The Use of Open Source Software Platforms by Independent Software Vendors: Issues and Opportunities

Herwig Mannaert and Kris Ven
University of Antwerp
Prinsstraat 13
B-2000 Antwerp, Belgium
{herwig.mannaert,kris.ven}@ua.ac.be

ABSTRACT
The Cast4All Content Conductor Platform is an integration and provisioning suite to manage data broadcasting networks in general and digital cinema networks in particular. The framework makes extensive use of open source components and contains several extensions and modifications to those components. It is a typical case of an Independent Software Vendor (ISV) building application software on top of open source platform software. In the spirit of the open source movement, the extensions or modifications to the open source components could be contributed back to the community. However, in this paper we discuss several issues that companies face in such a situation. They extend far beyond the obvious decision whether to keep the developed code proprietary, and should not be neglected. It is argued that a closer collaboration between open source projects and independent software vendors would be beneficial to all.

Categories and Subject Descriptors
K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software development, software maintenance*.

General Terms
Design.

Keywords
Open Source Software, Independent Software Vendor, platforms.

1. INTRODUCTION
Open source software (OSS) has become a viable alternative for many commercial software packages. This is supported by the many success stories of the implementation of OSS which have been reported in academic as well as professional literature (see e.g. [4,11]). Relatively few studies have investigated the adoption of open source software by companies. Most research on this topic has primarily focused on *platform software* such as operating systems and web servers (see e.g. [3,6,12,13]). This is a logical choice since most OSS projects have firm roots in the server-side environment (e.g. Apache, Sendmail and Linux). OSS projects focussing on desktop applications have only appeared more recently (e.g. Mozilla and OpenOffice.org). Not surprisingly, most successful OSS projects are server-side applications. Moreover, preliminary studies suggest a relationship between the strategic importance of the IT infrastructure and the tendency to implement OSS [5,6]. It is hypothesized that new technologies such as OSS are more likely to be used for those parts of the IT infrastructure that have a low strategic value to the firm. This is consistent with the commoditization of IT that has been argued by some authors [2]. The strategic value of IT degrades over time and applications eventually become a commodity. When this happens, companies want to cut costs as much as possible. Switching to OSS is one way to realize this. Probably the most illustrative example of this is the Sabre system, which was during the 1970s the typical example of a system with high strategic value. When Sabre needed to be redesigned to cope with new requirements, Linux and MySQL were used as a platform [1,10]. One of the reasons why many companies choose OSS is that it offers them cheap and reliable software [3]. Additionally, large companies such as IBM and Novell are now actively supporting OSS, which will increase the customers' confidence in OSS. We therefore expect that OSS will continue to play an increasingly important role on the platform market.

OSS also provides interesting opportunities for Independent Software Vendors (ISV). An ISV can use proven OSS platform software as a basis to develop his own applications. By using OSS components, the cost for the final customer will be much lower than when commercial software is used. Commercial firms such as Microsoft and Hewlett-Packard encourage the use of their proprietary platforms by ISVs and offer them special "business partner" programs. However, currently there is no such cooperation between ISVs and OSS projects. This introduces several issues for the ISVs as well as the OSS projects that should not be neglected.

We will present a case study in which OSS was extensively used by an ISV in an advanced environment. In the project, considerable effort was made to customize and extend various OSS components. In such a case, the ISV has the possibility to donate these extensions and/or modifications back to the OSS community. In this paper, we argue that such a decision implies far more issues than the obvious question whether to keep the developed code proprietary. We will discuss several issues with respect to the possible efforts, problems, and dangers in the

Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE) May 17, 2005, St Louis, MO, USA.

process of donating these modifications back to the OSS community.

2. CASE STUDY: DIGITAL CINEMA DISTRIBUTION

As part of a joint and ongoing R & D activity, Cast4All and the University of Antwerp have developed a software framework for the distribution of rich multimedia content in general, and digital movies in particular [9]. This *content distribution management* software provides the following features.

- Distributed asset management for the movie files, including auto-detection and indexing of all files in the central NOC (Network Operating Centre), monitoring and reporting of the available files on all servers in remote theatres, and the possibility to centrally enter deletion orders for file replicas on remote servers.
- Integrated network management of all network nodes, including reporting of disk space and network connectivity, auto-configuration of receiver threads listening to multicast channels and the possibility to centrally enter start and stop commands for all processes on remote servers.
- Preparation and control of the actual data broadcasting, including forward error correction, automatic splitting of extremely large files (a standard MPEG-2 digital movie is 60 GByte) to improve the scalability of the error correction, multicast security at the transport layer, possible partial retransmission based on acknowledgements, and precise bandwidth control for multicast transmissions.

The application framework has been designed and developed as a set of components in the J2EE (Java 2 Enterprise Edition) component model, and is accessible through a web-based user interface. Internally, the software uses a number of open source software components, both at the application and the platform level.

- **Linux***:* the preferred operating system for the servers.
- **JOnAS***:* the *ObjectWeb* J2EE application server.
- **Tomcat***:* the *Apache* web server or servlet container.
- **Cocoon***:* the *Apache* XML publishing framework.
- **Axis***:* the *Apache* web services framework.
- **Ant***:* the *Apache* build environment.
- **Swarmcast***:* the forward error correcting code from *Onion Networks* for reliable multicasting.
- **Pftp***:* an application for real-time multicast transmissions with precise bandwidth control.
- **OpenSSL***:* the underlying for a dedicated solution for multicast security at the transport level [8].
- **IPTables***:* the preferred firewall on the servers.

Though the price has been the main and decisive argument in the decision to adopt these open source components, several other features have played a major role. Note that none of these are directly related to the availability of the source code.

- *Functionality:* several components are cutting edge technology. We mention here the *Ant* build environment and the *Cocoon* servlet, generally accepted to be the first fully-fledged XML publishing framework.
- *Flexibility:* the functional requirements could only be met through the ability to tailor the software. Examples are the forward error correction that needed to be adapted for the transfer of huge digital cinema files, and the dedicated solution for multicast security based on a generic and open library.
- *Quality:* some components offer a nearly unmatched quality. We mention here the very accurate bandwidth control of *Pftp* and the fact that *Linux* is one of the few operating systems that allows the correct configuration of the TTL (Time To Live) for multicast transmissions.

This usage of OSS is in line with traditional studies, showing the adoption of OSS for platform software such as operating systems and infrastructure servers. It also shows that several OSS platform software components can be combined with several application-level OSS components, to produce an integrated application framework in a complex environment.

3. THE ISSUE: HOW TO DEAL WITH EXTENSIONS

Though the findings of the previous section are quite traditional, we want to discuss some issues that are rather crucial from the viewpoint of the ISV (Independent Software Vendor). As is generally known, the success and acceptance of platform software is highly dependent on the adoption by, and the success of, ISVs. What makes the use of OSS platform software typical, is that developers are allowed, inclined, and often obliged to extend and/or even modify the open source components. Within the scope of the described development for example, the following extensions and/or modifications have been performed to open source software components.

- The Cocoon servlet for XML publishing has been integrated into the deployment and operational context of the Tomcat and JOnAS application servers.
- The Swarmcast library has been adapted to allow the automatic splitting of files into parts (and concatenation after transmission), in order to deal with extremely large files.
- The Cocoon framework has been extended to allow the flexible definition of multi-language user interfaces, and to enable graphical representations of functions and hierarchical trees.

3.1 Inhibitors to Contributing

According to the spirit of the open source movement, patches and even extensions could be donated back to the community. However, we have found that several factors may prevent this from happening. None of them stem from a possible reluctance of the ISV to share its intellectual property.

It is not that obvious for an ISV to find its way into the community and get extensions or modifications accepted. It takes for example some effort to get familiar with the practices and guidelines of the different OSS projects. Contrary to popular

belief, large OSS projects are governed in a structured way, placing high standards on, for example, code quality [14 This means that any patch submitted to the project must pass the peer-review process in which coding style, general quality and the interoperability with other parts of the application are assessed. The issues mentioned until now are in fact applicable for every potential contributor to an OSS project — even commercial entities (see e.g. [7) — and are not unsurmountable, given enough time is invested. ISVs however are faced with some additional problems. Some of the modifications to the OSS application will be too specific to include for the general public. Therefore, many OSS project leaders that want to keep a tight control over their code base will prefer not to include these changes. This was one of the reasons why the Firefox browser was designed to be kept to a strict minimum, and implement additional features by means of plugins which are kept separate from the Firefox code base.

The impact of the ISV's modifications on the OSS project will be even greater if it concerns the integration of two OSS projects. In this case study for example, Cocoon was integrated into Tomcat and Jonas. In such a situation, the implementation of the change must be coordinated and approved by two OSS projects. In practice, it is very well possible that these projects prefer to keep independent from each other, or seek cooperation with other, alternative OSS projects instead.

Even if the patch or extension does get accepted, it may require significant effort to maintain and further develop it in subsequent versions. In most cases, the modification must first be made more generic, so that is usable for a larger audience and that it fits better in the OSS project. This task is not always beneficial to the ISV. Furthermore, by merging the patches into the main repository, the ISV is likely to become the maintainer of this part of the application, given the often specialized nature of these patches. Consequently, subsequent changes made in other parts of the application may require the ISV to adapt his modules as well in order to keep the application working as a whole. When the patches wouldn't have been submitted, the ISV would be free to keep using a previous version of the application until he decides to upgrade to a newer version.

As mentioned before, the nature of the modifications made by the ISV may be quite specific. Hence, the number of external contributors that the ISV could attract will be small or even non-existent. Not having enough contributors or interested users has a negative impact on the motivation of OSS volunteers to keep maintaining the project. In the case of the ISV, the tendency to contribute the patches will be even more tempered.

The same issues make it unattractive to start a new OSS project providing patches to the existing OSS project (such as RTAI[1] which provides a real-time extension to the Linux kernel), or by starting a fork of the existing project (apart from the social barrier that exists against forking). In these latter cases, the effort required by the ISV will be even greater, since he will have to provide the necessary infrastructure to support the development (e.g. version control, bug tracker and mailing lists) and coordinate the efforts of possible contributors.

[1] http://www.rtai.org

3.2 Risks of not Contributing

From the viewpoint of the ISV however, having patches or extensions not being integrated into the open source project may also imply significant business risks. The OSS project may sooner or later tackle the issue, and may take an entirely different approach towards the problem. In this case, the ISV would end up with a parallel development track that competes with the OSS project. In case the ISV is a rather small company, as for instance in our case study, this would be highly undesirable. On the other hand, porting the existing application code to the newly developed extension framework, could lead to the rewriting of a significant part of the ISV application software.

But even if the OSS project does not tackle the envisaged extensions and simply ignores them, the ISV is still running a risk. The possibility exists in this case that several functions and application programming interfaces that are used by the extensions, will gradually become unimportant. This could lead to the modification or even disappearance of these programming interfaces, cutting off the ISV from future versions and upgrades of the OSS component. In the long run, this could even prevent the ISV from porting its application code to a new version of an operating system.

We can say that the ISV takes significant business risks in not contributing to and participating in the OSS project(s) it is using. This means that the ISV will often have a business interest in donating back extensions and/or modifications to the community of the OSS project.

4. CONCLUSIONS

In this paper we have have presented a case study of an Independent Software Vendor (ISV) using OSS as a platform. We have identified several issues that prevent the modifications made by the ISV to be contributed back to the OSS project. These issues include the effort required to get patches included in the project and the problems in attracting enough contributors. Note that none of these reasons are due to the reluctance of the ISV to share its intellectual property.

On the other hand, not sharing these modifications with the OSS project may face the ISV with considerable risks. Parallel development tracks or changes in programming interfaces, may force the ISV to rewrite application code or to foresake on future versions of the OSS component. This calls for additional efforts to lower the barriers for ISVs to submit and maintain their extensions or modifications in collaboration with the open source projects.

More research on this topic might provide useful insights on how other ISVs cope with the issues addressed in this paper. It may offer more information on the rate of participation in OSS projects by ISVs, and which factors currently hinder the further contribution of patches.

We argue that these results might also be useful for OSS projects, as it would show how OSS projects interact with ISVs. At the moment, we believe most OSS projects have poor relations with ISVs using their platform. Because of this poor relationship, OSS projects might loose out on some important know-how. By improving the relation with ISVs, it is likely that more ISVs will adopt OSS. Evidently, this would also be beneficial to the OSS

projects who will become more widely accepted and more successful.

5. REFERENCES

[1] G. H. Anthes. Sabre flies to open systems. Computerworld, May 31, 2004. http://www.computerworld.com/managementtopics/management/project/story/0,10801,93455,00.html.

[2] N. G. Carr. It doesn't matter. *Harvard Business Review*, 81, 5 (May 2003), 41–49.

[3] J. Dedrick and J. West. Why firms adopt open source platforms: a grounded theory of innovation and standards adoption. In J. L. King and K. Lyytinen, editors, *Proceedings of the Workshop on Standard Making: A Critical Research Frontier for Information Systems* (Seattle, Washington, December 2003). 236–257.

[4] B. Fitzgerald and T. Kenny. Open source software in the trenches: Lessons from a large scale implementation. In *Proceedings of 24th International Conference on Information Systems (ICIS)* (Seattle, Washington, Dec. 14–17, 2003).

[5] S. K. Kwan and J. West. Heterogeneity of it importance: Implications for enterprise it portfolio management. In *Academy of Management conference, Organizational Communication and Information Systems division* (New Orleans, La., August 2004).

[6] S. K. Kwan and J. West. A conceptual model for enterprise adoption of open source software. In S. Bolin, editor, *The Standards Edge: Open Season*. Sheridan Books, Ann Arbor, Michigan, 2005 (forthcoming).

[7] S. Lussier. New tricks: How open source changed the way my team works. *IEEE Software*, 21, 1, 68–72, 2004.

[8] H. Mannaert, P. Adriaenssens, and B. D. Gruyter. Multicast security for rich content distribution. In P. Dini and P. Lorenz, editors, *Proceedings of the Third International Conference on Networking* (Guadeloupe, French Carribean, Mar. 1–4, 2004), 561–565.

[9] H. Mannaert, B. D. Gruyter, and P. Adriaenssens. Web portal for multicast delivery management. *Emerald Journal for Internet Research*, 13, 2, 94–99, 2003.

[10] Mysql contributes to up to 80% reduction in total cost of ownership for sabre holdings. http://www.mysql.com/it-resources/case-studies/mysql-sabre-casestudy.pdf.

[11] J. S. Norris and P.-H. Kamp. Mission-critical development with open source software: Lessons learned. *IEEE Software*, 21, 1, 42–49, 2004.

[12] J. West. How open is open enough? melding proprietary and open source platform strategies. *Research Policy*, 32, 7 (July 2003), 1259–1285.

[13] J. West and J. Dedrick. Open source standardization: The rise of linux in the network era. *Knowledge, Technology & Policy*, 14, 2, 88–112, 2001.

[14] L. Zhao and S. Elbaum. Quality assurance under the open source development model. *Journal of Systems and Software*, 66, 1 (Apr. 2003), 65–75.