Zeitschrift: Comtec : Informations- und Telekommunikationstechnologie = information and telecommunication technology

Herausgeber: Swisscom

Band: 81 (2003)

Heft: 12

Artikel: Open source

Autor: Balsiger, Peter

DOI: https://doi.org/10.5169/seals-876712

Open Source

Today, many successful software products are available for free and their source code is released as open source. The most important ones are the Apache web server (market share of 64%), the MySQL database (4 million users), and the GNU/Linux operating system. These products are not only used by computer enthusiasts but increasingly also by companies for commercial electronic services. The success of open source software is astounding, considering the fact that it is competing with products from some of the largest software companies in the world. We take a look at the open source movement to see what benefits can be gained from it, either through direct use or by adapting some of its strategies or processes.

The programme "Software and Security Technologies" explores new opportunities for Swisscom arising from current software technology trends and assesses the impact these trends may have on the efficiency of the service creation process, as well as on the quality of the services. In particular, the programme focuses on IT-related areas such as Web Services, trends in software architectures, Ontology Languages and Knowledge Representation technologies, and advanced software security issues.
With its Innovation Programmes, Swisscom Innovations follows the objective of assessing the impact of technological developments, finding new business opportunities, promoting technical synergies, and developing concrete innovation proposals. Further, the expertise built up enables active engineering support of business innovation projects.

PETER BALSIGER

The open source movement was officially initiated in 1984 by Richard Stallman with the creation of the GNU project (GNU is a recursive acronym for "GNU is Not Unix"). The project aimed and still aims at providing a complete set of computer software consisting solely of open source software. From that time on, open source software has gained more and more attention and with the success of Linux and the Apache web server, the *'open'* concept has been widely adapted. Competitive open source software is now available for almost any area. Notable examples are the Apache web server, Mozilla (web browser), Linux (operating system), Eclipse (development environment), MySQL (database), Open Office (office suite), GIMP (image manipulation), and gcc (C++ compiler).
Nowadays, *'open'* is no longer used only for software: *Open course ware* provides course material of schools and *open directories* or *open diaries* are available.

"'Free software' is a matter of liberty, not price. To understand the concept, you should think of 'free' as in 'free speech', not as in 'free beer'."
The Free Software Foundation

The terms *"free software"* and *"open source software"* are often used interchangeably. Both terms describe similar requirements for open source software, only their motivation differs. Free software puts the weight on the "free" aspect of the software, and thus requires that all derivation of free software must again be free. Open source, on the other hand, puts weight on the fact that access to the source is open to all. In this article, we focus on the open source aspect and use both terms synonymously.
The official definition of open source [1] by OSI states ten conditions that an open source licence must adhere to. The following points comprise the core of this definition:
- The software can be freely redistributed.
- The complete source code of the software is freely available.
- Derived works must explicitly be allowed.

The definition of open source does explicitly allow selling open source software, as long as its source is freely available. Companies like Red Hat, SuSE, IBM, or HP have shown that this is a viable business model.
Furthermore, Eric Raymond states in [5] several unwritten rules or taboos that define the culture of the open source community. It is important to understand these rules to be able to successfully participate in open source projects. Summarised, the most important rules are
- projects should not be split up (forked) into multiple continuations,
- changes to a project are only distributed with the authorisation of the project leader responsible,
- the name of a person is never removed from a history, credits, or maintainer list.

These rules should be observed in all participations of open source projects.
In the following, we will take a closer look at the different licensing schemes, the possibility to use and extend open source products and, finally, summarise some learnings from open source project management.

Licences

Every piece of work created is automatically protected by local copyright laws. Thus, it can only be used by its copyright holder but not by others. A licence is used to grant others to use it. The licence thus does not actually limit access to some work but instead provides – probably limited – access to it.
A multitude of licences are currently used, but they can be roughly divided into four groups. The *non-free* licences comprise the licences of most commercial applications. To customers paying for the licence, it allows using an application, but it does not give any further rights. The direct opposites are the *boundless licences,* which give away all rights to anybody with no restrictions whatsoever. Work with such a licence is often described as being public domain. Any open source licence needs to fulfil the requirements of the open source definition mentioned above. The conforming licences can be grouped into two categories. The *demanding licences* put some specific but small demand on the licensee. These demands may not be strong, because that would limit the possibility for free derivation and thus violate the requirements for open source. A typical demand is to simply cite the original author or project. Prominent examples of this type of licences are the BSD or Apache licences. The *enforcing or copyleft licences* do not put any demand on the usage, but enforce a specific licence for any derivation. The most prominent licence of this style is the GPL (GNU General Public Licence). It allows free use and derivation, but requires that all derivations are published using the GPL as well.

Using Open Source Software

The many useful and competitive applications released as open source make it increasingly important to know how such applications can be used in a commercial setting. The following scenarios show some possible uses of open source. Similar to any commercial application, an open source application can simply be used as it is. In this case, there are no restrictions or requirements and usage is usually free of charge. Open source products do not give any legally binding

guarantees to work as intended. Because there is no company responsible for the product, there is no-one who could be liable for damage caused by the product. Almost all licence agreements of commercial products explicitly exclude any liability for damages caused by it. Only rarely can a company be held liable for shortcomings in their products. While no company is responsible for an open source product, it is still most of the time easier to get support or bug fixes from the relevant development community than to get the same support from a company. Additionally, the future of an open source product is secured. There is no company that could go bankrupt or that is bought by a competitor and thus removes the product from the market.
Another possibility to take advantage of open source products is to distribute them to customers, either alone or as part of a larger product. No problems arise as long as only products with demanding licences are used. But if at least one of the used products has an enforcing licence, like the GPL, the complete product has to be released using this licence. Many demanding licences are compatible with enforcing licences and thus make no problems. However, any part that cannot be released with the enforcing licence – for example because it was commercially licensed or because internal developments cannot be publicly released – prohibits the use of the specific product.
Because open source products are released with their full source code, it is possible to make extensions. All open source licences will allow this without restrictions, but the distribution of the extension differs on the type of licence involved. Demanding open source licences only require the specific demand to be fulfilled, like citing the original author. It is even possible to sell the extended work. As an example, it would be possible to make an extension of the Apache web server and sell it to customers, as long as a note on the products indicates that it is based on Apache software. Enforcing licences, on the other hand, require the extensions to be published using specific licences, for example the GPL. This means that although GIMP can be freely derived, any derivation must be released under the GPL as well. This does not prohibit selling the derivation to customers, but it requires publishing the complete source of the extension free of charge and must again allow extensions.

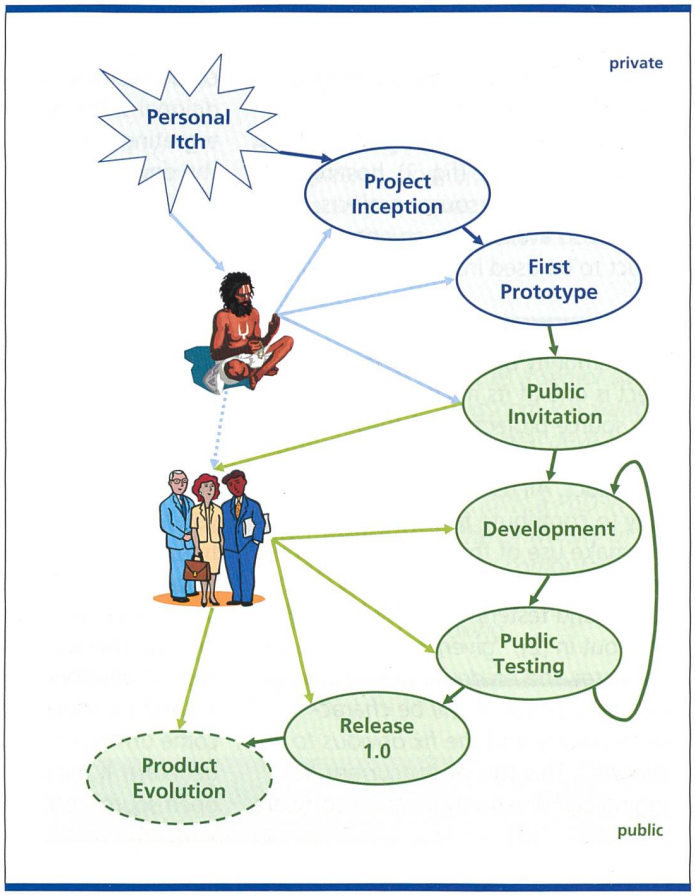

Fig. 1. Overview of the general open source development process.

Learning from Open Source

Several open source projects have succeeded in producing large and complicated products of good quality, all without many of the traditional development processes. It therefore stands to reason that some aspects of the open source development process [2] (fig. 1) could be applied to traditional projects.

Leadership

Open source projects are usually carried out without traditional project organisation. The lead of open source projects is taken by the person who either initiated or most actively promotes the project. Because no money is earned by anyone involved, the motivation and the skills of each contributor count. The project leader must lead among peers. In the long term, project leadership may change. Internal software development might similarly benefit by using a peer leader to direct a project. Of course, traditional projects have many more administrative requirements than open source projects. To take some burden from the peer leader, it might thus be useful to provide a responsible person for administrative tasks.

Communication

Most open source projects are done by people from all over the world, people who never actually see each other face to face. These people have to rely solely on technical communication facilities. Regular meetings and decision finding is not possible. Nevertheless, open source projects succeed in achieving good communication and fast decisions by using sophisticated web portals supporting the developers. These portals usually provide message boards, bug tracking, documentation and file space, and of course a repository for the source code. This infrastructure allows anybody to participate in a project and still be able to know exactly what was contributed by whom. Furthermore, all communication between team members is open for all, thus preventing separation of the project team into indi-

Pointers

OSI: www.opensource.org
FSF: www.fsf.org
GNU: www.gnu.org
SourceForge: sourceforge.net

vidual groups [3]. Such development portals are also available for traditional projects and could help to greatly improve communication and thus efficiency. The most popular development portal currently is SourceForge (fig. 2), hosting about 65 000 open source projects. This portal is also available as an enterprise product to be used inside companies.

Concurrent Debugging

The community behind an open source project is one of its main advantages. Open source projects treat their users as co-developers, not just as people using the product. All users are given the possibility to contribute to the project. Many users make use of this opportunity, thus drastically increasing the number of developers and testers. As Eric Raymond points out in [2], *"given a large enough beta-tester and co-developer base, almost every problem will be characterised quickly and the fix obvious to someone"*. This sort of concurrent debugging can drastically increase software quality and at the same time increase customer satisfaction. Treating users as co-developers can easily be done by traditional software projects as well. By integrating the customer more tightly into the development and testing process, similar benefits could be achieved as for open source projects.

Why Open the Source?

There are several reasons why a product should be released or created as open source ([4]). The most obvious reason to create a product as open source is cost sharing. If you successfully attract external developers to contribute to your project you actually get additional manpower for free. The same is true for external users helping you to find and correct bugs. Another benefit is risk spreading. Often, inhouse developments heavily depend on specific developers. If these developers leave the company, the software can become un-maintained and can start to degenerate. If the product is released as open source, other developers will hopefully take care of it and the dependency on the original developers is reduced. Releasing a product as open source can also help to break the market leadership of competitors. If a competitor uses its market dominance to push other products out of the market, releasing the product as open source can be a viable counter strategy. An open source product cannot be pushed out of the market, because there are no financial requirements for maintaining it. Thus, customers risk less using it and may thus be more inclined to use it. Netscape used this strategy when Microsoft started to monopolise the browser market. By making their browser open source in the

Abbreviations

FSF	Free Software Foundation
GNU	GNU is not Unix
GPL	GNU General Public License
OSI	Open Source Initiative

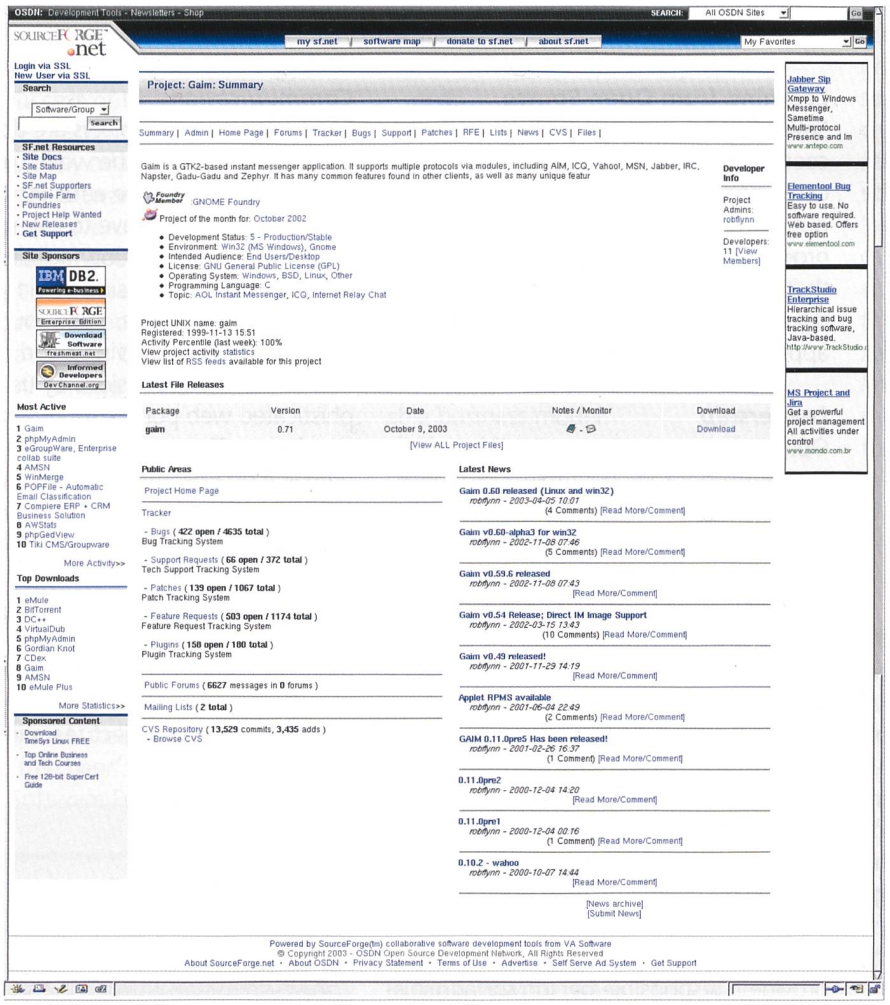

Fig. 2. Example of an open source project hosted on SourceForge.net.

Mozilla project, they ensured that the product stayed on the market.
Often, a company is more interested in selling services than in selling a software product. Making the software product open source allows others to create derivations and extensions. This increases the market for the service business. Possible services for open source products are product support (e.g. Red Hat or SuSE), providing content accessed through the product, or generating traffic when using the product.
Surprisingly, it is even possible to release a product as open source and still get customers to pay for it. A product can be released using an enforcing licence like the GPL, creating the usual open source movement and allowing potential customers to freely experiment with the product. If a customer actually wants to create a derivative work, he has to release his derivations as open source as well. Often this is not desired, especially if the derivation contains business logic of the customer. In that case, the same product can be sold to the customer under a different licence, allowing derivations without enforcing the opening of their source. Such a strategy is especially useful for frameworks, libraries or products which are used as a base for business relevant applications. As an example, the MySQL database is published using this model. It can be freely used under the terms of the GPL, but if a customer wants to release a product containing MySQL without being forced to release it as open source, he has the possibility to buy another licence.
Supporting open source projects is also a good way of building a positive reputation in a chosen market. Such sponsoring can not only consist of financial donations. It is also possible to donate resources, like hardware, server space or communication resources. Such donations can actually be quite cheap for a company and still have great impact on a typical open source project with a tight budget. As an example, the publishing house O'Reilly successfully uses sponsorships to positively influence their reputation in the development community.
Clearly, if a product is sold to customers, it may not be useful to release it for free as open source. But most software developed is mainly used for internal purposes and was never intended to be sold anyway. Releasing such software as an open source product will not interfere with revenues and might even increase them.

Conclusion

Many open source projects have shown that they are capable of producing complex products of high quality. Many of these products are already widely used and there is no reason why open source products should not be used in a commercial environment as well. Actually, as shown, many benefits indicate that it may even be favourable to commercial software and service companies to release a product as open source. When creating or even buying software products, it should thus be considered if derivations and evolutions of the product could later be released as open source. Furthermore, it should be considered if it is possible to participate in open source projects, whether by actively supporting development in a project or by providing resources or money to specific projects. [10]

Acknowledgement

The research work and report on Open Source Software was been commissioned by Swisscom IT Services.

Peter Balsiger *works as a senior software engineer at Swisscom Innovations. He has been part of several development projects, including projects with open source products. He holds a PhD in computer science from the University of Bern for work on theoretical computer science and logic, mainly in the field of automated theorem proving.*

References

[1] www.opensource.org/docs/definition.php
[2] E. Raymond: "The Cathedral and the Bazaar", 2000
www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar
[3] A. Cockburn, "Agile Software Development", Addison Wesley, 2001.
[4] E. Raymond: "The Magic Cauldron", 2000
www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron
[5] E. Raymond: "Homesteading the Noosphere",
www.catb.org/~esr/writings/cathedral-bazaar/homesteading/

Zusammenfassung

Open Source

Verschiedene Open-Source-Produkte sind ihrer kommerziellen Konkurrenz ebenbürtig oder zum Teil sogar überlegen. Die erfolgreichen Open-Source-Projekte haben ausserdem gezeigt, dass es möglich ist, auch ohne eine herkömmliche Projektorganisation komplexe und qualitativ hoch stehende Software zu entwickeln. Es ist deshalb sinnvoll sich zu überlegen, wie diese Produkte innerhalb eines Unternehmens Gewinn bringend eingesetzt werden können und welche Aspekte von Open-Source-Projekten die traditionellen Projekte verbessern können.
Die originale Verwendung eines Open-Source-Produkts ist meist unproblematisch. Erst bei seiner Einbindung in eine grössere Applikation oder bei seiner Erweiterung können unter Umständen Schwierigkeiten auftreten.
Generell lässt sich sagen, dass es sich lohnen kann, Open-Source-Entwicklungen zu verwenden und dass es häufig sinnvoll ist, eigene Entwicklungen als Open Source zur Verfügung zu stellen, bei Open-Source-Entwicklungen mitzuarbeiten oder solche Entwicklungen zu fördern.