Open Source in Industrial Contexts – A Living Paradox?

Christoph Niedermeier, Reiner Schmid, Winfried Seidel

Corporate Technology
Siemens AG
Otto-Hahn-Ring 6
81730 München
{Christoph.Niedermeier,Reiner.Schmid,Winfried.Seidel}@siemens.com

Abstract: Open Source Software (OSS) has been well established in industrial software processes during the last couple of years. However, as the combination of OSS and commercial software is considered ambiguous due to license compliancy problems by quite a few people, an investigation of the industrial demands regarding software architecture and processes as well as usage and release of commercial products that contain OSS or are built upon OSS is desperately needed. This paper aims at providing guidance to the technical management that is considering usage of OSS in their software or embedded products and is therefore focusing on architectural and process-related issues. In addition, the pros and cons of collaboration with Open Source (OS) communities are discussed.

1 Introduction

Usage of Open Source Software (OSS) in industrial and business contexts has been increasing dramatically during the last few years [OST99]. In spite of recent confusion regarding the Linux operating system, this trend is expected to continue in the future. However, there has been a lot of controversy concerning the problems that might arise when OSS is used in commercial software products or in embedded systems [OSS03]. These problems mainly result from possible conflicts between OSS licenses – the most well-known and widely used license being the Free Software Foundation’s (FSF) [FSF04] General Public License (GPL) [GPL91] – and proprietary software licenses that are usually associated with commercial software products and also – though less apparent – with software being part of embedded systems.

Due to these problems, quite a number of people consider the combination of OSS and proprietary commercial software products a paradox. Nevertheless, the usage of OSS in such products is a common practice advocated by many individuals and organizations. Whereas the advantages of using OSS have been extensively discussed elsewhere [FEL02], the problems that could arise and possible strategies how to avoid them well deserve a more detailed treatment. Therefore, the purpose of this paper is to analyze these possible conflicts and point out strategies either how to avoid them altogether or how to minimize risks and efforts that might turn out later by carefully adapting the software processes.

Major implications of OSS licenses and possible conflicts with proprietary software processes will be discussed in the paper. In particular, the following aspects are important: The need to keep confidential information / proprietary know-how secret may collide with the requirement to publish all modifications of OSS. Therefore, a clear separation between OSS and proprietary software is required to avoid these implications imposed by OSS licenses without infringing them. Solutions how to achieve this separation may have effects on the software / system architecture, software processes or legal and business procedures. In this paper, we focus on architectural and process-related procedures. An important example for the latter is long-term cooperation with OSS communities in order to establish a trust-relationship and ensure mutual benefits.

The main purpose of this paper is to provide decision support for the technical management regarding the design of software processes and software architectures that are compliant with the obligations stipulated by OSS licenses and allow companies to benefit from the advantages of OSS without sacrificing the proprietary information required for their core business.

2 Open Source and Industrial Software Development

Whenever OSS is used in a commercially oriented environment a number of questions need to be answered before development of products or services can proceed. These questions should be answered before major effort is involved to avoid later problems. Important in this respect is to distinguish between two different kinds of OSS usage: use of OSS as development tools and IT infrastructure components or use of OSS as integral part of software products that are sold commercially. Even with restrictive licenses as for instance the GPL the first form of usage is uncritical. The later form forces the product development to deal with business decisions related to liability, legal issues and competitive advantages to a considerable extent and therefore is likely to have impact on software development processes [WIE01].

In order to be able to use OSS in a product development process, the development organization has to understand the obligations incorporated into OSS licenses to a certain extent. Although not every developer needs to become a lawyer and license expert, some basic knowledge is mandatory for the involved developers and managers. In case of small internal projects technical and organizational aspects, like appropriateness, feature-set and quality of the used OSS components are driving factors for a project decision. When OSS components are used as integral parts of a product sold commercially, the implications have to be worked out in advance before a development project gets momentum. This step should be accomplished before basic OSS components of the architecture are incorporated for development in order to avoid later changes that might cause considerable costs.

In the simplest case the product is solely built by integrating existing open source components without major modifications. Depending on the accompanied licenses of the software components the resulting software product can be kept proprietary for some licenses, e.g. for BSD-like licenses, but must be made available to customers in source code for more restrictive licenses like the GPL [OSI04] that require derived work to be published. Contrary to widespread opinions, there are OSS licenses that do not require derived work to be published.

Further considerations are needed for development of products that combine proprietary and OSS components. If it is required to keep intellectual property included in proprietary software components as an asset of the company, these components need to be clearly separated from the OSS components, in particular those published under GPL-style licenses. The so-called *viral effect* may contradict with the intellectual property of a company if not taken care of beforehand. The problem of mixing code and possibly implementation ideas between both worlds should not be underestimated. A clear cut separation between open source and proprietary software development teams is the right organizational structure. This strongly holds for enterprises that are in the business of developing software in the same field where competing solutions exist for the proprietary and OSS domain.

Benefits of OSS usage in commercial environments are manifold and are widely recognized [FEL02]. The possibility to modify the source code of a given component helps in fulfilling customer demands in a flexible way usually not possible with closed source components. Despite all these benefits, there are also challenges to be mastered in order to become or stay competitive when OSS is used by commercial organizations. From a technical point of view the OSS being used should be analyzed if it is appropriate for the intended task. It may take some time to get the status and viability of an ongoing OSS project. Beside this the technical quality of each piece of software needs to be analyzed. Legal aspects are another issue to be considered. Liability and copyright agreements related to accompanying licenses need to be considered in line with the project and business goals.

Strategic considerations in the pre-product phase should include make-or-buy decisions that can be applied to OSS in a similar way as to commercial software. Although the OSS can be maintained in-house it is also possible to have contracts with OS service companies that deal with packaging and quality assurance of the envisioned software components. However, OS service companies are usually not offering unlimited liability but in practice liability is also limited in case of proprietary software.

3 Towards an OSS License compliant Software Process

As has been shown in the preceding section, OSS licenses do not necessarily fit neatly into the traditional software development process in industry. Nevertheless some building blocks for a new kind oft software process are available that enable development organizations to cope with the specific demands of the use of OSS [GEH04]. This survey is just a first step towards establishment of a unique software process that is compliant with OSS licenses. Therefore, it just indicates necessary elements of such a process rather than fully describing it in a structured way.

Many commonplace OSS licenses require giving reference to the OS code used or even demand the publication of the source code if it has been modified compared to the available releases. There are several possibilities to cope with these demands.

The first and simplest one is to avoid changing OSS code at all, if this is possible. In this case, the obligations from the use of the open source code are usually limited to giving a reference to the source code, for instance the internet resource offering the code. If the required, changes to the OSS code are limited in scope, and if they are useful to a broader audience, one should consider contributing them as patches to the open source community. If the patches are accepted by the project administrators and are then published, the efforts for making them available can be saved, as the reference to the code source is sufficient to comply with the stipulations of the OSS license.

If the patches have not been accepted or there is no way to introduce them in the OS project itself, the modified software used in the product must be made available under the terms of the OSS license of the original software. The OSS can be made available in different ways, e.g. by adding a CD to the product containing the source code or more efficiently by offering the source code for download via a web site or an OS development portal.

Most of the liability risks can be dealt with in the same way as with proprietary software. Additional safeguards may be applied by painstaking evaluation of the quality of OSS software and the environment in which the OSS is developed, e.g. viability of the community, test procedures used and the number of active developers. Active participation in OS communities can improve software quality, if needed.

Some widespread OSS licenses, the GPL being the most prominent example, require derived work of OSS code also to be open source, if it is to be passed on. In many cases, however, this is not possible with industrial software. For instance, software for embedded systems often relies on proprietary information only available under non-disclosure agreements. In this case a clear separation of OSS and proprietary components is required. The related questions are complex and not finally solved yet.

Proprietary Linux kernel modules are often used to solve this separation problem. These are then distributed as binary modules only. This procedure is, however, according to the opinion of Linux developers, in particular Linus Torvalds, not consistent with the GPL and therefore not legal [FIN03]. This is, however, accepted if the mechanisms at the basis of the respective kernel module have been developed independently from the Linux kernel. Given the lack of legal substance for this procedure, it involves risks.

Another important way to separate OSS and proprietary components is to separate according to layers, as dynamic linking of proprietary software is permitted with the LGPL for instance. In particular for embedded systems that often use static linking instead of dynamic linking the use of this concept is more complicated. In this case, it is required to also supply generation rules for the executable instead of providing the final binary.

Active open source projects continue to develop. As problems may arise if the current version of the OSS and the version used in the industrial project are out of sync, procedures are required to avoid these. There are essentially two possibilities: periodic synchronization and active involvement in the respective OS community. Periodic synchronization of the product with the newest release avoids the necessity to maintain older versions of the source code and to make them publicly available. An active involvement in an open source community offers additional benefits. The direction in which the open source software develops can be observed and be affected.

4 Conclusion

In the paper we suggested building blocks for a future software process that allows usage of OSS in industrial software and product development while avoiding the problems that might arise from conflicts between obligations of OSS licenses and business requirements. More work has to be done in order to shape and improve that software process, but some basic guidance for the technical management can already be provided today. While in principle there seem to be no blocking points regarding usage of OSS in combination with proprietary software products, the following rules should be obeyed:

First of all, each case should be evaluated individually so that specific requirements of the product to be developed as well as specific properties of the OSS to be used in that context are taken into account. As most enterprises do not yet have extensive experience with OSS, a learning phase for the specific product context is advised. Of particular importance are strategies regarding control of future obligations arising from OSS license terms as well as adaptations of the software development process. Active collaboration in OSS communities may be reasonable but should occur on a long-term basis as otherwise a negative impact on the corporate image may nullify the positive aspects of such a step. A fruitful cooperation between enterprises and OSS communities is possible if the mutual interests are understood and respected by both sides.

References

[FEL02] Feller, J., Fitzgerald, B., Understanding Open Source Software Development, Addison Wesley 2002

[FIN03] Fink, M., The Business and Economics of Linux(TM) and Open Source, Prentice Hall, Upper Saddle River 2003

[FSF04] Free Software Foundation, http://www.fsf.org/

[GEH04] Gehring, R. A., Lutterbeck, B., Open Source Jahrbuch 2004, Lehmanns Media, Berlin 2004

[GPL91] Gnu General Public License, http://www.gnu.org/licenses/gpl.html

[OSI04] Open Source Initiative (OSI), http://www.opensource.org/

[OSS03] Workshop “Open Source Software in Industrial Environments 2003”, NetObject Days, Erfurt 2003

[OST99] Osterhout, J., Free Software Needs Profit, Communication of the ACM, April 1999

[WIE01] Wieland, T., Open Source im Unternehmen, in: v. Raison, A., Schönfeldt, R., Editors, Linux im Unternehmen, dpunkt Verlag, Heidelberg, 2001