implementation, *Proceedings of 24th International Conference on Information Systems*, Seattle, December 2003, pp. 316-326.

German, D. M. (2003) GNOME, a case of open source global software development, In International Workshop on Global Software Development, (co-located with ICSE 2003), pp. 39-43, <gsd2003.cs.uvic.ca/gsd2003proceedings.pdf>

Gurbani, V.K., Garvert, A. and Herbsleb, J.D. (2005) A Case Study of Open Source Tools and Practices in a Commercial Setting, In *Proceedings of the 5th Workshop on Open Source Software Engineering*, 27th International Conference on Software Engineering: ICSE 2005, St. Louis, 17 May 2005, pp. 24-29.

Hecker, F (2000) Setting up shop: the business of open-source software, http://www.hecker.org/writings/setting-up-shop

Keen, P. (1991) Keynote address: relevance and rigor in information systems research, in Nissen, H., Klein, H. and Hirschheim, R. (eds) (1991) *Information Systems Research: Contemporary Approaches and Emergent Traditions*, Elsevier Publishers, North Holland, 27-49.

Koenig, J. (2004) Seven open source business strategies for competitive advantage, *IT Manager's Journal*, May 15 2004

Krishnamurthy, S. (2005) "An Analysis of Open Source Business Models", in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge

Lyddy-Collins, N (2005) Perspectives on open-source software licensing policy in a commercial software development environment, Unpublished Masters Thesis, University of Limerick.

Mahadevan, B (2000) Business models for internet-based ecommerce: an anatomy, *California Management Review*, Vol. 42, No 4, pp. 55-69.

Matusow, J (2005) Shared source: the Microsoft perspective, forthcoming in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge.

Michlmayr, M, Hunt, F & Probert, D (2005) Quality practices and problems in free software, in Scotto, M and Succi, G (Eds) *Proceedings of First International Conference on Open Source (OSS2005)*, Genoa, 11-15 July 205, pp. 24-28.

Moore, G (1999) *Crossing the Chasm,* Harper, NY.

Norris, J (2004) Mission-critical development with open source software: lessons learned, *IEEE Software,* Vol. 21, No. 1, pp. 42-49.

O'Mahony, S (2005) Non-Profit Foundations and their Role in Community-Firm Software Collaboration, in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge, pp. 393-414.

O'Neill, Eoghan (2005) *An analysis of open source business models and partner networks*, Unpublished MSc Thesis, University College Cork.

Onetti, A & Capobianco, F (2005) Open source and business model innovation. the Funambol case, in Scotto, M and Succi, G (Eds) *Proceedings of First International Conference on Open Source (OSS2005)*, Genoa, 11-15 July 205, pp. 224-227.

Raymond, E. (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, US, O'Reilly.

Roy, A. (2003) Microsoft vs. Linux: Gaining Traction, *Chartered Financial Analyst*, Vol. 9, Issue 5, pp. 36-39.

Rusovan, S, Lawford, M and Parnas, D. (2005) Open Source Software Development: Future or Fad? in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge

Russo, B, Braghin, B, Gasperi, P, Sillitti, A & Succi, G (2005) Defining TCO for the transition to open source systems, in Scotto, M and Succi, G (Eds) *Proceedings of First International Conference on Open Source (OSS2005)*, Genoa, 11-15 July 205, pp. 108-112.

Sanders J. (1998) Linux, Open Source, and Software's Future, *IEEE Software* September/October 1998 pp 88–91

Schach, S., Jin, B., and Wright, D. (2002) Maintainability of the Linux kernel, in J Feller, B Fitzgerald, S Hissam, and K Lakhani (Eds.) *Proceedings of 2nd Workshop on Open Source Software Engineering*, ICSE2002, Orlando, Florida, available at http://opensource.ucc.ie/icse2002.

Torvalds, L. and Diamond, D. (2001) *Just for Fun: The Story of an Accidental Revolutionary*, Harper Collins, New York.

Tushman, M and Anderson, P (1986) Technological discontinuities and organizational environments, *Administrative Science Quarterly,* Vol, 31, pp.439-465.

Wheeler, D. (2005) Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! http://www.dwheeler.com/oss_fs_why.html

Acknowledgements

I would like to record my gratitude to Joe Feller, Rishab Aiyer Ghosh, Carlo Daffara and Maha Shaikh for feedback on this topic, and also to the MISQ reviewers. The work was supported by EU project grants, CALIBRE and COSPA, and by the Science Foundation Ireland Principal Investigator Grant, 02/IN.1/I108