SOS-ware DEVILS

[Strategic Open Software DEVelopment ILlnesseS]

George Blanas, PhD, Assistant Professor

Department of Project Management, TEI of Larissa, Greece

blanas@teilar.gr

Certain categories of software play a strategic role in contemporary public and private organizations. While software use is accelerated and diffused to more and more people and organisations, software development follows a reverse trend where fewer players form oligopolies, with some of them having almost reached a state of monopoly in certain areas. The evil consequences of such an evolution can be numerous, some of them relate to economic and security dependence and some others to phenomena of knowledge dependence and hysteresis.

Some propose that a resistance to these accelerations and trends is still possible and the solution is the development of open software. While this proposition can be true in principle, there are several factors that impede the development of software that can strategically compete with closed source software. Those impediments are found either in the selection of the type of software that is strategically necessary, or the project development obstacles specific to open software development.

Within the current paper, we formulate a general framework that categorises the types of illnesses in open strategic software development from a number of viewpoints and the types of damages that could be inflicted to organizations and states as a result of false expectations if these illnesses persist. Finally, we identify the areas where research is considered to be urgently needed.

Keywords: Strategy, open, software

1. Introduction

According to Benkler (2002) there is a need for bettering of institutional conditions, to facilitate creative development and human collaboration and to decrease the ability of various mechanisms for private appropriation of public goods like the open source software (OSS).

We argue that the appropriation of OSS is only the seen part of an iceberg. Most of proprietary software is developed based on knowledge and algorithms that have been developed in the academia that are free. The prices of proprietary software do not reflect the fact that most of its value is a collection of values that correspond to efforts and knowledge that is not part of it. This argument can be easily. Profound example is MS-Windows which takes most of its value from the X-Windows project, but its price reflects its dominant monopoly position instead. While in other goods added value can be easily distinguished and separated, software's hidden nature creates false impressions to most consumers who can only see some of its characteristics but not its fabric that is mostly borrowed.

According to Chiao (2003) 'there are puzzles about why software that costs billions has been built but distributed using royalty- free licenses, and why people volunteer to contribute code. Without investigating real world constraints, analysts turn to alter utility functions when they are faced with incongruence between hypotheses and data. Since these functions are never observable, analysts have a high degree of freedom to tamper with their forms such that one can always find a new "theory" that fits the data. In a polar case, given a series of observations, one can always fit the data perfectly by using a "theory" that specifies a dummy variable for each observation. Such "theories" are generally ad hoc. One example is the recent surge of altruistic utility functions'. As Bonaccorsi & Rossi (2003) notice, fortunately, it is not a problem of the real world, which quickly moves on even without economic theory. Raymond (1998) claimed that open source programmers wish to establish a reputation for ingenuity in the greater hacker community. While Linus Tornalds has put forward the argument that the creation of software is a just for fun activity and has written a book on it (2001), economic theorists have avoided to explore this view, probably because there is a void in relative literature in other areas of economic activity. We argue that the old acknowledgement that software is an art should be incorporated in combination with Maslov's hierarchy of needs. People who contribute high quality

software, especially people who lead these efforts have satisfied most of the lower end needs and seek recognition or self fulfilment. The relation of software quality to people quality is an issue that requires empirical research. On the other hand, it is common today for big companies like IBM or Sun and HP to support OSS projects and to employ programmers to complete them. There is a need for empirical research on the people characteristics employed in these projects, as they may change the ways that OSS is developed.

The economics of OSS have been analysed from several viewpoints from many researchers. The research is relatively new and incomplete in its core. In the following section, we describe a framework for an integrated analysis.

2. FINE - Framework for Information Network nodEs

Blanas (2003) has developed a general framework on the quality of information systems development. The framework is based on the networking paradigm and focuses on the operation, management and evolution of a network node (Fig 1)

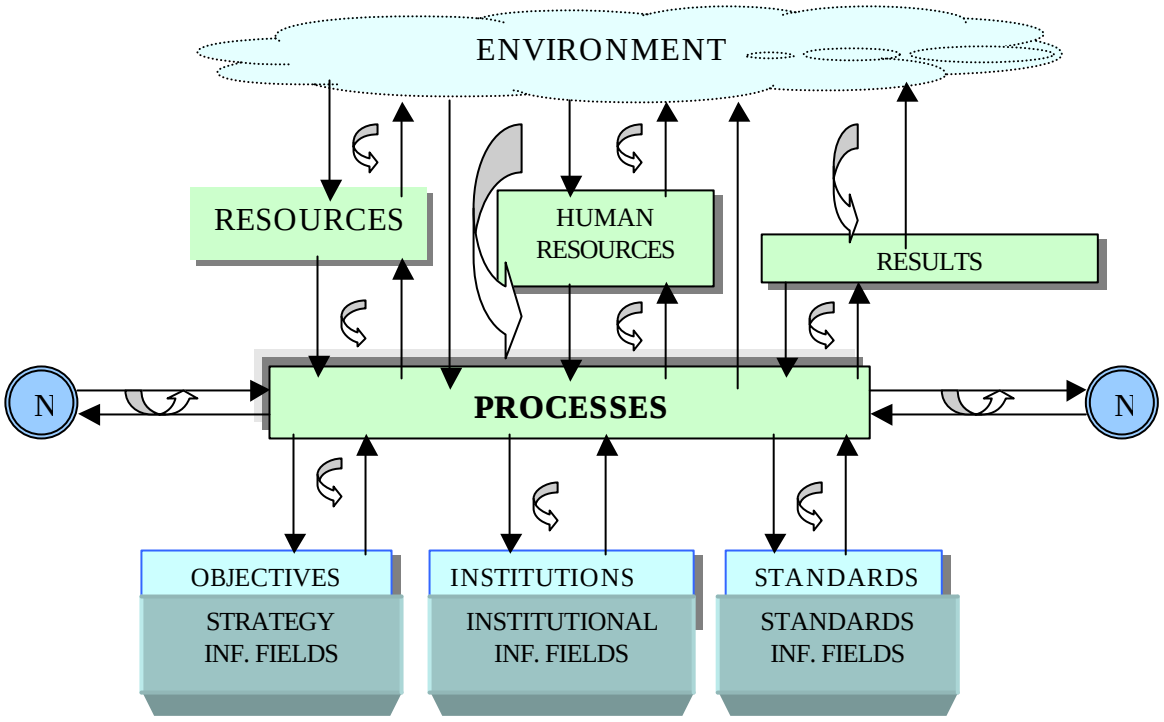

Figure 1 INFORMATION NETWORK NODE (Blanas, 2003)

The basic element of an information system described as a network is the node and the connections to and from it. The recognition and description of the processes in an

information system is important in order to understand the operation of the information network.

The node is able to enclose a number of capabilities and quality characteristics in various extents and intensities. In figure 1 we can distinguish first and second level feedbacks.

The first level feedbacks are the following:

[1.1] Processes → Human Resources → Processes

Human resource management on the development of leadership and motivation, evaluation, education, and training.

[1.2] Processes → Resources → Processes

Algorithm application, processing, storage requirements, access, evaluation, can be of strategic, managerial, administrative, operational type in respect to the value of information, level of automation, disaggregation and security, processing speed, storage capacity, and cost. Information has immediate relationship to the storage media and the access mechanisms.

[1.3] Processes → [Results →] Environment → Processes

Interface processes with shareholders, clients and citizens, operations, also on processes of service, evaluation, and development of new products and technologies. The level of understanding of the interface information depends on the level of asymmetry between institutional structures for the communication and processing of information.

[1.4] Processes → Objectives → Processes

The processes (conformance, participation, evaluation) refer to the level of recognition of environmental problems or chances, and the adaptation of the node to them, and they contribute to the acceptance, development and application of strategy. The configuration of objectives can become the main second level process for a strategic information field.

[1.5] Processes → Institutions → Processes

Processes of institutional-managerial type (compliance, participation, evaluation) related to policy making, compliance nad participation to existing institutions and recording of any adaptation difficulties.

[1.6] Processes → Standards → Processes

Organization, operation, administration, assurance, logistics of projects and procedures, based on compliance, use and evaluation of standards. The standardization of information flows reflects the regulations and conventions of management. Many standards are immediate results of political-legislative instititionalization..

[1.7] Processes → Node → Processes

Outsourcing of subprojects and procedures, access to information residing in other nodes (networking) and on the interactions with other nodes' processes.

The second level feedbacks are the following:

[2.1] Results → Environment → Results

Environmental evaluation of node results to the environment.

[2.2] Resources → Environment → Resources

Environmental evaluation of procurement management and environmental issues management.

[2.3] Human Resources → Environment → Human Resources

Environmental evaluation of human resource management criteria and processes.

[2.4] Objectives → Strategic Information Fields → Objectives

Configuration of information field strategies.

[2.5] Objectives → Institutional Information Fields → Objectives

Configuration of information field institutions.

[2.6] Objectives → Standards' Information Fields → Objectives

Configuration of information field standards.

In the proposed framework, we consider that problems are developed in cases of loss of equilibrium or asymmetry in communication across mechanisms. Equilibrium can be lost in cases of lesser capabilities or lost opportunities for learning or adaptation. Asymmetry can be developed from incomplete information or from control of the information flows. The capabilities of the various feedback mechanisms configure the capability maturity level of the node.

The first level feedbacks continuously improve the processes that comply to the current objectives, standards and institutions, using the evaluation processes and collect meta-information for the evaluation system.

The second level feedbacks use the meta-information from the first level and the existing environmental knowledge with probable use of benchmarking and propose the development of new objectives, institutions and standards.

If some feedback falls behind, that is an indication of a deficiency in resources, institutions, capabilities, environmental scanning, or will.

It is profound that the ability of a node to selectively diffuse or protect the information residing in its local memory depends on the corresponding abilities of the related nodes within the network

In the following section, we review some strategic OSS development illnesses under the FINE framework in order to detect the losses of equilibrium or the asymmetries observed in OSS development.

3. OSS DEVILS under the FINE framework

There are still many issues in OSS development that have not been researched. We present some of these issues and we examine their possible impact using the second level of the FINE framework. Our analysis is focused on the strategic information fields with certain extensions to the institutional and standards' information fields for its completeness in those cases that they are strongly related.

3.1 Strategic information fields

It is to be seen whether OSS will succeed. The control of information fields is a very sensitive area for economic and security interests. The more OSS succeeds the more those who control the information fields today will try to control the OSS nodes. The crucial nodes are the private software distribution companies who have strategically

assumed additional roles like software development. The nature of software as a commodity leads naturally to the development of oligopolies. The tendency for oligopolies will accelerate with the provision of extra software capabilities and support that fewer distributions will be able to provide. These capabilities may not be OSS or they may come under different kinds of licenses and depend on various political and strategic networks. The differentiation of the companies that are strategically positioned in the OSS market today is already happening. The power of control that these companies will be able to exert in the future depends on the success of OSS, the size of software, and on market concentration. OSS is supposed to be able to free people or countries from restricted private control of the strategic information fields. As an example of restricted private control, only one country – Israel - except the USA can replace the operating systems in strategic weapons like military aircrafts today, and another two – France and Russia – can use their own software systems on their own aircrafts. The control of the aircraft operating system results in the remote control of the aircraft itself. We need to answer the following questions today: ***Who could – would prevent one or more of these oligopolies at some stage to diffuse incomplete or viral software to different customers - like the closed source companies have the capability to do undetected today? Who could be able to detect these additions or alterations?*** Tracking of changes made is almost impossible to be done in time by anybody else except for the developers-distributors themselves. We know that the volume and complexity of the code in some strategic software like the operating systems excludes but very few from this role, regardless of the fact that the code is 'visible' since it is open. There is a need for a strategy that will be able to reassure that this won't happen. A better strategy would be to try to find ways to facilitate free markets and prevent oligopolies to happen. If this strategy cannot succeed, we must be prepared to protect our societies from the power of restricted private interests to control the information fields.

Even in an ideal situation of non-controlled distributors giving away viral free OSS, the dangers of knowledge dependence and hysteresis would result in the widening of knowledge and development capability gaps that would result in asymmetries of information and control between OSS diamonds and the rest of the world. ***How could these asymmetries could be alleviated? Do we need copies of existing economic diamonds to be developed at home, even if we develop the wheel again? Can this be***

done? Do we need to answer questions on critical masses and investment requirements?

Markets use software to advance their investment efficiency in all types of industries. Certain type of software is needed for the work to be done effectively. ***What type of software is that?*** We need to identify the strategic open software needed in all economically strategic industries. The open source communities work in anarchy. They do not identify these needs. Most of the programmers have little involvement with the workings of most economic activities. Our explanation that there is minimal OSS development in most areas of economic activity other than computer science utilities is exactly this absence of hybrid people who are skilled programmers and competitive professionals that possess the qualities for OSS involvement. If these areas are not addressed in time, closed source becomes the de facto standard. In fact this has already happened in most areas. Most managers do not take risks in hidden agendas because they have no time or knowledge background to understand them fully. The ultimate proof of this proposition are the so-called Microsoft certified computer laboratories where even computer science schools and departments have resorted to closed source tools to teach their graduates. ***There is urgent need for the selection, evaluation and support of OSS projects in areas of strategic importance.***

There is no definite answer on all cases whether OSS or closed source software is a better option. ***There is a definite need for evaluation of strategic areas where the one or the other option is preferable and for whom, and on the areas where the two options may coexist in duality.***

According to Johnson (2001) an OSS project that contains a large number of sub-projects tends to be less complete as the number of sub-projects increase, while the closed source vendor tends to finish all sub-projects in order to remain in business. Closed source developers have no competition in their effort to influence the perceptions of managers in favor of proprietary software regarding learning difficulty, number of features, better documentation, and user friendliness on the whole. ***There is a profound need for a change in OSS development strategies if we want to see OSS competing in other areas beyond computer science utilities or telecommunications protocols.***

3.2 Institutional information fields

The questions in the strategic information fields section may require institutional or other measures. The danger exists that the control exerted today will continue in new forms and people and governments will get accustomed to it. Any institutional measure cannot succeed unless there is sufficient knowledge capability and power to support it. ***How – is it possible to be achieved in time?***

Another important issue is the loopholes of the various licenses that direct certain OSS developments. Since licenses are very important institutional frameworks that will control the evolution of software whether closed or open source in many ways, the capability of understanding and using licenses differs very much between nations. ***The questions on what software needs to be developed and under which type of license is still open for many developers and requires urgent attention***.

Labor and capital are the main inputs for the development of OSS. One could add knowledge, creativity, quality and standards that may not be direct derivatives of labor and capital. The question on where and how to complement the need for more and better input is crucial if open source software is going to substitute closed source offerings. ***We need to answer questions on the requirements for state support, institutional measures, and knowledge creation***.

3.3 Standards' information fields

We argue that OSS has not taken its right place because its development does not follow the needs of the markets. It may serve the artistic needs of some talented people, but it is a public good that could be destroyed if left with no public support. This support may be in the public formation and enforcement of standards for OSS procurement and development. It may also be in standardization of processes for OSS usage in public sector applications for the achievement of critical masses.

OSS is not available in most languages. There are not enough quality books to support learning, nor enough readers to create this need. Seminars and continuous learning activities are centered around closed source software that is easier to be reached by instructors and students. Knowledge diffusion on the operation of closed source software is rapid and acceptance as de facto standards is difficult to be reversed. ***What actions can be taken and by whom, if any?***

Building source code is not as important as sharing the design of projects. A quality design is the blueprint for project development. The quality of source code depends very much on the type of development tools used. It is very important, next to the selection of strategic OSS to be developed, on what tools should be used, and whether there are available quality designers and programmers to apply them and be able to cooperate effectively with them. According to Scacchi (2001) 'the requirements for OSS are elicited, analyzed, specified, validated, and managed through a variety of web-based descriptions, called *software informalisms* in contrast to the recommended use of formal, logic-based requirements notations ("formalisms") that are advocated in traditional approaches'. There is scarcity of programmers, especially quality people with advanced knowledge of modern approaches and design and programming tools. ***How could we expect that the void in OSS can be filled? How would they cooperate with content professionals in the various areas of expertise?***

The closed source oligopolies have managed to reside in the core of the knowledge diffusion vortex. Seminars, books, periodicals, all work towards establishing their de facto use and acceptance for different levels of user expertise. Many of these activities are sustained initially with public money. Public employees in most European countries have been trained on how to use certain proprietary software using public money. This trend has been happening for many years now and the result can be very difficult to reverse. ***What type of software should be used in those activities, in the public sector? On what level of abstraction could training be applied? There is a definite need for policy making on knowledge diffusion and training issues related to OSS acceptance and penetration***.

Reading most of the papers on OSS you get the impression that the code is of such creative value that it can be licensed and cannot be copied. This is such a fundamental illusion in most of the cases! Researchers who probably have not written even one line of code believe that code is something that has nothing to do with reality, it is something unique and magic. The fact is that very few new useful algorithms are developed today, and this usually happens in academia, where the code gets published and is mostly free! In the vast majority of the cases, programmers use existing knowledge of algorithms already tested and used elsewhere. Their skills are basically craft skills on usage of languages and tools, on selection of the appropriate algorithms and on the arrangement of modules. These skills require a level of creativity that does

not differ in principle from the level of creativity of a stone layer who selects the right stones to make houses. Even the same craftsman may use different stones if he decides to build the same house at another time period. The stones and their arrangement, cannot be licensed, so should be the case with software. What is important is the design – not the source code – and unfortunately this is normally hidden in the complexity of the code. ***We need open documented designs NOT coding.*** The most important part to which the user is hooked and gets used to, is the user interface – the way that the user interacts with the software. ***We need open documented user interfaces for standard applications NOT coding.*** This depends in many ways on the tools being used and the strategic information systems – like operating systems - on which it is based. Programmers can achieve the same user interface and functionality using different tools and algorithms – again the important part is the design and the information systems environment. For example, the interface cannot be exactly the same on MS-Windows and Linux-KDE. There are fundamental differences in the operating system, the windows interface, and the structure and the capabilities of the programming tools being used under the two environments. People with lower skills in programming, using Visual Basic to create applications in their domain on MS-Windows cannot transport easily their work to Linux. And here lies another basic problem. Simple tools, like VB used by people who are weak programmers create a lot of new useful software everyday – they use it, sell it, work with it. ***Should we seek for solutions for these people to be attracted to open source platforms and tools? How can this be achieved?***

4. Conclusions

From the above analysis we gather that the OSS development requires certain strategies to be followed in order to be able to succeed and restore some level of competition on the control of information fields that are now monopolised in certain strategic areas in economy and security. But even if OSS succeed to compete to closed software, there are great dangers for the control of information fields to remain oligopolised. It seems that there is an urgent need for a knowledgeable public sector to intervene strongly. We gave more questions than answers. We showed that there are many and difficult research questions to be answered as soon as possible. These questions are shown in ***bold italics*** in the 3rd section.

A research project is running under the direction of the author at the TEI of Larissa in order to answer some of the above questions and dilemmas.

References

Benkler Y. (2002) *Coase's Penguin, or, Linux and The Nature of the Firm*, www.benkler.org/CoasesPenguin.pdf, v.04.3, August

Blanas G. (2003) *Total Quality Management - Human Resource Management and Information Systems Networks,* Athens: Patakis, pp 107-111

Bonaccorsi A. & Rossi C. (2003) *Why Open Source software can succeed*, Research Policy Special Issue on Open Software Development.

Chiao, Benjamin Hak-Fung (2003) *An economic theory of free and open source software: A tour from lighthouse to Chinese-style socialism*, cess.nyu.edu/hfs, ver 4.83, February

Johnson J. Pappas (2002) *Open Source Software: Private provision of a public good*. Journal of Industrial Economics, 52: 197-234

Linux T. & David D. (2001), *Just for Fun—the Story of Accidental Revolutionary*, New York: HarperCollins Publishers

Raymond E. (1998) *Homesteading the Noosphere*, www.tuxedo.org/~esr/writings/

Scacchi Walt (2001) *Understanding the Requirements for Developing Open Source Software Systems* , IEE Proceedings - Software