The Power of Open Source ERP

CLAUDIA CARSTEA
Department of Mathematics, Informatics and Socio-Human Sciences
“George Baritiu” University
Brasov, No. 6, Lunii Street, Brasov
ROMANIA
carstea.claudia@yahoo.com

Abstract: The paper presents a sustainable business model for open source software tools, managing and disseminating documents in heterogeneous software (source code files, database objects, graphical objects, text files) for concurrent economic applications. The paper motivates the utilization of open source models for the maintenance and adaptation of the application or generic software. It describes the representation of the software Internet computing, the architecture of the open source-based XML repository manager and the most important issues for its implementation. The system uses encryption and other security mechanisms to ensure that only authorized users can access a concurrent economic application and the date can not be intercepted. When I talk with with other people about Free-Libre / Open Source Software (FLOSS), I still hear a lot of people mistakenly use the term “commercial software” as if it had the opposite meaning of FLOSS (aka open source software, Free-Libre Software, or OSS/FS). That’s in spite of the rise in commercial development and support for FLOSS, most FLOSS projects’ goal to incorporate improvements (which are actually a form of financial gain), official definitions of “commercial item” that *include* FLOSS, and FLOSS licenses and projects that clearly approve of commercial support. Terms like “proprietary software” or “closed source” are plausible antonyms of FLOSS, but “commercial” is absurd as an antonym.

Key words: Open Source, ERP, Business Model, FLOSS

1 Introduction

The OSI are the stewards of the Open Source Definition (OSD) and the community-recognized 2 body for reviewing and approving licenses as OSD-conformant.The OSI is actively involved in Open Source community-building, education, and public advocacy to promote awareness and the importance of non-proprietar software. OSI Board members frequently travel the world to attend Open Source conferences and events, meet with open source developers and users, and to discuss with executives from the public and private sectors about how Open Source technologies, licenses, and models of development can provide economic and strategic advantages. [2,14]

It's important to not confuse "FLOSS" and "non-commercial". To see why, let’s first define our key terms, “FLOSS” and “commercial”:

1. FLOSS can be briefly defined as software whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program (without having to pay royalties to previous developers). (This summarizes the Free Software Definition; the Open Source Definition is longer, but for purposes of this essay has the same basic result).

There are many places that define the term "commercial", so let's look at a dictionary and a government's official definition: The New York Times’ Everyday dictionary, 1982 says that Commercial means either (a) “oriented to profit-making”, or more generally (b) “of, pertaining to, or suitable for commerce”, where commerce means “intercourse, dealings, the buying and selling of commodities, or trade” So we’re talking about something (a) oriented toward profit, or at least (b) something pertaining to public trade or dealings. Even with just the first meaning, there are many commercial FLOSS programs; when we include the second meaning (which some people forget), nearly all FLOSS programs are commercial. [3,11,15]

U.S. law governing federal procurement (specifically U.S. Code Title 41, Chapter 7, Section 403) defines "commercial item" as including "Any item, other than real property, that is of a type customarily used by the general public or by non-governmental entities for purposes other than governmental purposes [i.e., it has some non-government use], and (i) Has been sold, leased, or licensed to the general public; or (ii) Has been offered for sale, lease, or license to the general public ..." Again, by this definition nearly all FLOSS is a commercial item. For most of this essay I'll use the dictionary definition, but I'll focus on this U.S. government definition in this essay's section discussing official definitions. [8]

As FLOSS has become more prominent in the computer industry, many speakers have tried to differentiate FLOSS from software released under other license terms. That's fine, but some people have unfortunately been trying to use the term "commercial" as something distinct from FLOSS.

This confusion that FLOSS and commercial software are opposites is a dreadful mistake. Speakers who differentiate between FLOSS and commercial products, as if they were opposites, are simply unable to understand what is happening in the software industry.

And if you cannot understand something, you cannot make good decisions or even create good advice about it. Some concepts aren't important, but software controls every important device on the planet. If you wish to understand the 21st century (and beyond), you need to understand the basics of what controls software... because software controls everything else. It is no longer acceptable to make such a terrible mistake.

So, with that in mind, let's examine why treating "FLOSS" and "commercial" as opposites is fundamentally wrong. [6]

2 Official definitions of "commercial item" include FLOSS

Many official definitions of terms like "commercial item" include nearly all FLOSS programs. Let's look at various official U.S. definitions and policies, to show that this is clearly true in the U.S.

The U.S. government's own official definition of "commercial item" makes it clear that nearly all FLOSS programs are considered commercial items. The U.S. law governing federal procurement (specifically U.S. Code Title 41, Chapter 7, Section 403) is reflected in the Federal Acquisition Regulation (FAR) System widely used for acquisition (some organizations use more specific regulations based on the FAR, such as the Department of Defense' DFARS). U.S. law and the FAR have a very detailed definition of the term "Commercial item" which clearly includes essentially all FLOSS; as noted below, specific organizations like the U.S. Navy have specifically stated this.

Before we look at that definition, let's note that this definition is really important (it's not just in some dusty, unused part of U.S. law or policy). The FAR specifically requires in part 12 that U.S. government agencies shall, by policy, try to use commercial items or nondevelopmental items wherever they can. More specifically, part 12 requires agencies to "*(a) Conduct market research to determine whether commercial items or nondevelopmental items are available that could meet the agency's requirements; (b) Acquire commercial items or nondevelopmental items when they are available to meet the needs of the agency; and (c) Require prime contractors and subcontractors at all tiers to incorporate, to the maximum extent practicable, commercial items or nondevelopmental items as components of items supplied to the agency.*"

What's a nondevelopmental item? The FAR defines it as *(1) Any previously developed item of supply used exclusively for governmental purposes by a Federal agency, a State or local government, or a foreign government with which the United States has a mutual defense cooperation agreement; (2) Any item described in paragraph (1) of this definition that requires only minor modification or modifications of a type customarily available in the commercial marketplace in order to meet the requirements of the procuring department or agency; or (3) Any item of supply being produced that does not meet the requirements of paragraphs (1) or (2) solely because the item is not yet in use.*

Since governments need a lot of software *not* developed exclusively for governmental use, the policy in the FAR turns out to be a rather strong requirement to use commercial items wherever possible. [4,5]

So let's walk through the U.S. government definition of "commercial item" as given by FAR part 2, which clearly shows that ***typical FLOSS programs are commercial items*** for purposes of the U.S. government. Here's what they mean by the term "commercial item":

1. *Any item, other than real property, that is of a type customarily used by the general public or by non-governmental entities for purposes other than governmental purposes, and (i) Has been sold, leased, or licensed to the general public; or (ii) Has been offered for sale, lease, or license to the general public...* -- I should note that later on in part 12 the phrase "purposes other than governmental purposes" is clarified as meaning purposes "that are not unique to a government." Nearly all FLOSS is used by the general

public or non-governmental entities for purposes other than exclusively governmental purposes. Even FLOSS programs that implement functions often performed by governments are often not exclusive to this purpose, for example, there are FLOSS integrated library systems, but many other non-government organizations (for-profit or not, including some larger universities and companies) that *also* have integrated library systems.

2. More importantly, note that the software only has to be licensed or offered for license to the general public. FLOSS is typically licensed through the general public, normally through one of a few well-known licenses such as the GNU General Public License (GPL), the Lessor GPL (LGPL), MIT, or BSD-new licenses. It doesn't even need to be sold or leased to be a commercial product; licensing *itself* to the public makes it commercial.

By itself, this clause makes nearly all FLOSS programs commercial items (from the U.S. government's point of view), because nearly all FLOSS programs are licensed to the general public and have at least some use not strictly limited to a government. But there are even *additional* ways that a program can be considered a commercial item!

3.*Any item that evolved from an item described in paragraph (1) of this definition through advances in technology or performance and that is not yet available in the commercial marketplace, but will be available in the commercial marketplace in time to satisfy the delivery requirements under a Government solicitation;* -- So even if the FLOSS isn't released to the public yet, it may still count, as long as it *will* be released in time. This enables a FLOSS project that is in embryo to still compete. After all, it may not be completely ready yet, but as long as it will be that's fine. Note that this can be especially helpful for FLOSS "bounty systems" (also called sponsor systems or pledge systems), where people commit money in exchange for having someone create a FLOSS result. Thus, if funding is already committed to create a FLOSS project that will be released to the public in time, it can still be considered commercial. In a somewhat similar manner, a program that is written but not released to the public (and thus not yet a commercial item at all under the first definition) could be ransomed for release as a FLOSS program. [9]

4. Any item that would satisfy a criterion expressed in paragraphs (1) or (2) of this definition, but for: (i) Modifications of a type customarily available in the commercial marketplace; or (ii) Minor modifications of a type not customarily available in the commercial marketplace made to meet Federal Government requirements. Minor modifications means modifications that do not significantly alter the nongovernmental function or essential physical characteristics of an item or component, or change the purpose of a process. Factors to be considered in determining whether a modification is minor include the value and size of the modification and the comparative value and size of the final product. Dollar values and percentages may be used as guideposts, but are not conclusive evidence that a modification is minor;. -- Thus, a government acquisition program can obtain a FLOSS program, pay for minor modifications to meet its needs, and still consider it a commercial item. [7]

5. Any combination of items meeting the requirements of paragraphs (1), (*2), (3), or (5) of this definition that are of a type customarily combined and sold in combination to the general public;* -- So combinations are okay.

6. Installation services, maintenance services, repair services, training services, and other services if (i) Such services are procured for support of an item referred to in paragraph (1), (2), (3), or (4) of this definition, regardless of whether such services are provided by the same source or at the same time as the item; and (ii) The source of such services provides similar services contemporaneously to the general public under terms and conditions similar to those offered to the Federal Government; -- So commercial companies that sell support for FLOSS programs are quite within their rights for being a commercial item.

3 Alternatives and problem formulation

The most common antonym for FLOSS is "proprietary software", though there are other terms like "closed source", "non-Free", and "non-FLOSS". Most terms have minor problems of one kind or another:

- "Proprietary software" usually works, but it is sometimes also used to describe software that (1) uses its own formats or protocols instead of open standards, or (2) is *never* brought to market directly (such software may be included as a custom system sub-component specifically to prevent acquirers from switching to another supplier). Still, most of the time, when people use this term, they mean the opposite of FLOSS.
- "Closed source" has a different

problem - some people mean by this phrase that the source code is not available. Yet there are some programs whose source code is available but are not FLOSS programs, making the term possibly very confusing. Again, though, most people simply mean non-FLOSS programs by this term, so it is fairly clear.

- "Non-Free" has the problem that it just means "costs money" to most people. The phrase "Free software" has the same type of problem, by the way, which is why I prefer the terms FLOSS, "Free-libre software", or "open source software" instead of "Free software".
- "Non-FLOSS" is the most unambiguous, but few use the term.

I tend to use "proprietary software" as the antonym, simply because it seems to be the most widely used and thus better understood. *Any* of these terms is better as an antonym compared to "non-commercial". It's time to end the nonsense. Terms like "proprietary software" or "closed source" are plausible antonyms of FLOSS, but "commercial" is absurd as an antonym. The term "commercial" cannot be justified due to: (1) the rise in commercial development and support for FLOSS, (2) most FLOSS projects' goal to incorporate improvements, which are actually a form of financial gain, (3) official definitions of "commercial item" (at least the U.S. government definition) that *include* FLOSS, and (4) FLOSS licenses and projects that clearly approve of commercial support. [13]

Trying to use the word "commercial" as an antonym for FLOSS is becoming more absurd every day. Even if you use the narrower definition for commercial that means "for profit", there are too many for-profit FLOSS projects for this use to make any sense. When you consider the full set of meanings for "commercial", including the one involving public trade, nearly *all* FLOSS projects are commercial. In short, there are two kinds of commercial software: proprietary and FLOSS. [10]

This has real-world implications. For example, many organizations prefer commercial software instead of home-grown software (for which they have to pay *all* of the maintenance costs). This implies that such organizations must search for and evaluate FLOSS projects when they search for commercial software, and if there isn't an appropriate product available, they need to consider starting such a FLOSS project as one of their possible implementation approaches (examples of the latter are the devIS EZRO and the Georgia Public Library Service's Evergreen systems). If acquirers ignore FLOSS options, they are ignoring an important and growing part of the commercial sector. [11]

A speaker who uses the term "commercial" as an antonym for FLOSS is probably someone who doesn't understand FLOSS yet. And someone who doesn't understand the fundamentals of how software is governed will be constantly confused about what controls every device on the planet. Be wary of people who have such a basic lack of understanding; they are far less likely to give good software advice or to make good software-related decisions.

4 What Are the Benefits of Open Source ERP?

4.1 Open source vs. proprietary software

Most of the software businesses use today is proprietary software, meaning it is the legal property of a vendor that makes the software available to others via a proprietary licensing agreement. In most cases, the vendor limits access to the underlying source code, while providing customers with the right to use the binary software "as is."If users need an improvement to the proprietary software (such as an additional feature or a bug fix), the only source for the improvement is the original vendor. Open source works differently. Rather than restricting access to the underlying source code, open source software includes both binary and source code, along with a license, to make improvements to the base software product. Typically, customers rely on their open source vendors for improvements just as they do with proprietary vendors. The critical difference is that the widely distributed source code gives customers additional support choices.

Open source products are just as feature-rich, innovative and dependable as their proprietary cousins. However, they also provide other advantages:

- Increased adaptability and visibility: Few end users change the underlying code of an open source application. But when the need arises, open source provides access to the code to make changes to suit each distributor's unique business needs. Open source customers enjoy a refreshing level of transparency from their vendors around activities such as bug reporting and fixing and roadmap planning.

- Easier integration with current systems: ERP solutions touch every aspect of a company, from warehousing to accounting. As such, a company's ERP solution should easily integrate with existing IT infrastructure components, such as application servers, directory services and storage arrays. Open source solutions are compatible via standards-based interfaces with multiple technologies, including support for lowest-cost commodity operating systems, databases, utilities and hardware.

Enterprise Resource Planning (ERP) software helps distribution businesses improve margins and competitiveness by automating processes and improving visibility into business operations. The ERP market offers a wealth of solutions for distributors to choose from, both open source and proprietary. Functionality is the most important driver when selecting from these options, but companies should also evaluate total cost of ownership, availability of support, ease of use, ease of integration and how well the software will grow with an organization.

4.2 2009 the year for open-source ERP?

While open source has made its mark in just about every product segment within enterprise software, ERP (enterprise resource planning) has remained firmly proprietary. As CIO.com's Thomas Wailgum suggests, however, the time may be ripe for change.
The reason? Years of empty promises and overloaded invoices from the incumbent ERP vendors may finally ring hollow in a global recession:
Does a massive, 18-month, multimillion-dollar ERP rollout, with the odds of implementation and user acceptance stacked against you and 22 percent annual maintenance costs to boot, seem appropriate now? [12, 14]

ERP industry guru Vinnie Mirchandani likes to say that there are too many "empty calories" in ERP spend, especially in SAP and Oracle maintenance fees. Now is clearly not the time to be ordering up large portions of highly caloric ERP software rollouts. Just as the economy is proving to be Google's toughest competitor and the biggest reason to innovate, so, too, may the economy be chief information officers' biggest reason to get out of the "no one ever got fired for spending way too much on bloated ERP" mindset and shift spending to software as a service and open-source ERP.

Openbravo, Compiere, and other open-source solutions have been around for several years, and they are surprisingly robust and feature-rich. 2009, with all its financial challenges, may well be the opportune time for CIOs to kick the tires on these alternative solutions.

4.3 Open Source Matters

Open Source is one of those buzzwords that probably does not matter much to most people, but it is our bread and butter. Here at DBS, we use Open Source products to run our servers. We use it to build and manage websites. We use it for hosting, marketing campaigns and internal business applications. It isn't just us, though. Much of the Internet is built with Open Source products. Google, for instance, is built on an Open Source operating system called Linux and it seems it has worked out OK for those guys (understatement of the year). And an Open Source web server called Apache has been the #1 web server on the planet since 1996 (based on Netcraft web surveys) despite Microsoft's effort to flex its muscles in the server realm. Firefox, the web browser, is something that possibly resonates with more people and it is also Open Source.[8]
Okay…so Open Source matters to us, but why should it matter to our clients? Well, because there are a number of benefits to Open Source development that trickle down and benefit our clients, and ultimately, our client's users. Using Open Source tools will allow us to produce quality products that:

· Often save us from having to re-invent the wheel, time and time again. Shorter Development Time matters.

· Allow for fast adoption of new web-based technologies. Because open source projects tend to be reflection of the latest web technologies, they tend to foster those technologies. Technology matters.

· Offer better support for web standards. By their nature, Open Source products are the opposite of vendors that try to circumvent standards with proprietary protocols. Standards matter too.

· Cost less. As crass as it may sound, saving money always matters.

These are meat and potato reasons our customers receive tangible benefits from our participation in the Open Source software community. They are more likely to get a quality product, faster, and cheaper than if the same project were built solely with proprietary, closed source products.

There is one last, philosophical reason why Open Source matters: because it is open and because it is “free”. We, in the United States, live in a free and open society. At least, we aspire to those lofty ideals. We preach and foster their acceptance around the globe and find it hard to apply any kind of negative connotation. We have free speech, a free press, freedom to worship as we please (or not), freedom of choice, free assembly and so on. Free is good, no?

Yes, free access to the code we use to build servers, sites and applications is a good thing. It allows us to share and improve a global codebase that is open to all of us, not just the few. It is not held in proprietary hands and licensed to us temporarily as the masters of that codebase see fit. It belongs to us, we, the people. Yes, freedom matters too.

5 Conclusion

The licenses, and the discussions surrounding both licenses and projects, also make clear that developers in FLOSS projects typically have no problems with commercial development and support, even if you use the narrower definition of “for-profit” for “commercial”. Indeed, many projects are established by commercial organizations as a kind of consortia (e.g., X Windows and Apache), while others are established by single commercial organizations (e.g., MySQL and Qt).

Official commentaries on the two common formal definitions of FLOSS both specifically state that FLOSS is not “non-commercial”:

1. The Free Software Definition says “Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important.”
2. The Open Source Definition says in point 6 “The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business... Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.”

The FSF goes further; in their article Selling Free Software, they say: "we encourage people who redistribute free software [FLOSS] to charge as much as they wish or can. If this seems surprising to you, please read on... When we speak of "free software", we're talking about freedom, not price... Since free software is not a matter of price, a low price isn't more free, or closer to free. So if you are redistributing copies of free software, you might as well charge a substantial fee and make some money. Redistributing free software is a good and legitimate activity; if you do it, you might as well make a profit from it."

The most popular FLOSS license in the world is the GNU General Public License (GPL), and in version 2 of the GPL it includes one method for copying and distributing the program (”method 3c”) which can *only* be used for noncommercial distribution (presumably they mean the for-profit definition). Since other methods are not so encumbered, the clear implication is that commercial (for-profit) distribution methods *are* permitted, as long as they obey the license. Which brings me to an interesting point that can cause confusion.

While the vast majority of FLOSS developers are happy with for-profit commercial development and support of FLOSS, they do *not* support companies that violate the program license or try to find and exploit legal loopholes in it. In some cases, organizations that violate FLOSS licenses have had to be brought into court so that they would obey the license. Many FLOSS developers become quite upset with companies that fail to obey the FLOSS software license, and external observers sometimes misunderstand this anger as a general opposition to commercial use by FLOSS developers. But this would be a mistake; such anger is directed at violators, not to commercial users in general. Such vehemence is typically true of proprietary software developers, too; proprietary software vendors are quite unhappy with those who do not obey the proprietary license, and are usually even more eager to bring a violator into court. In addition, proprietary licenses permit fewer actions than FLOSS licenses, so there are many more ways a commercial organization could accidentally violate a proprietary license (and risk being brought into court) compared to a FLOSS license. For example, making additional copies for multiple users is encouraged by all FLOSS licenses, but forbidden by most proprietary licenses (unless additional fees are paid). All commercial software developers, both proprietary and FLOSS, expect their users to obey the license provided or negotiate something else, as is required by law.

Some representatives of proprietary software companies leave the mistaken impression that all FLOSS programs are non-commercial. For example, Microsoft gave presentations in 2002 claiming that “research results placed under the GPL are precluded from commercial use”,

> “Government-funded research released under the GPL can never be commercialized” and “non-commercial software will have an artificial advantage over commercial if government-funded R&D is covered by the GPL” .

The problem with these claims was that there were already commercial programs released through the GPL, including ones originally based on government funding. Besides, many government research results are only released to a single proprietary vendor (and not even other proprietary vendors); that is far more exclusionary than a GPL release, which would at least permit multiple vendors to follow up on the research results. Clearly, if a government believes that it will achieve its goals best by releasing some research result under a FLOSS license, it should do so. If a government is willing to release results to a single proprietary vendor sometimes, it should be willing to release under a FLOSS license like the GPL sometimes, for the same reasons. The same slideset correctly notes that the perception of open source (FLOSS) vs. commercial source is flawed, but incorrectly leaves the impression that FLOSS software *cannot* be commercial software. Gates made similar statements. It does not matter if this mistake is intentional or not; the key is to realize that this is a mistake.

Some supporters of the BSD licenses argue that the BSD licenses are more business-friendly. Supporters of the GPL retort by noting a number of companies (such as Red Hat, MySQL AB, and Troll Tech) that strongly prefer the GNU GPL as evidence that the GPL is more business-friendly. Yet others argue that the LGPL is most business-friendly. The reality is that different licenses are better for different business models, but that is not my point. What’s interesting here is that many people argue over which license is more business-friendly -- which I believe is simply a catchphrase for “commercial”. If so many people are arguing about which FLOSS license is best for commercial use, then clearly commercial utility is considered a *good* property of a license by many.

References:

[1] Alshawi, Sarmand; Themistocleous, Marinos; Almadani, Rashid *Integrating diverse ERP systems*, a case study. in: The Journal of Enterprise Information Management Volume 17, Number 6, 2004.

[2] Bernroider, Edward; Koch Stefan: *ERP selection process in midsize and large organizations*. In: Business Process Management Journal, Vol. 7 No. 3, 2001. MCB University Press

[3] Chalifour, Josh: *TEC Talks to the Open For Business Project-Free and Open Source Software Business Models*- Part One: OFBiz. in: Technology Evaluation Centers, 2004

[4] Chan, Tzu-Ying; Lee, Jen-Fang *: A Comparative Study of Online User Communities Involvement In Product Innovation and Development,* Cheng Chi University of Technology &Innovation Management, Taiwan, http://opensource.mit.edu/online_papers.php

[5] Davenport, T.: *Putting the enterprise into the enterprise system, in*: Harvard Business Review, July-August 1998

[6] Dethlefs, Rolf: ERP, CRM *und Projektmanagement mit Open Source Software,* Unternehmenskritische Open-Source-Software bei der Probusiness AG. http://www.heise.de/open/artikel/68259/, 2006

[7] Elliott, Margret, Scacchi, Walter: Mobilization of Software Developers: The Free Software Movement. University of California, Irvine, 2005 http://opensource.mit.edu/online_papers

[8] Gamma, Erich; Helm Richard; Johnson Ralph; Vlissides John: Design Patterns*: Elements of Reusable Object-Oriented Software* Addison-Wesley Professional, Boston, 1995

[9] Golden, Bernard: Community: The Difference between Friend and Faux Open Source. http://www.navicasoft.com/Newsletters/August%20 2005%20Newsletter.htm, 2006

[10] Lauren, Andrew M. St.: *Understanding Open Source and Free Software Licensing* O'Reilly Media, Inc.; August 2004

[11] Pink, Kathy; Janke, Jorg; Wassmer, Anne: *Compiere User Manual Version 2.52e*, www.compiere.org, 2005

[12] Ramanathan, J.: *Process-based architecture for back office: successful supply chain requires EAI*, ASCET, Vol. 2, No. 253, 2000

[13] Silverstone, Len: *The Data Model Resource Book*, Vol. 1: ALibrary of Universal Data Models for All Enterprises John Wiley& Sons Inc., 2001

[14] Teltumbde, Anand: A framework for evaluating ERP projects in: International Journal of Production Research, 2000

[15] Tompson, James: *GNUe Common: A Developer's Introduction* 0.5.15 http://www.gnuenterprise.org/tools/common/docs/Developers, 2005