# Open Source Automation Development Lab (OSADL) eG

Whitepaper ¨Free and Open Source Software  (FOSS)¨

www.osadl.org

# Open Source software in the automation industry – just a fashion or a change of paradigm?

by Carsten Emde, Open Source Automation Development Lab (OSADL) eG

## Introduction

As is the case with many other fields, Open Source software has gained a foothold in the automation industry and, at first glance, has opened up many new possibilities, especially for extending the service life of machines and other embedded systems. But what exactly is Open Source software? What are the considerations for its use, development and conveyance in an industrial environment?

In the following, the principles of Open Source software will be explained as well as guidelines for its specific use will be given.

## What is Open Source software?

The terms "Open Source software" and “Free Software" represent different ideological perspectives, but legally and practically, both terms are equivalent descriptions of a specific type of software licensing. This article will therefore use the common term "Free and Open Source Software" (FOSS) for this type of license. Many active FOSS projects such as the Linux kernel use the GNU General Public License (GNU GPL). The GNU GPL grants the rights to unrestrictedly

1. run the program, for any purpose,
2. study how the program works,
3. convey copies,
4. modify the program and convey it in its modified form.

The second and fourth points require free access to the source code. This access is therefore a necessary but not exclusive condition for qualifying the software as Open Source software. When the software is conveyed to others, the recipient must also be able to acquire the mentioned four rights. If the user only uses the software himself and does not convey it, all the requirements from the Open Source licenses are moot. Contrary to a frequently-stated belief, money may be charged for FOSS, but only for associated services such as the effort required to convey FOSS, and not for the license itself. Apart from this consideration, a service provider may of course also charge customers for the activities required to further develop and maintain FOSS.

## What is "copyleft"?

The word "copyleft" is a newly-created term for a particular property of an Open Source license. This play on words employs the term "left" as the opposite of "right" and also in the sense of "to leave" (somebody something). But none of them actually explains what "copyleft" is.

Copyleft does not come into play, unless existing Open Source software is modified or extended. Normally, the author of such modifications or extensions can freely decide how to license them. If, however, the license of the original software contains a copyleft provision, the author who modified or extended the software voluntarily refrains from using his normal licensing right when converying the software, but must use the original license for the modifications or extensions. As a consequence, such licensed Open Source software always remains free and open, even after any modification or addition. It is, for example, generally accepted that the choice of a copyleft-provisioned license of the Linux kernel largely contributed to its quality and its success.

The presently-known different Open Source licenses differ according to the strength of the copyleft effect. The BSD (Berkeley software distribution) license does not dictate to the licenseholder which license will govern the provided software along with any applied modifications and additions; it is therefore an Open Source license without copyleft. An example of an Open Source license with strong copyleft is the above-mentioned GNU GPL; it forces the licenseholder to retain the original license in its identical form for the provided software and any modifications and additions.

## Pragmatism versus ideology

The concept of "Free Software" was first presented by Richard Stallman in 1983, and it developed into the formulation of the mentioned GNU GPL. The underlying idea is that a program in a computer language with a source code that is unavailable to the user represents a type of restriction that should be rejected for fundamental ethical considerations. Only when the source code of a program is made available the desired freedom is ensured, according to Stallman.

The alternative term of "Open Source software" developed by Bruce Perens, Eric Raymond and others toward the end of the 90s has a more pragmatic aim. Disclosing the source code generates a much larger team of developers in comparison to proprietary software, and this team can create software products with a much higher level of quality and stability.

Starting from around this time, FOSS was increasingly used in an industrial context and in automation industry, that is, in embedded systems or for mechanical and plant engineering. When you ask the manufacturers of industrial products why they prefer FOSS, you frequently hear responses such as: "the machine builder has complete control of his machine," "independence from discontinuations," or "avoiding vendor lock." This indicates that the original motivation envisioning the freedom of the software user was justified to a certain degree. In fact, one of the special features of FOSS is that the user is entitled to comprehensive usage over time instead of a right of use for a very restricted period, as is the case with proprietary software; of course, the right to use FOSS is non-exclusive. In addition, the mentioned pragmatic reasons of the Open Source movement make this type of software attractive to industry, since quality and stability have always been a chief requirement for industrial software.

## An open knowledge economy

Perhaps even more relevant than the above discussion of freedom and openness is the perspective that views the FOSS development model in economic terms. In this context, FOSS has been scientifically analyzed from the vantage point of economics and business administration. The wealth of knowledge and experience of a company is considered capital in terms of the unique position of a company. Part of this capital is suitable for distinguishing a company’s products from those of a competitor ("differentiating know-how") whereas another part of the capital does not possess this ability ("non-differentiating know-how"). The latter part is generally larger. To maximize economic success, a company must invest a maximum amount of its resources in developing and maintaining the differentiating know-how. It should refrain from individually investing in non-differentiating know-how. Precisely because the know-how is non-differentiating, it is recommendable to work jointly with other companies and even competitors on FOSS projects. The savings are obvious since unnecessary parallel developments are avoided.

Qualms are continuously expressed that the use of a FOSS operating system with a copyleft license will force companies to disclose company secrets in process technology and applications. However, this is not the case because the application interface between the operating system (the "kernel") and the application generally follows the so-called POSIX standard. This interface standardized by the IEEE and Open Group is supported by a wide range of different operating systems; an application associated with a specific kernel therefore cannot be considered a change or addition to the kernel. Thus, the application is considered as not "derived from the operating system" and therefore does not have to be released under the GNU GPL license. Mechanical engineers can therefore use Linux as an operating system that is licensed under the GNU GPL without having to disclose their applications or other process details.

## Joint development of basic technologies and relevance of open standards

The joint development of basic technologies by a large number of market participants has another positive effect in addition to the mentioned reduced costs: Standardization is achieved as a side-effect because the joint development of basic technologies and their use necessarily requires jointly-defined interfaces and protocols. This argument also applies in reverse: Many existing open standards were developed earlier for Open Source projects such as Ethernet, TCP/IP and other network protocols, page description languages, etc. Outstanding implementations of these standards are therefore available for FOSS-based systems. These frequently are of clearly higher quality than subsequent implementations developed for proprietary operating systems.

## Countering globalization using globalization’s tools

The joint development of basic technologies would be impossible without globalization, and especially without the Internet. The same globalization is also responsible for subjecting some companies to increased cost pressure from competitors in countries with lower labor costs. If this cost pressure is countered by the joint development of basic technologies, it is quite possible that a free software component that a German mechanical engineer obtains from the Internet and uses will have been developed in a country previously associated only with software piracy. By using FOSS, the disadvantages of globalization are offset to a certain degree by its advantages.

## "Joint development of basic technologies" – how does that work in practice?

The usual FOSS development model implies that a company which is the first to require a particular software functionality is forced to develop it on their own and disclose the source code according to the Open Source license conditions. Consequently, a pioneer in a specific field will assume the costs for developing a software that subsequently can be adopted freely by less advanced competitors. A company may accept the basic requirement of disclosing source code as a matter of principle of Open Source, but the resulting unbalance in financing is unacceptable. To achieve a more equitable distribution of development costs among a large number of beneficiaries, an organization is needed that is jointly financed by numerous interested companies and is able to commission software projects requested by a majority of the participants. One of these organizations is the cooperative Open Source Automation Development Lab (OSADL) eG that was founded at the end of 2005 (http://osadl.org/). Since then, OSADL has realized a substantial number of jointly financed projects such as further developing the real-time capabilities of the Linux kernel, creating various Linux drivers, developing board support packages, migration tools, etc.

## Recommendations on the use of FOSS in industrial systems

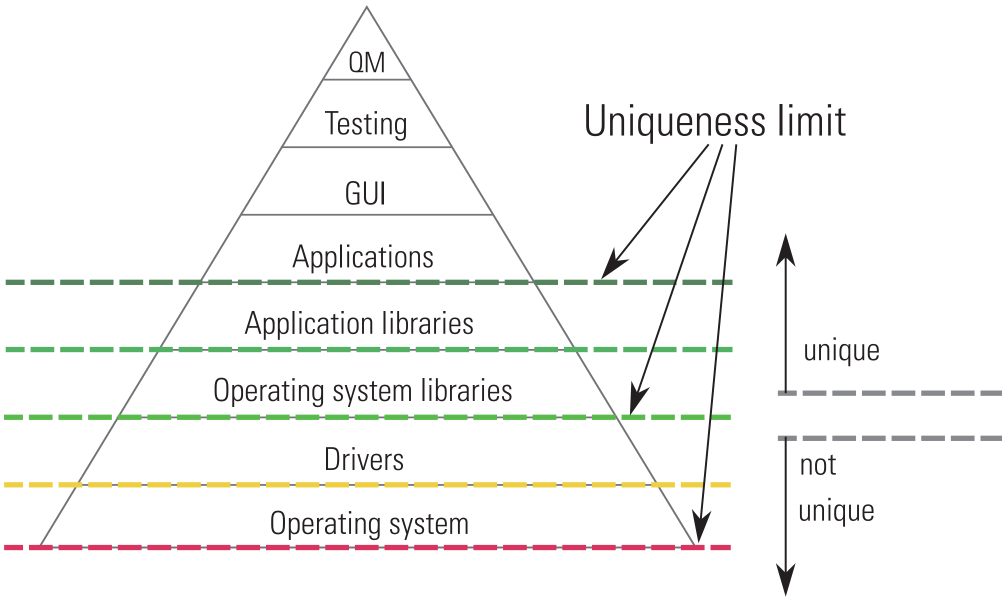
As mentioned, not all areas of a company that develops or uses software are suitable for participating in a FOSS project; only those areas should participate that use technologies of general interest. Before a decision is made to participate in a FOSS project, one must determine how much the project will affect company knowledge and experience that cannot be disclosed without endangering the company's market position. Guidelines can be offered for specific sectors, however substantial deviations can arise in individual cases. Typically, machine builders do not gain a unique market position from the features of the operating system and its programming interface. On the other hand, special controls for unusual requirements may necessitate developing components for operating systems with unique benefits for the respective machine builder. In such cases, it would be unreasonable to disclose the source code, and a FOSS operating system with a copyleft license could not be used. It is, however, also conceivable that software initially considered worthy to stay proprietary may not be so upon closer analysis, and the advantage of open development can justify the release of the software in a FOSS project.

The various levels of know-how of a company are best illustrated by a pyramid where the bottom level of the pyramid includes basic technologies without a unique benefit, and the top area includes highly proprietary intellectual property. Somewhere between the base and the top of the pyramid is the "uniqueness limit" to be individually determined for a specific company (see Figure). If this limit is directly at the base of the pyramid, there is little or no expertise that can be exchanged with competitors, and FOSS (at least with copyleft) cannot be used. If the uniqueness limit is higher (yellow and green lines), there is no particular reason why FOSS with copyleft cannot be used .

Once the decision to use FOSS or to develop an independent FOSS project has been made, legal advice should be obtained. This concerns the way in which the customer must be informed that FOSS is being used in a specific project, how best to disclose software, and which rights and obligations are associated with the granted FOSS license. In addition, the company should precisely determine to what extent assertion of own patents is affected. Generally, legal considerations only concern rules of the game and methods that are easy to observe once they are known. Attorneys who have specialized in FOSS can provide assistance, or legal advice can be obtained from the private Institut für Rechtsfragen der Freien und Open Source Software [Institute for Legal Issues on Free and Open Source Software] (ifrOSS, http://ifross.org/). Furthermore, OSADL members have free access to a knowledge data base of best practices of using Open Source software in industrial products and get legal advice through the organization's General Counsel. Such legal advice is of particular importance, since the legally compliant conveyance of products includes the observance of license obligations and is part of the management's duties.

## Conclusion

The use of FOSS in the automation industry is not a panacea, however, FOSS often can measurably reduce costs and improve quality. If used appropriately, it offers a new option for protecting investment in machine-related software and its long-term management. Organizations such as the cooperative Open Source Automation Development Lab (OSADL) eG offer a collaboration platform for interested companies and ensure the fair distribution of investments.



## Figure

Example of a "uniqueness pyramid" that shows the different levels of knowledge of a company. Above the uniqueness limit lies the knowledge that differentiates a company from its competitors; below this limit are the basic technologies that are not directly visible to the market. If this limit is at the bottom of the pyramid for a company (red line), that is, practically all the knowledge is market-relevant and confidential, FOSS should not be used. The higher the limit (yellow and green lines), the more (economically) attractive is the use of Free and Open Source Software (FOSS).