

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <<http://www.upgrade-cepis.org/>>

UPGRADE is the anchor point for UPENET (UPGRADE European NETWORK), the network of CEPIs member societies' publications, that currently includes the following ones:

- Mondo Digitale, digital journal from the Italian CEPIs society AICA
- Novática, journal from the Spanish CEPIs society ATI
- Piroforiki, journal from the Cyprus CEPIs society CCS
- Pro Dialog, journal from the Polish CEPIs society PTI-PIPS

Publisher

UPGRADE is published on behalf of CEPIs (Council of European Professional Informatics Societies, <<http://www.cepis.org/>>) by Novática

<<http://www.ati.es/novatica/>>, journal of the Spanish CEPIs society ATI (Asociación de Técnicos de Informática <<http://www.ati.es/>>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by Novática, and in Italian (abstracts and some articles online) by the Italian CEPIs society ALSI

<<http://www.alsi.it/>> and the Italian IT portal Tecnoteca <<http://www.tecnoteca.it/>>.

UPGRADE was created in October 2000 by CEPIs and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <<http://www.svifsi.ch/>>).

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain, <rfoalvo@ati.es>
Associate Editors:

- François Louis Nicolet, Switzerland, <nicolet@acm.org>
- Roberto Carniel, Italy, <carniel@dgt.uniud.it>
- Zakaria Maamar, Arab Emirates, <Zakaria.Maamar@zu.ac.ae>
- Soraya Kouadri Mostéfaoui, Switzerland, <soraya.kouadrimostefaoui@unifr.ch>

Editorial Board

Prof. Wolfgang Stucky, CEPIs Past President
Prof. Nello Scarabottolo, CEPIs Vice President
Fernando Piera Gómez and Rafael Fernández Calvo, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

UPENET Advisory Board

Franco Filippazzi (Mondo Digitale, Italy)
Rafael Fernández Calvo (Novática, Spain)
Panicos Masouras (Piroforiki, Cyprus)
Andrzej Marciniak (Pro Dialog, Poland)

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson.

Cover page designed by Antonio Crespo Foix, © ATI 2004

Layout: Pascale Schürmann

Editorial correspondence: see "Editorial Team" above

Advertising correspondence: <novatica@ati.es>

Upgrade Newsletter available at

<<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>>

Copyright

© Novática 2004 (for the monograph and the cover page)

© CEPIs 2004 (for the sections MOSAIC and UPENET)

All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Next issue (December 2004):
"Cryptography"

(The full schedule of UPGRADE is available at our website)

- 2 Editorial
Four Years of UPGRADE

SPT, Software Process Technology

Guest Editors: Francisco Ruiz-González and Gerardo Canfora

Joint monograph with Novática*

- 3 Presentation
Software Process Technology: Improving Software Project Management and Product Quality – *Francisco Ruiz-González and Gerardo Canfora*
- 6 Software Process: Characteristics, Technology and Environments – *Francisco Ruiz-González and Gerardo Canfora*
- 11 Key Issues and New Challenges in Software Process Technology – *Jean-Claude Derniame and Flavio Oquendo*
- 17 A Taxonomy of Software Engineering Environment Services: The Upcoming ISO/IEC Standard 15940 – *Dan Hyung Lee and Juan Garbajosa-Sopeña*
- 22 Open Source and Free Software: A New Model for The Software Development Process? – *Alfonso Fuggetta*
- 27 Applying The Basic Principles of Model Engineering to The Field of Process Engineering – *Jean Bézivin and Erwan Breton*
- 34 Software Process Modelling Languages Based on UML – *Pere Botella i López, Xavier Franch-Gutiérrez, and Josep M. Ribó-Balust*
- 40 Supporting the Software Process in A Process-centred Software Engineering Environment – *Hans-Ulrich Kobialka*
- 47 Managing Distributed Projects in GENESIS – *Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, and Maria-Luisa Villani*
- 53 Software Process Measurement – *Félix García-Rubio, Francisco Ruiz-González, and Mario Piattini-Velthuis*
- 59 Process Diversity and how Practitioners Can Manage It – *Danilo Caivano and Corrado Aaron Visaggio*

MOSAIC

- 67 Data Architecture
A Disquisition on The Performance Behaviour of Binary Search Tree Data Structures – *Dominique A. Heger*
- 75 News & Events: News from CEPIs and EUCIP; SCI 2005 (Call for Papers)

UPENET (UPGRADE European NETWORK)

- 77 From **Novática** (Spain):
IT and Disabilities
Braille and The Pleasure of Reading: We Blind People Want to Continue Reading with Our Fingers – *Carmen Bonet-Borrás*
- 84 From **Piroforiki** (Cyprus):
Information Technology in Today's Organizations
Is the IT Productivity Paradox Resolved? – *Kyriakos E. Georgiou*

* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by **Novática**, journal of the Spanish CEPIs society ATI (*Asociación de Técnicos de Informática*) at <<http://www.ati.es/novatica/>>, and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIs society ALSI (*Associazione nazionale Laureati in Scienze dell'informazione e Informatica*) and the Italian IT portal Tecnoteca at <<http://www.tecnoteca.it/>>.

Open Source and Free Software: A New Model for The Software Development Process?

Alfonso Fuggetta

© Alfonso Fuggetta, 2004

Open source software is having a significant impact on the ICT market. Unfortunately, many claims associated with open source software are either misleading or simply false. This makes it difficult to really appreciate and exploit the potential of open source software. This paper proposes some considerations and reflections that aim at critically revising some assumptions about open source software. The ultimate goal is not to deny the role of open source software; rather, the paper aims at identifying the really novel and original characteristics of open source software with respect to more traditional approaches.

Keywords: Business Model for Software, Open Source, Software Development Processes.

1 Introduction

Open Source Software (OSS) is certainly one of the most important and relevant phenomena of this decade. The success of Linux and Apache is pushing practitioners and researchers to reconsider some of the classic assumptions about software development. Even software giants such as Microsoft [5], Sun, and IBM have been changing or adapting their strategy to take into account this unconventional approach.

1.1 The Reasons of A Success

The supporters of OSS claim that open source is able to address and solve a number of issues. In particular, it is supposed to be a more effective and efficient way of developing high quality software. Moreover, it makes it possible to disseminate innovation and technology more easily and effectively. It is also an extremely attractive approach to lower the costs of IT investments. In Europe, many observers consider open source an effective strategy to counterbalance the American dominance in software technology and, consequently, to revamp the European software and computer industry, which was dramatically weakened in the 90' by the collapse of BISON (i.e., Bull, ICL, Siemens, Olivetti, and Nixdorf).

Beside these technical and economic issues, open source software is also – and for many supporters, primarily – an ethical and cultural issue. In particular, free software advocates, such as Richard Stallman, claim that software must be ‘free’ (as in “free speech”) because proprietary/close software violates five basic users’ rights:

- The freedom to use the software.
- The freedom to study the source code.
- The freedom to modify it.
- The freedom to copy it.
- The freedom to redistribute it.

Indeed, free software and open source are often considered equivalent concepts. However, even if most practical consequences are basically the same, the two approaches have different backgrounds and motivations. For the sake of simplicity, in the remainder of the paper I will use the term OSS to identify the whole world of open/free software. When needed or convenient, the distinction between these two notions will be made explicit.

1.2 Some Basic Concepts

Software products can be classified in two main categories: packages and custom (or bespoke) software.¹

- *Packages* are software products developed to address general needs for a number of different users. Moreover, they are distributed through licenses that define customers’ rights and obligations. The license defines a package as proprietary or open source (or some intermediate variant). Typical examples are Windows, Linux, Office, and StarOffice.
- *Custom software* is developed for specific needs of a customer, who pays the cost of software development. Usu-

Alfonso Fuggetta is a Full Professor at *Politecnico di Milano* (Italy), *Dipartimento di Elettronica e Informazione*. He is also Director of CEFRIEL, the ICT “Centre of Excellence for Research, Innovation, Education, and Industrial Labs” partnership established in 1988 by *Politecnico di Milano*, the Regional Council of Lombardy, and the most important ICT companies operating in Italy. Fuggetta is a Faculty Associate of the Institute for Software Research of the University of California, Irvine, since 2002. He is also Chairman of the Scientific Committee for New Economy, Innovation, and Scientific Research of Lombardy, and of several committees of the Italian Government including the Government Committee on Open Source Software in the Public Administration. <alfonso.fuggetta@cefriel.it>

1. Of course, there are also intermediate situations where a software product is a combination of packages and custom software.

ally, this is accomplished by signing a service contract between the customer and a software house or system integrator. Such contract can and should always guarantee to the customer the full ownership of the software (possibly non-exclusively), as it pays the entire cost of development. In particular, full ownership means unrestricted access to the source code, i.e., even more than what is granted by open source licenses such as the GPL: a customer might even decide to put under public domain the custom software it purchased! A typical example of custom software is the software used to manage a specific procedure in a Public Administration.

From the above observations it can be argued *that the notion of open source apply primarily to packages*, as discussed in detail in the remainder of the paper.

1.3 Open Source, Open Standards, and Open Format

OSS is often associated with open standards and open format. Indeed, these concepts are orthogonal.

An open standard is a *set of requirements* that are not controlled by a single company. An open standard is useful to guarantee that any product adhering to the standard provides compatible and coherent features and operations. This is crucial to provide interoperability and the possibility to replace a product with a compatible one.

An open standard may be adopted by both OSS and proprietary software. Actually, Internet defines a set of “open standard” protocols such TCP (Transmission Control Protocol), which are available both in proprietary and open source operating systems (e.g., Linux and Windows).

Certainly, proprietary software must interact with the “rest of world” using open standards and open formats. Again, this is not the same as requiring that software must be open source in order to be open standard.

1.4 Misleading Beliefs, Unjustified Expectations

The notion of “open source” is associated with three different concepts:

1. A set of *licenses*, such as the well-known General Public License (GPL).
2. A *range of software development practices*, which exploit the notion of open source to facilitate cooperation, quality assurance, and innovation.
3. *Products* that are developed and distributed using an open source license and open source development practices.

In turn, these three factors are supposed to induce a whole new way of conceiving and running the software business. In practice, software developers and producers are supposed to make money from selling services (distributions, training, documentation, consulting, ...) rather than from license fees.

In general, there is a growing trend towards considering OSS as ‘the’ strategy for the future of the software industry. Unfortunately, most of the beliefs and claims about OSS are fascinating on the surface, but false or misleading when studied in more detail: very often, they are either unjustified or simply *independent of the nature of software* (i.e., they equally apply to open and proprietary software). The effect of this situation is

the growth, especially in Europe, of unjustified expectations that might turn out to overlook the real problems, ignore key issues, and devaluate the real impact and significance of OSS itself [4].

1.5 Goal of This Paper

This paper presents some considerations about the many claims and expectations associated with OSS. The goal is not to deny the role and opportunities associated with OSS. Rather, it aims at identifying the real and novel characteristics of OSS in order to effectively exploit them. For this reasons, the following sections will discuss some main questions related to technical, economic, and social aspects of OSS. The last section will provide some concluding remarks and suggestions for future work.

2 Is OSS A New Development Process?

OSS supporters claim that the openness of software and the cooperation styles supposedly used in many OS projects define a new and extremely effective software development process. Such process is based on the notions of decentralized development, distributed testing, and effective exploitation of distributed expertise and knowledge. The originator of this position is Raymond, whose seminal work on “The cathedral and the bazaar” has been followed by many other studies and contributions trying to define the “open source development process”.

Indeed, *there is no such process*. There are two main motivations that support this claim [4]:

- Most software development initiatives are carried out by a limited (often just one) number of developers. This observation has been corroborated by several studies of open source repositories such as Sourceforge.net [6]. Large open source projects such as Linux and Apache do have a very well structured and organized process, which resembles those of other proprietary products [8].
- The features and characteristics of the “open source development process” can be applied and observed also in proprietary software. For instance, Microsoft exploits daily builds and feature orientation as a way to make software development flexible [3]. Approaches such the Spiral model and Extreme Programming (XP) have stressed the notion of incremental and evolutionary development, which is supposed to be a main characteristic of OSS. Actually, those approaches can be equally applied to proprietary and open source software.

In general, from a technical viewpoint it is hard to consider open source as a totally new development paradigm. Certainly, the vision behind OSS represents a strong motivation factor that has been able to involve and influence a very large number of developers and users. This is one of the real novel aspects of OSS: its fascinating ability to motivate people.

Actually, even if not directly related to the hypothetical open source development process, some ideas and suggestions related to open source do have a role to play in specific situations. The most relevant one is the sharing and joint development of custom software in a community, e.g., the Public Administra-

tion sector. There are hundreds (even thousands) different Public Administrations that share the same problems and requirements. For example, town councils have the same needs and, therefore, can share the same software. This can be achieved by either using the same package or reusing the same custom software. Custom software, which should be in the full ownership of the purchasing administration, can and should be shared with other administrations using some form of open source licensing (probably, something similar to community sourcing). This is another aspects of OSS that should be much more deeply considered, even if it is not directly related to classical packages such as Windows and Linux.

3 Is OSS The Only Way to Protect Customers' Rights?

OSS advocates claim that open source is the only way to protect customers' right. This is at least misleading. As for custom software, the customer should always own it and, therefore, *the problem does not exist by definition*. Indeed, the problem does exist for packages.

A first problem is to access the source code in order to check what the software really does. This is useful to guarantee that a software package does not accomplish undesired or illegal operations or, also, to support testing of custom software developed using package features (e.g., a custom software using Oracle DB, Windows, and .Net). In order to solve this problem, *it is sufficient to make the source code of a package 'accessible' to the user* (i.e., the user can see and modify it, but it cannot redistribute or copy it illegally). This requirement is much weaker than making it open source. Still, it is sufficient to solve the problem and is probably acceptable to most producers of proprietary software.

A second problem concern the inability of proprietary packages' users to change the company in charge of maintaining the software, or to take control of the software whenever the developer is unable or unwilling to continue maintaining the package. This is a critical problem that can be at least partially solved by introducing specific norms to protect customers. For instance, if software is sold with a license that does not have time limitation, the producer should not be allowed to drop support and maintenance services once a new version of the package is released, unless the source code of the older version is made open. Similarly, a company that is patently unable to provide support and maintenance (for instance because it has deep financial problems) should be forced to release the source code.

4 Is OSS An Effective Way to Disseminate Knowledge?

OSS is supposed to be the means to solve a number of problems related to dissemination of knowledge, international cooperation, and the digital divide. Again, this is misleading.

First, knowledge about software cannot be distributed by simply looking at the source code. As a provoking statement, I argue that *the larger (and significant) is the code, the smaller is the amount of information that can be disseminated by simply looking at that code*. Indeed, software engineering research and practice have demonstrated that developers need high-level requirement and architectural documents that are able to

describe a software system. In the past years, an entirely new discipline has been started (reverse engineering), whose goal is to extract meaningful information from source code.

Second, even assuming that looking at the source code does transfer knowledge, it would be sufficient to make the code 'accessible' (as mentioned above). It is not necessary to make it open source, i.e., to grant the customer also the right to copy and redistribute the software.

5 Is OSS Cheaper?

Stallman strongly argue that OSS does not mean 'free' as in "free beer". OSS can be commercially distributed, as companies such as Red Hat do (see later on). Also, software needs to be maintained and supported. OSS advocates say that open source developers make money by selling services. Therefore, *OSS does cost*, as any other software.

As a consequence, any conclusion about OSS being cheaper than proprietary software should be based on a detailed analysis of all the costs related to owning and operating a specific software solution. This is called *Total Cost of Ownership* (TCO). Someone argues that TCO has been invented to support the claims of proprietary software producers. Indeed, this notion is widely used to decide any sort of purchase (e.g., a car or computer hardware) and therefore it is difficult to understand why it should not be valid and applicable also in the case of software.

6 Is OSS An Effective Business Model?

OSS is considered a viable and even unique way to build a convincing and enduring business model. People will be less inclined to pay for software licenses and will rather prefer to pay for specific services associated with using the software. This appears as an additional motivation for opening the code of software packages.

The argument is once again ill conceived. First, software costs and it is not clear if companies can survive and make profits by simply selling services. In his latest book [2], Michael Cusumano has accomplished an analysis of existing companies, their profits and strategies, and their success stories. He argues that in the future most software companies (including Microsoft, IBM, and other producers of proprietary software) will increasingly rely on a model where revenues will be *a mix of product licenses and services*.

It is useful to assess Cusumano's comments by considering in more detail the real business models where OSS is supposed to play a role. There are five basic models to consider:

1. *Development and distribution of 'pure' open source packages (and related services)*. A typical example is Red Hat, who makes money distributing and supporting Linux.
2. *Development and distribution of open source packages (and related services) developed for open source and proprietary platforms*. This is the case of companies such as Zope, which has created an open source development platform for both Windows and Linux.
3. *Development of proprietary packages (and related services) for open source and proprietary platforms*. This is the case of StarOffice (not OpenOffice!), developed by Sun for

Linux and Windows. Similarly, IBM sells a number of proprietary packages for the Linux platform.

4. *Development of open source packages (and related services) that make a product line more attractive.* This is once again the strategy of IBM and Sun, which promote the diffusion of Linux and other open source packages (in particular, Apache) on their hardware products to increase their competitiveness with respect to the Wintel platform.
5. *Development of custom software (and related services) using open source platforms.* This is the case of many software houses and system integrators who base their developing activities on Linux, Zope, Tomcat, and JBoss rather than on Windows (and related technologies).

There are also companies that are exploiting hybrid approaches. For instance, MySQL uses a dual-license approach that integrates open source and proprietary concepts.

In general, the above models appear to cover all the possible basic ‘bricks’ of a hypothetical business model based on open source. However, *only the first two strategies are really and truly exploiting open source.* The third and fourth alternatives are commercial strategies of companies who want to promote their proprietary software and proprietary hardware platforms. As for the fifth one, indeed the business model of a system integrator does not change that much. A system integrator (i.e. a developer of custom software) has always been used to consider alternative platforms. In some cases, this is caused by specific requirements of the customer. In other situations, where such requirements are missing or weaker, a system integrator is used to look for the most convenient platform for its development. Certainly, open source packages such as Linux, Apache, and Tomcat make it possible to build software systems with lower costs for licenses. In general, a system integrator will consider the TCO of each different alternative and select the most convenient one. This already happened in the past. For instance, the change from large mainframes to minicomputers and, later on, to PC networks was motivated by the relatively much lower entrance costs of the newer technology. Summing up, for system integrators there is nothing really and dramatically new.

In conclusion, will these models be the future of the software industry? Hard to say. First, only two of them are really a significant departure from traditional models. Second, the number of companies who are really succeeding using those approaches is still relatively small. Even Red Hat, perhaps the most successful and large open source player, is still in search of the right strategy to be market profitable.

7 Is OSS A Strategy to Strengthen The Software Industry?

A final important claim of many OSS supporters is that open source can be an important means to strengthen the software industry, especially outside the US. Even if I totally share the concerns of those that correctly require a stronger non-US software industry, I am very skeptical that open source alone might have a significant impact. Europe, for instance, misses a clear

and committed industrial strategy for software. Mobile phone producers are a typical example. Microsoft (and PalmOS) is promoting a standard platform for mobile phones. This standard can repeat the success of the personal computer: there are many hardware producers, but one standard software. European producers, conversely, have multiple and incompatible platforms that make it difficult for developers to invest in developing applications. If an application is developed for Windows Smartphone or Pocket PC, a developer is sure that it can run on any device compatible with those systems. Conversely, can an application developed for the Symbian-based Ericsson P900 run also on the Symbian-based Nokia Communicator? The answer is no. So, an application developer will hardly consider Symbian an attractive platform, while he/she will be certainly inclined to invest in Pocket PC or PalmOS development. Eventually, the end user will make his/her choice on the basis of the applications available on each platform.

This is the real problem. Europe misses an industrial strategy. “Supporting open source” is not a strategy. At best, it is a request to fund open source projects with public money. This is not what we need. We should consider lessons such as Airbus. By creating an aircraft company able to compete with Boeing, Europe has become the main player in the market. This was accomplished using public money, but there was a clear industrial strategy aiming at creating innovative products. Software is not innovative just because it is open source. *Europeans are starting from the tail (a dissemination strategy such as open source) rather than from the head (the identification of the innovative products to develop).*

8 A General Remark

Indeed, the more I consider the literature on OSS, the more I am convinced that in many situations the really important thing for many customers is that open source is ‘free’ as in “free beer”. This is a huge risk. *Software is not ‘free’*; it does cost. The issue is then who is going to pay this cost. Imagining that software can be created at no or low cost is a major threat to innovation. Indeed, how many open source projects are really innovative? Why most of them are just replicas of existing software? Customers should not be led to believe that software could be acquired “for free”.

9 Conclusions

This paper has summarized some of the main issues and discussions about OSS. The paper does not assume an a-priori position in favour or against open source. Rather, it tries to critically discuss and assess many claims made on OSS. Unfortunately, many of them are ill conceived, misleading, or patently false. This is dangerous to OSS itself in the first place.

Certainly, it is important to deepen the discussion on the real novel aspects of open source. Even more important, we need to find effective and convincing solutions to the problems that our society and the software industry have to address in the next years. This paper wants to be a contribution in this direction.

References

- [1] R. Conradi, A. Fuggetta. Improving software process improvement. *IEEE Software*, July 2002.
- [2] M. Cusumano. *The Business of Software*. Free Press, 2004.
- [3] M. A. Cusumano and R.W. Selby. *Microsoft secrets*. The Free Press, 1995.
- [4] A. Fuggetta. Open source software: an evaluation. *Journal of Systems and Software*, April 2003.
- [5] J. Greene. Microsoft's Midlife Crisis. *Business Week*, April 19, 2004.
- [6] K. Healy, A. Schussman. The ecology of open source software development. Technical report, University of Arizona. Available at <http://opensource.mit.edu/online_papers.php>.
- [7] D. G. Messerschmitt, C. Szyperski. *Software Ecosystem*. The MIT Press, 2003.
- [8] A. Mockus, R. T. Fielding, J. Herbsleb. Two case studies of open source development: Apache and Mozilla. *ACM TOSEM*, Vol. 11, Issue 3, July 2002.