Evaluation of open source software: comparative perspectives between proprietary and open code development

Authors:

Andrew Schofield
PhD student
Information Systems Institute
Salford University
Salford M5 4WT
UK

T. +44 161 295 5025
E. a.j.schofield@salford.ac.uk

Dr Amit Mitra
Director of Postgraduate Research
Information Systems Institute
Salford University
Salford M5 4WT
UK

T. +44 161 295 5537
E. a.mitra@salford.ac.uk

This paper addresses the issues connected with the evaluation of Information Systems developed using Open Source software. It identifies the key differences between traditional forms of development using proprietary software and Open Source methodologies and analyses the consequences of IS/IT evaluation.

Abstract

The Open Source movement and the general new interest and acceptance of Open Source software development indicates the approach of a paradigm shift, in the way Information Systems (IS) and Information Technology (IT) Evaluation should be carried out. With the use of Open Source software in IS design comes a whole new set of norms and standards which affect how IS/IT must be implemented and evaluated.

The majority of IS/IT Evaluation activities are performed from a perspective that has traditionally considered the business aspects to be of overriding importance. Such a focus is often less on the actual technical quality of the system and more on its potential value and the affect it will have on efficiency and efficacy of organisations. Managers who oversee such evaluation projects are often only concerned with implementation costs and how much money they could save or generate through its use. In any case, there are certain key issues that IS/IT Evaluation activities concentrate on. To evaluate these issues and the successfulness of a system, a company must have something to compare their system with. Apart from basic calculations based on profit, loss, assets and liabilities, and perhaps comparisons with other similar evaluation projects which have taken place within or outside the business in question, there may also be an implicit set of norms which organisations base their evaluations upon.

Open Source software development however, puts many factors, including costing, into an entirely different perspective in comparison to traditional styles of system development. Traditional system development usually includes use of proprietary software outsourced from a bespoke developer, the use of off-the-shelf software, or the complete in-house development of a system. Open Source operates with different rules which means that many of the well established factors of IS/IT Evaluation are no longer relevant or are completely inappropriate when applied to Open Systems.

The present paper attempts to address the issue of Open Systems Evaluation. Specifically the paper will examine fundamental differences between traditional IS software systems development, and projects which make use of Open Source software and ideologies. This will be accomplished by constructing a framework that illustrates the differences between the two. Upon defining the differences the paper will investigate how Open Systems, or more specifically systems making use of Open Source software, can be evaluated with the emphasis on the differences when compared to traditional, proprietary based evaluation.

Key words: Open source, proprietary software, bespoke software, IS evaluation

Introduction

Information Technology (IT) is in a constant state of upheaval, evolution and change. Consequently the definitions of standards with regard to performance and effectiveness are continually being re-defined. Trends indicate that this situation will never change and that technology will continue to progress in the same way (c.f. Wolstenholme et al 1993, Brooks 1995).

Open Source software (OSS) has been the biggest change is IS development in recent times and many authors (c.f. Moody 2001; Pavlicek 2000; Raymond 1999, Stallman 1999) have described the phenomenon as a revolution. We use the term OSS to generally refer to all Free and/or Open Source software, which may be freely altered, exchanged and to which the source code is available. It is acknowledged that there are variants of the definition of OSS but hopefully the context in which the term is used within the paper will make it clear which specific part of the definition attention is being drawn to. Considering how fast technology is changing and how the OSS mode of operation is now being practiced by many organisations around the world, it is important that the general principles of IS/IT Evaluation recognise that OSS differs from traditional development.

Research in the IS/IT Evaluation realm has tended to concentrate on several main areas. George (2000) identifies three main aspects of IT Evaluation research

Motivation for IT Evaluation Research: What & When	Purpose: efficiency, effectiveness, understanding
	Nature of research
	Theory base and reference discipline
Focus of IT Evaluation Research: Why	Level of Analysis
	Scope of IT
	Stage in life-cycle of the IT
	Frequency of evaluation
Research Approaches in IT Evaluation Research: How	Approaches: objective or subjective
	Research model
	Measurement issues
	Data analysis

Table 1: Aspects of IT Evaluation Research Source: (George 2000, p.1096)

Table 1 illustrates the issues that are commonly addressed by IT Evaluation research. The aspects shown on the right on table 1 are fairly generic but when applied to OSS models of system development, it is conceivable that many factors will be affected. Particularly noteworthy issues are those of the IT life-cycle aspects, measurement issues and data analysis. These issues are particularly relevant to OSS evaluation for reasons that should be made apparent by this paper. Sarefeimidis & Smithson (2003) state that there are gaps between theoretical work of IS evaluation and how it is practically implemented. With the difference between OSS and traditional techniques being overlooked, these gaps seem almost certain to widen as OSS gains acceptance.

IS/IT Evaluation basically consists of comparing the system with some standard. Although this may not be explicitly defined in the literature it is implied in many literary definitions of IS/IT Evaluation (c.f. Irani et al 2000a; Irani 2002; Jones 2000;). Evaluation of an IS

therefore needs two things to compare it to. Firstly a set of organisational requirements is needed to compare with the capabilities of the current system. A definition of what is considered to be an acceptable level of functionality and performance must be defined and the actual system compared with these parameters. Secondly there must also be an acknowledgement of how these levels compare to wider norms and standards within the IS domain. These could be considered as internal and external comparison factors i.e. comparisons made with a specification developed inside the organisation, and those defined outside it, such as the capabilities of a competitor. In business and management terms this is described as the external business environment, the analysis of which is considered essential for a business to be successful. (Capon, 2000; Worthington & Britton, 2000). To be able to function in the business field, especially the highly electronic arena that exists today, an organisation's evaluations must also look towards the outside world, and not just in towards itself.

This paper attempts to identify major differences between proprietary and OSS systems development in terms of evaluation using literature such as the above as an evaluation framework.

Open Source Evaluation

Evaluation Factors

Within both proprietary and OSS systems development, the variety of methods used from one organisation to the next coupled with more intangible factors such as motivation, organisational politics and self-interest, result in a myriad of different classifications of projects (Schofield & Mitra 2004). However, the key differences between proprietary and OSS development lie in the issues of licensing, group development and peer review, and economics.

Licensing

Flexibility is OS software's greatest asset and one of the primary reasons for people and organisations choosing it as an alternative approach to proprietary solutions. The flexibility is made possible because of some very clever licensing documents such as the GNU General Public License (GPL). (See www.gnu.org for more information). The most common use of OSS licences allows the code for a piece of software to be freely accessed, distributed and altered to fit the user's needs. The software is in most cases available free of charge as well as being free of restriction. The only restriction in licenses such as the GPL, is that the freedom it establishes can never be removed and must be passed on to those who receive the software. This freedom is achieved by allowing the source code for a program to be seen and altered by anyone. The customisability of the software thus allows a software system to be tailored to the individual needs and requirements of an organisation. This has substantial ramifications for IS Development and consequently IS Evaluation. Organisations implementing Information Systems have traditionally had two options open to them

1. Purchase a proprietary solution by using off-the-shelf technology and software
2. To develop a system in house, perhaps involving the use of some off-the-shelf software or to outsource the bespoke development to another company.

Both these approaches mean that IS Evaluation activities will be looking at the system as a fairly fixed or rigid system. If an off-the-shelf system is used, it will be fairly generic and therefore the evaluation will be looking at how well the system fits into the working processes of the organisation. This is the major weak point of off-the-shelf solutions caused by the inflexibility and fixed nature of this kind of software. This of course depends on the size of the organisation and the scale of the IS project. Smaller companies with limited technological requirements, may have many more choice and alternative solutions that the larger organisations. Larger organisations tend to need more specific or bespoke software as their business activities, like their system requirements, are more complicated. The question then becomes, what does the organisation do with the results of the evaluation? Due to this inherent inflexibility it seems fair to conclude that the organisation may have to alter its working practices and processes, to fit into the requirements of the IS system (c.f. Hammer and Champy 2001). This is the opposite of what an organisation tries to achieve by implementing an IS and is clearly not a desirable scenario. As a result of this however, many organisations will simply put up with systems that are not best suited to them (c.f. Irani et al 2000b; Khalifa et al 2000). It is therefore suggested that the customisable nature of OSS facilitates a kind of system development which is more in line with business requirements an operations.

For an organisation developing an IS using OSS, the circumstances are somewhat different. Software developed in house can be done so much more efficiently due to the freedom associated with OSS. Firstly there is a large amount of OSS available which an organisation can use. The majority of this software is of a very high quality and much of it can be used as it is. Research by Fitzgerald & Kenny (2003) indicates that organisations may certainly be able to operate using entirely OSS with little or no code development work. As proprietary software systems also need some work to install and configure, there is little difference in this regard between the two types of software. Secondly, as most OSS licenses are designed to prevent restriction rather than enforce it, post-development operating costs will also be reduced due to the lack of need to renew software licenses. The real advantage of OSS is however the ability to alter, adapt and customise a system. If a piece of software does not do the job, the organisation can alter its functionality. Providing the necessary skills are in the organisation, this almost guarantees a system that fulfils the requirements if it developed correctly.

Development

The methods and techniques used for development of OSS are completely different from the traditional proprietary methods. The traditional development cycle depicts a methodology used in an office with a development team working together to engineer software from scratch. Although this approach is often adopted there are drawbacks to the way the system operates. The now famous work of Brooks (1995) demonstrated that the inefficient part of the system was the communication overhead between the developers. However OSS development avoids these problems because of the way it can make use of the OSS community (See figure 1).

As figure 1 shows, after the initial development of the software, the prototype is released into the community for all to use, and more importantly test and improve. This leads to the realisation that testing is no longer just a phase in the project life cycle, as it often in traditional development projects, but rather that it is an intrinsic part in the communal development process. Several authors (c.f. Raymond, 2000; Pavlicek, 2000; Moody, 2001; Evers, 2000) have identified this factor as one of the key reasons for the success of the OSS

development methodology. Lanzara & Morner (2003 p.1) state *"Technology, rather than formal or informal organization, embodies most of the conditions for governance in open-source software projects",* suggesting that technological determinism has a role to play in the process.

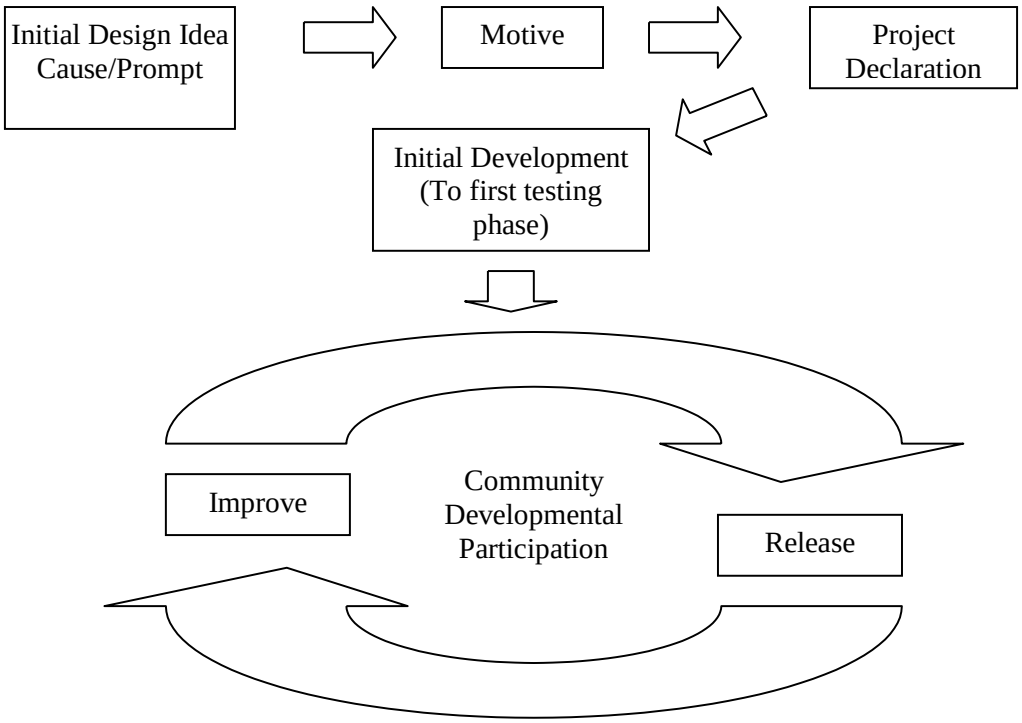

Figure 1: Conceptual Context-free Open Source Software Development Model Adapted from Raymond (2000)

The advantage of being open to organisations developing in-house software is that the community will contribute to the development of the code in several ways. In most cases, OSS developers participating in projects in this way are intrinsically motivated because they have chosen to take part in the project themselves. This may also explain the high quality of the software produced in OSS circles (c.f. Hertel et al, 2003; Lakhani & Wolf, 2003). These external co-developers may contribute to the project by finding and reporting bugs or by actually fixing them themselves and submitting the fixes to the core developers for consideration. It is also common for these contributors to propose improvements or new features, write them themselves and submit them for inclusion into the program. Figure 2 shows how this is achieved. The co-developers operate within there own micro-environments and contribute voluntarily to the larger macro-project. Because of the methods of communication which are dictated by the technology, there is not the same communication overhead as is found in tradition development. Organisations may therefore not only make use of existing software as a foundation to build there software upon, but may also benefit from other people's development of the code.

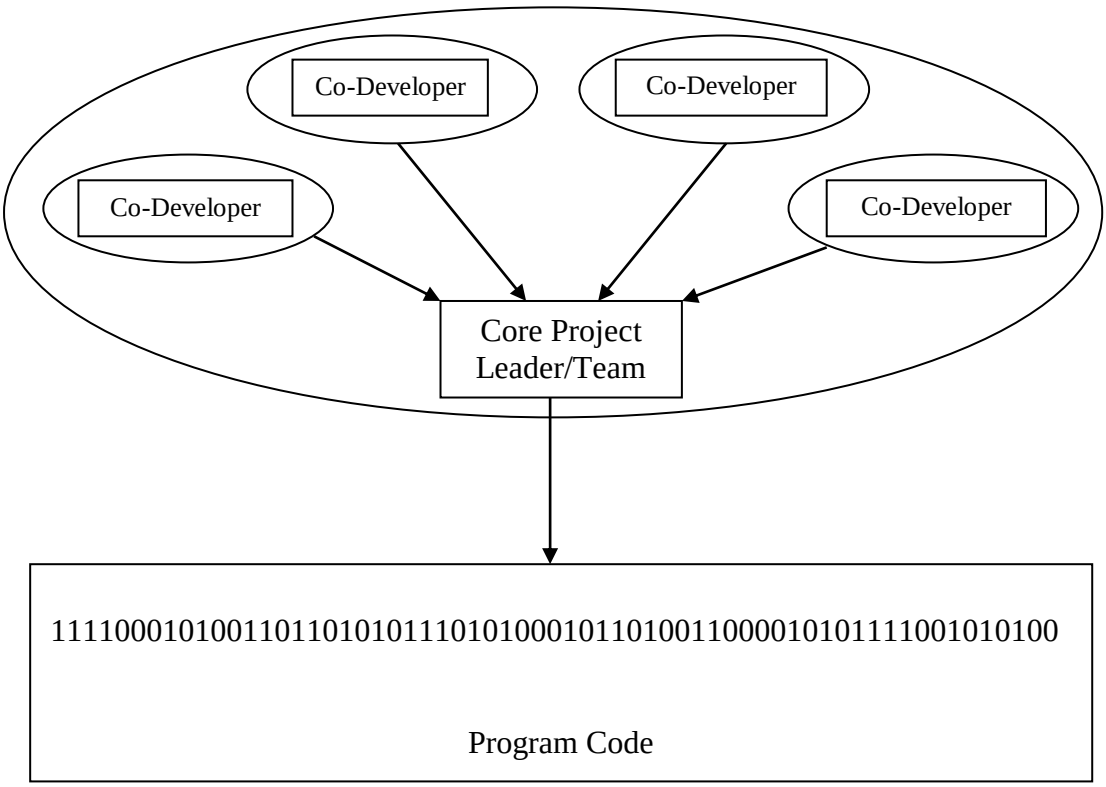

Figure 2: Open Source Structural Development

The differences in the method of development are very important for the evaluation of organisation's software system. As was discussed earlier, evaluation is concerned with comparing the system as it currently is, to a set of requirements. A possible risk with the OSS style of development is that third party developments i.e. those done by external co-developers in the OSS community will not be developed specifically with the organisation in mind. This mismatch between the requirements and the developer has long been a topic for discussion in software engineering literature as it is often the cause of ineffectual software.

Economics

Proprietary software is usually written in order to generate profit for the developing company. On the flip side, companies normally buy software because they believe it will benefit the organisation in some way, perhaps to improve efficiency or effectiveness. This highlights what Raymond (1999) defines as the difference between "use value" and "sale value". To the developer, sale value is the most important aspect. To the user, use value is more important. IS/IT Evaluation has traditionally focused on the economical factors of IS development and implementation to basically judge whether the system is economically viable. Again OSS is entirely different. Firstly because the majority of OSS is available for free. This means that the monetary value of the software is no indication of its viability as an effective software tool. Put simply, asking the question "is it value for money", isn't particularly relevant when using free OSS as the answer invariably going to be yes. Therefore economically based forms of IS evaluation are not appropriate for use with OSS.

If OSS is developed by an organisation, whether it starts from scratch or improves some existing software, the organisation will have the cost of paying the developers and possibly training them as well. However by making use of the OSS community, the organisation could get a significant amount of its development work done for it by unpaid volunteers. Raymond (2000) states that one of the reasons for the quality of OSS is that the external co-developers are not hand picked by the core developers. It is this randomness and self selection process which introduces many varying perspectives and approaches allowing problems to be found and solutions proposed in different and original ways. Organisations themselves may participate in this exchange of ideas and code and work together to produce better systems (Pavlicek 2000). As specific types of organisations have specific requirements, they may work together to produce a system which will meet both their needs. Linux is an excellent example of this. Different distributions of the operating systems have been created for different purposes e.g. Linux for embedded systems, or systems where the availability and robustness are of paramount importance. For organisations, this is very good economical model as it shares the cost of development whilst improving the likelihood of a quality system being produced. Evaluation therefore needs to consider the amount of valuable resources that the organisation can make use of at no cost and not just how much of the direct costs can be cut by using OSS.

The adoption of the OSS approach will affect companies who choose to outsource the development of a customised software system. Although the developing company will charge for the design, installation and maintenance of the system, they will be in a position to offer these services at a lower price. This is because in many OSS cases, code from existing software can be re-used, therefore there will not be as much actual development work but rather adaptation and customisation. When managers are presented with a choice between two companies offering similar solutions and one has a lower price tag, it seems fairly certain that they will choose the most economical option.

Maintenance

Many IS/IT evaluations take place after a new product has been installed and has been in use for some time (c.f. Jackson, 2001; Irani, 2002; George, 2000). The issue of maintenance then becomes an important issue. This aspect is apparent in the literature on Legacy systems and demonstrates that IS/IT evaluation should also be used to identify problems with existing systems (Bennett, 1995; Bocij et al, 1999; Brodie & Stonebraker, 1995; Ptak et al, 1999). This is an area where proprietary and OSS systems differ considerably. In the majority of cases proprietary software, when purchased, includes certain guarantees as to the performance and effectiveness of the product. Included in these agreements are the provision for support and assistance with problems. How far this agreement goes depends on the type of software and the license provided. OSS solutions provided by a company also usually include support but OSS acquired from other sources such as the Internet, will usually give no guarantee or direct support. This is a common concern for large organisation developing in-house OSS solutions.

No support for a software system is something that most IT managers would find appalling. That is unless of course they were aware of the support available from the OSS community. As was discussed earlier, the OSS community is extremely efficient at finding and fixing bugs. From a maintenance point of view having this amount of support is a very positive thing, even if it does not take the same form as support for proprietary products. Fizgerald & Kenny (2003) observed that bulletin boards and other forms of online community based

forums were the main source of support for their case study. This may not be an appealing thought to many however, as it almost seems as though no one is accountable for the software.

The alternative is the outsourced OSS approach. Companies which specialise in developing bespoke software based on OSS code almost invariably offer support for there systems. This is often the only way of generating profit. As Raymond (2000 p.12) states *"Give Away Recipe, Open A Restaurant"*. IS/IT evaluators must consider how maintainable a system is. If all support has been removed for a product, it is questionable if the system is a viable asset to the company.

Factorial Analysis

It can be seen that there are many differences between OSS and proprietary software. Not just in the software itself, but the development styles used, the strategies available to organisations, the economical aspects and the more vague and intangible factors of community behaviour and motivational aspects. As a basic framework there are definable factors which relate to, and have consequences for the evaluation of an IS. Table 2 shows these factors and the consequences for both OSS and proprietary systems.

	Open Source	Proprietary
Flexibility	Ability to alter code to suit organisational needs	Alteration needs special permission from owner. (May not be given)
	Other work may be used/integrated into design	Software may not be used in other programs.
Availability	Software may be acquired without a license.	Software may require initial payment for licensed use.
	No license renewal	License may require renewal payment.
	No limit to use.	Number of available copies and type of use may be restricted by license type.
	More training maybe required before staff can use OSS.	Some training maybe required but many parts may already be familiar.
Reliability	Unless outsourced, OSS rarely comes with guarantees.	License normally includes guarantee and support
	OSS community will most likely support any faults.	Developer will fully support own software but probably only for a limited time.
	Software will be supported almost indefinitely by OSS community	Software will be supported until developer removes support or ceases trading.
	Code developed by external co-developers will not be customised for the specific organisation.	All non-off-the-shelf software will be tailored to the organisation.
Returns (on investment)	In house developments may efficiently make use of existing code in community so a smaller work force may be needed.	In house developments must be done from scratch or on pre-authored code by a suitably sized development team.
	Outsourced code can still be supported by community	Out sourced code may be supported only by developer.
	Software maybe acquired free of charge.	Software will cost an amount specified by developer.
	Community support is free, outsourced support will be charged for.	All support from developer will be charged for.
	Software acquired at no cost may often be used ‘out of the box’ i.e. no development work required.	Purchased software must be used in its ‘out of the box’ state.
	Cost of training or hiring skilled staff maybe high. E.g. Linux certified engineers are rarer than Microsoft engineers.	Training and hiring costs will be fairly low as skills are common.

Table 2: Factorial Analysis of Open and Proprietary Evaluation Aspects

The above framework highlights the differences between proprietary and OSS based IS. It is evident that there are large differences in almost all areas. Given the above, it seems a foregone conclusion that any attempt to evaluate an OSS system using proprietary standards will give inappropriate results.

Relevance to Practitioners

For those involved in IS/IT evaluation, it should now be evident that implementing an OSS solution is very different from implementing a proprietary one. A different approach to evaluation must be taken and the above factors need to be considered in order to get a realistic and useful result.

A key issues in the evaluation and use of OSS is one of perspective. Evaluators and system developers making use of OSS need to get out of the traditional mindset that views software and systems as property. Although it may seem unrealistic, a approach should be adopted that views software as sharable tools that can be freely interchanged between individuals and organisations alike without any loss of profit. For those involved in the financial aspects of IS/IT evaluation, awareness of the huge difference in economical factors involved is essential. A very good example is Linux, which is available in certain distributions with around 3000 software packages. These are complete systems comparable to Microsoft Windows, and available completely free of charge. As was stated earlier, some OSS solution providers will also charge a fee but will supply guaranteed support. Fitzgerald and Kenny's (2003) research indicated that a full OSS implementation of an organisation system, including development and support, was approximately 25% of the cost of implementing the proprietary alternative. Evaluators must be aware of these huge differences in order to correctly evaluate the usefulness and value of an OSS based IS.

Conclusions

The present paper has attempted to call attention to the intrinsic differences between OSS and proprietary software systems and how these differences affect the act of IS/IT evaluation. Analysis of the factors led to the creation of four groups; Licensing, Development, Economics and Maintenance. These four groups encompass the focal areas relative to IS/IT evaluation from the legal issues relating to software use and development to issues of flexibility and efficacy of developmental methods, the financial considerations, and the separate issue of post-developmental support and continuous availability.

Traditionally evaluation has focused on the financial aspects of investing in an IS. In many cases this is still true today. The methods used in these situations do not take the larger picture into account and also provide a skewed view of the economical factors. The return on investment (ROI) method is a good example of an evaluation method that does not provide reliable results. We have seen that OSS is a completely different approach that will almost certainly render many of these methods inappropriate. There may also be several political and social facets to an evaluation which would not arise with the use of proprietary systems and vice versa. There is often a sense of reluctance to make the move to OSS as it is still a new and developing area. For IS/IT evaluation, this is an important issue especially when economics are involved. Fitzgerald & Kenny (2003) found that the view of an IT manager was;

"If you have a product which costs €1 million-it may seem appropriate to spend €500,000 on consulting. However if the product costs nothing, then spending €500,000 somehow seems to be a more difficult decision to take, yet the saving is still €1 million".

Fitzgerald & Kenny (2003 p.324)

The biggest obstacle to the evaluation of OSS systems is perhaps the same as that which is preventing the mass adoption of OSS as a viable business model. Business is based on competition and the use of IT is, in many people's view, a tool for leveraging competitive advantage. Therefore the premise of giving away developed software, sharing it with others who may well be direct market competitors, and even getting software at no cost does not compute when compared to the traditional business model. Consequently it seems fair to conclude that companies developing systems in co-operation with other companies or the OSS community should approach IS/IT evaluation with the same holistic view.

From the very beginning of the paper the importance of comparable norms in IS/IT evaluation have clearly been important. One message that has been apparent throughout this research is that OSS brings a new set of standards to the equation. From a practical perspective standards are important. A basic example would be Microsoft Office which commands the bulk of the office suite market. MS Office file formats have consequently become a de facto standard. To succeed in today's communication rich business world, companies must be able to exchange data in these common formats. OSS office suites often now support these formats, which is an indication of the increasing realisation of the importance of standards. In most cases, with the exception of server technology, OSS is not considered as standard or even as being in common use. To this end, Investment analysis and evaluation attempts will dub OSS as being inappropriate even if it performs much better than the proprietary counterpart.

A catch 22 situation now seems to exist wherein companies are reluctant to invest in OSS solutions, which may be unable to grow in popularity if not backed by companies who feedback into the OSS community. Having said this, some large and very influential companies, such as IBM, Netscape and Oracle have invested in OSS and if the future is indeed 'open', IS/IT evaluation methods will need to be altered to bring them in line with the new ways.

References

Bennett. K. H (1995), "Legacy Systems: Coping With Success*", IEEE Software, January 1995,* Vol 12, No. 1

Bocij. P, Chaffey. D, Greasley. A, Hickie. S (1999) "*Business Information Systems: Technology, Development and Management",* Financial Times Management, Pitman Publishing, UK

Brodie. M. L & Stonebraker. M (1995) "*Migrating Legacy Systems - Gateways, interfaces & the incremental approach"*, Morgan Kaufmann Publishers, Inc., California, USA

Brooks, F.P., (1995), *"The Mythical Man Month",* Addison Wesley Longman, Inc.

Capon, C. (2000), *"Understanding Organisational Context",* Prentice Hall

Evers, S. (2000), "An Introduction to Open Source Software Development", Available at: http://user.cs.tu-berlin.de/~tron/opensource/opensource.html, Accessed (February 2004)

Fitzgerald, B. and Kenny, T. (2003), "Open Source Software in the Trenches: Lessons from a Large-scale OSS Implementation", *Twenty-Fourth International Conference on Information Systems,* Seattle, Washington, USA, December 14-17

George, B. (2000), "A Framework for IT Evluation Research", *Americas Conference on Information System 2000,* Long Beach California, USA

Hammer, M., Champy, J. (2001) *"Reengineering the Corporation: A manifesto for Business Revolution",* Nicholas Brealey, UK

Hertel, G., Niedner, S., Herrmann, S. (2003), "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel", *Research Policy, Special Issue on Open Source Software Development,* Available at: http://opensource.mit.edu/papers/hertel.pdf , Accessed (February 2004)

Irani, Z., Love, P., Zairi, M. (2000a), "Information Systems Evaluation", *Mini-track Introduction, Americas Conference on Information Systems 2000,* Long Beach California, USA

Irani, Z., Love, P., Hides, T.M. (2000b), "Investment Evaluation of New Technology: Integrating IT/IS Cost Management into Model", *Americas Conference on Information Systems 2000,* Long Beach California, USA

Irani, Z., (2002), "Information Systems evlatuation: navigating through the problem domain", *Information & Management 40,* pp. 11-24.

Jackson, M. (2001), "What's so important about evalution?", *Library Management,* Vol 22, No. 1/2, pp. 50-57.

Jones, S., Hughes, J. (2000), "Understanding IS Evaluation as a Complex Social Process", *Americas Conference on Information Systems 2000,* Long Beach California, USA

Khalifa, G., Irani, Z., Baldwin, L.P. (2000), "IT Evaluation: Drivers and Consequences", *Americas Conference on Information Systems 2000,* Long Beach California, USA

Lakhani, K. R., Wolf, R.G. (2003), *"Why Hackers Do What They Do: Understanding Motivation Effort in Free Open Source Software Projects",* MIT Sloan School of Management Working paper, Available at: http://freesoftware.mit.edu/papers/lakhaniwolf.pdf, Accessed (February 2004)

Lanzara, G.F. and Morner, M. (2003), 'The knowledge ecology of Open-Source Software Projects', paper presented at seminar on '*ICTs in the contemporary world*' at LSE Department of Information Systems on 2nd October

Moody, G. (2001), *"Rebel Code: How Linus Torvalds, Linux and the Open Source Movement Are Outmastering Microsoft",* The Penguin Press, UK

Pavlicek, R. C. (2000), *"Embracing Insanity: Open Source Software Development",* Sams Publishing, USA

Ptak. R. L, Morgenthal. JP, Forge. S (1999) "*Manager's Guide to Distributed Environment: From Legacy to Living Systems",* John Wiley and Sons Inc, NY, USA

Raymond, E. S, (1999), "The Magic Cauldron", *Available at: http http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/* (Accessed December 2003)

Raymond, E. S, (2000), "The Cathedral and the Bazaar", Available at: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ (Accessed November 2003) and in Dibona et al (1999)

Schofield, A., Mitra, A. (2004), "Complexities of Classifying Open Source: Developing a Framwork for Categorising Open Software Development", *UK Academy of Information Systems conference 2004,* Glasgow, UK.

Serafeimidis, V. & Smithson, S. (2003), "Information systems evaluation as an organizational institution – experience from a case study", *Information Systems Journal 2003 13,* pp. 251-274.

Smithson, S. & Hirschheim, R. (1998) "Analysing information systems evalutation: another look at an old problem", *European Journal of Information Systems,* pp. 158-174.

Stallman, R. (1999*), "The GNU Operating System and the Free Software Movement",* in Dibona et al (1999).

Wolstenholme, E.F., Henderson, S., Gavine, A., (1993) "*The Evlaution of Management Information Systems: A Dynamic and Holistic Approach",* John Wiley & Sons Ltd, UK.

Worthington, I. & Britton, C. (2000) *"The Business Environment",* 3rd edition, Prentice Hall..