Using Open Source Software in Product Development: A Primer

Michel Ruffin and Christof Ebert, *Alcatel*

The use of open source software in industrial products is growing rapidly because its many advantages are well known.[1–4] However, although we in industry understand liability and intellectual-property-rights risks fairly well, the implications of OSS-related legal and managerial issues are new to many of us. Legal aspects can vary greatly from one piece of OSS to another depending on the license scheme used. For instance, the BSD (Berkeley Source Distribution) license[5] is very permissive, limiting the distributor's obligation to propagate the copyright, whereas the GNU[1] (a recursive acronym for "GNU's Not Unix") General Public License (GPL) comes with strong constraints on software distribution, patents, and more.

This article discusses the major legal aspects and risks in using OSS and how to mitigate them in product development. We'll answer the questions our engineers and product managers often ask when they're confronted with introducing OSS. We've ordered the information in a top-down approach, starting with a definition of OSS, its benefits, and then some risks and experiences with managing it in software engineering practice. We offer concrete examples of how Alcatel handles OSS's technical and legal aspects and mitigates our customers' and company's risks.

Naturally, such a short summary can't provide all the legal details of using OSS. Often the specific supplier and market situation and the product and engineering environment determine OSS's benefits. To avoid legal exposure, when we were in doubt about how much detail to include here, we used a rather restrictive approach. You can find detailed information at the sources provided in the references. Although we've checked the following content with Alcatel's intellectual-property experts, we strongly recommend carefully checking existing contracts or license schemes within your own specific constraints.

There are important legal aspects involved in using open source software in commercial products. Here are answers to common questions on how to handle these issues to mitigate risks.

0740-7459/04/$20.00 © 2004 IEEE

A legal perspective

Open source refers to software that you may freely use, modify, or distribute provided you observe certain restrictions with respect to copyright and protection of its open source status.[2] A major difference from freeware or software in the public domain is that OSS generally has a copyright.

Two roles are relevant in OSS licensing. The *licensor* is the OSS owner, author, or distributor and typically holds the copyright. The *licensee* is the software's user, either an end user or someone who has embedded the OSS in a product that is further distributed or licensed. We don't take much interest in the first (end user) case here, as the OSS concept implies that you can use the software as is. This is like a copyrighted book that everyone can read. However, we'll take a closer look at the second case, using OSS for developing software products, and especially consider the licensee's role.

Benefit for commercial products

A popular belief about embedding OSS into products is that it saves on license or development cost. But OSS is certainly not free software and often requires substantial investment before you can deploy it in the marketplace. For instance, using Linux in a real-time embedded environment needs not only time for your own learning curve to make it suitable and integrate it but also extensions that will require subsequent maintenance effort.

Depending on the product, its usage, and the market constraints, OSS has specific properties that can be advantageous. We've ordered the following by relevance:

- When the OSS is a de facto standard component and has a large user community, it's a lasting solution that can resist the commercial supplier uncertainties that can often abruptly end a product's life.
- Using mainstream OSS reduces cycle time for component updates and corrections. More, and often free, labor is available to localize and correct defects. Especially for embedded systems, OSS provides fast, new drivers and hardware-related features—even for rather exotic environments.
- Developing an application on a proven de facto standard (with API [application programming interface] and binary compatibility) protects the application against a change in supplier terms or conditions. With a proprietary solution, we often can't easily mitigate such commercial changes because the cost of changing suppliers—and thus the entire component with its nonstandard interfaces—is too high.
- End users expect that components, tools, and products will seamlessly interface. OSS standards such as Eclipse[6] or Corba[7] simplify the integration of products with each other and also with other suppliers' products. They ensure a longer lifetime for such APIs, because having customized, supplier-specific interfaces actually reduces mutual benefits.
- Independent software vendors distribute popular OSS solutions and components or integrate them into various packages. A global ISV market exists for many open source components from which a software house can choose to help accelerate integration efforts. This reduces the cost and risks of integrating several software components coming from various suppliers.
- Many software-based products request long-term support for each system or software release, especially embedded software. This incurs increasing product lifecycle costs due to extra-support contracts from suppliers. Because OSS makes its code available, you can substitute or rework support conditions. Other ISVs or software houses can start supporting specific OSS at any time, mitigating the risk we face when the sole supplier stops supporting its proprietary software. We've seen such evolution recently in Linux for desktop systems.
- A Mitre and US Department of Defense study concluded that OSS improves security.[3] Some fear that security is at greater risk because the source code is available, but many more people review the source code of broadly used OSS than that of proprietary software. For that very reason, security breaches are typically fixed quickly and with broad notification to the user community. Knowing and having the source code reduces the exposure to Trojan code.
- OSS is en vogue. Students use it in school, which substantially shortens their learning curve when they go to work for software companies. Engineers often have the same

Open source is certainly not free software and often requires substantial investment before you can deploy it in the marketplace.

OSS tools at home, which impacts the work climate and productivity positively.

To reduce the risks of technical or legal exposure during deployment, open source must meet several criteria.

Major risks

Major exposure occurs when you integrate OSS, usually as source code, directly into a product. If you then reproduce or sell the product without the licensor's permission, the licensor might claim for damages or force you to end the product's production, delivery, and sale. If the product license doesn't comply with the OSS license terms, this breaches the original license (one basis for current legal actions around Unix and Linux).

Another exposure is infringement of third parties' patents or other intellectual-property rights. Often OSS is a compilation of code from many sources, and it's not easy to detect whether some parts relate to protected IPR. The IPR's owner would likely seek compensation for the infringement from the OSS's licensor or licensee. As OSS license terms generally don't provide either an indemnity for such claims or a warranty stating that the OSS doesn't infringe on third-party IPR, the licensee wouldn't have any basis for regaining such damages from the licensor. Even if the licensor or licensee quickly reacts and changes the respective code, it would still remain in delivered products and is thus subject to legal action.

A third risk area is the use of OSS in development activities, when OSS is a tool. Often such software tools generate output that contains tool-created comments and a specific structure. The resulting artifacts and even the final product might, in exceptional cases, be considered a derivative work, giving the copyright holder—the tool's owner—certain rights to the resulting product.

Choosing the right OSS

To reduce the risks of technical or legal exposure during deployment, OSS must meet several criteria. It must

- Build on and follow a mature and commonly used industry standard, either a de facto standard such as Linux or a component standard such as TAO[8] or MICO Is CORBA[9]
- Have a strong OSS community
- Be broadly supported by several ISVs for distribution, evolution, and support
- Have a clear, indisputable legal status regarding IPR and the right to use it

There's one major difficulty with these criteria. Some OSS suppliers might claim they've evaluated the OSS or even that they can protect licensees, but this often isn't true. The risk lies with the licensee, as mentioned earlier. The licensee is responsible, as with all reused software, for ensuring quality and trust in all dimensions.

The packaging company's role

Packaging companies serve as intermediaries between the OSS community (with its unpredictable evolution) and the software house (which simply wants a certain component for its products). They generally provide support, stability of OSS versions, training, and documentation and can also shoulder IPR risks.

Packaging companies can also help influence OSS evolution. With proprietary software, customers can influence their suppliers' development roadmap to meet specific needs. This is not directly possible with OSS, especially if a user wants to stay aligned with a standard. However, users might be able to create and shape user groups or interest groups to implement and shape a standard. For instance, Alcatel works with its OSS distributors to contribute to the Linux and CORBA standard evolution through the Open Source Development Lab's Carrier Grade Linux workgroup[4] and the Object Management Group.[7]

Product life-cycle management

You must handle OSS carefully during the entire product life cycle. Introduce explicit checks in the decision-making processes across the product's life cycle. Such checks help to continuously and consistently disseminate information and experiences throughout the company.

Identify the OSS experts in your company who have the commercial, legal, and technical experience to handle day-to-day engineering and customer questions. The most common technical question of users who aren't using a packaging company is on the proliferation of dedicated OSS releases. Stringent configuration control is necessary. First, you must qualify a specific version and variant of the OSS; then, you must maintain control over it throughout the life cycle, accepting only official patches for defect correction. We recommend updating with a new version of the OSS only when you're planning to release a new version of your product. You must qualify your OSS components or applications as precisely and thoroughly as your own proprietary code: code reviews, analysis tools, and test coverage apply equally. Never assume a specific quality level when you download an OSS, especially in terms of security.

Other questions are of a more legal nature. Before any product development, an OSS expert should analyze for legal clearances all the external software that you'll embed in a product. If you don't have the necessary knowledge in-house, use external advisors. Maintaining an internal, centralized knowledge database on OSS—and on software that looks like OSS but isn't—facilitates and accelerates this process because it will point out the various legal aspects to study for a given piece of software. Answers differ depending on usage, markets, and customer behaviors.

Alcatel maintains a database, which all its engineers can access, containing experiences with "open source-like" software that someone in the company has studied (but not necessarily used). It provides information such as how to distribute the software, whether someone in the company has used it, who the favorite distributors are and how they behaved, how to contribute to OSS initiatives, what liability aspects to anticipate, and what end-of-life conditions to consider.

A common database such as this prevents the same questions and studies from repeatedly being done with each new component and usage scheme. In particular, smaller companies that must pay for outside expertise on OSS should build up this kind of knowledge base to avoid fragmented exercises.

Mitigating legal exposure

You can do several things to reduce your legal exposure.

- You can distribute proprietary code with GPL code (even on the same media), but you must package the code separately to avoid the GPL contamination effect (if not packaged separately, proprietary code has to become GPL). You must know how to handle packaging with GPL before deciding about using GPL software so that you can choose how to distribute your products.
- With most licensing schemes, you must distribute all copyrights from all contributors with the software. When many different contributions exist, this can be difficult—consider Linux. We often use packaging companies' services for this.
- For commercial or legal reasons, you usually can't exclude liability for distributing products containing OSS. Disclaiming statements are not valid under certain national and state laws. So, as OSS comes without any liability, its distributors must bear the entire risk without the possibility of recourse to the licensor. Packaging companies often make strong commitments regarding liability, allowing risk mitigation; this is what you pay for.
- The status of proprietary code dynamically loaded with GPL code is unclear in the GPL license. To what degree should drivers that are dynamically loaded in the Linux kernel become GPL themselves? If the value is in these proprietary drivers, the entire business case of using OSS might change. We recommend that you contact the OSS's "owner" and agree on a dedicated license scheme to clarify the legal impacts in such exceptional cases.
- Although BSD licensing has fewer constraints than GPL, it bears its own risks. Several times in the past, BSD OSS managed by a small group or company suddenly became a commercial product; then, after further evolution, it became a proprietary product. A large OSS community reduces this kind of financial risk.

The libraries used by a product that you're distributing to a client (in the form of a library to link with the client software) can conflict with the same library in the client software if the versions differ. This is not a specific OSS problem, but it occurs more often with OSS because of the popularity of some libraries, such as the GNU C libraries. There's no real solution here. We advise dedicated configuration management guidance—namely, carefully double-check and review which libraries and versions will be used in the final build, ensure that interdependencies are clean, and keep track of version changes and the propagation of corrected libraries. Regression tests can check availability of specific library contents and APIs.

Never assume a specific quality level when you download open source software, especially in terms of security.

About the Authors

Michel Ruffin is manager of software coordination at Alcatel. As part of the chief technology officer's team, he coordinates the technical procurement of software components for Alcatel products. He is also Alcatel's coordinator of standardization on software architectures and distributed applications. He chairs the OMG Telecom task force and is a steering committee member of the OSDL Carrier Grade Linux Working Group. In 2000, he received the OMG's Distinguished Service Award, and in 2001, he was nominated as a distinguished member of the Alcatel Technical Academy. He received his PhD in computer science from the University Paris VI. Contact him at Alcatel, 54 Rue La Boetie, 75 008 Paris, France; michel.ruffin@alcatel.fr.

Christof Ebert is director of software coordination and process improvement at Alcatel, where he drives R&D innovation and improvement programs. His current responsibilities include leading Alcatel's R&D to CMM Level 4. He also lectures at the University of Stuttgart, Germany, on real-time systems. He is a senior member of the IEEE and *IEEE Software*'s associate editor in chief for requirements. He received his PhD in electrical engineering from the University of Stuttgart. Contact him at Alcatel, 54 Rue La Boetie, 75008 Paris, France; christof.ebert@alcatel.com.

Any company dealing with OSS needs a few simple rules for using it in product development:

- Control the introduction and use of OSS; it must be explicitly authorized on a per-version basis.
- Disseminate technical, managerial, and legal information widely in your company.
- Systematically qualify OSS components before integrating them.

Would proprietary software give licensees more legal and commercial protection? The ISV and packaging company business is growing along with OSS, thus blending the best of proprietary software advantages with those of OSS. To benefit from the OSS processes of shared development, an alternative scenario could be to build a so-called *community source* openly shared inside a defined, closed community that follows OSS processes and experiences. This would protect OSS from the outside community. Alcatel has introduced community source for critical components over the past years.

Industry is increasingly using open source building blocks for mature technologies such as operating systems, middleware (such as CORBA and Java), databases, protocol stacks, and development environments. Mature OSS components often offer a more permanent solution than commercial ones, especially in times of economic uncertainties and small suppliers. They typically provide independence from a specific supplier, long-term support at affordable costs, low cost of ownership because of shared resources, and intrinsic effects such as greater availability of skills or engineering motivation.

Today's companies are moving up to the applications and service level, increasingly relying on reused, externally available components. OSS fulfilling "basic" needs is a natural evolution in that context. With a more mature OSS business model and value chain, it's increasingly important that software engineers become proficient regarding the advantages and constraints resulting from using specific OSS components. Following some of the suggestions we've made for managing OSS will help mitigate associated risks and take advantage of the benefits. Further information is available at the Open Source Initiative Web page at www.opensource.org.

References

1. GNU Project, *GNU's Not Unix!,* www.gnu.org.
2. Open Source Initiative, *The Open Source Definition Version 1.9*, www.opensource.org/docs/definition.html.
3. MITRE Corp., *Use of Free and Open-Source Software (FOSS) in the US Department of Defense*, www.egovos.org/pdf/dodfoss.pdf.
4. *Open Source Development Labs*, www.osdl.org.
5. *The BSD License*, www.opensource.org/licenses/bsd-license.php.
6. *Eclipse.org*, www.eclipse.org.
7. *Object Management Group*, www.omg.org.
8. *Real-Time CORBA with TAO*, www.cs.wustl.edu/~schmidt/TAO.html.
9. *MICO Is CORBA*, www.mico.org.