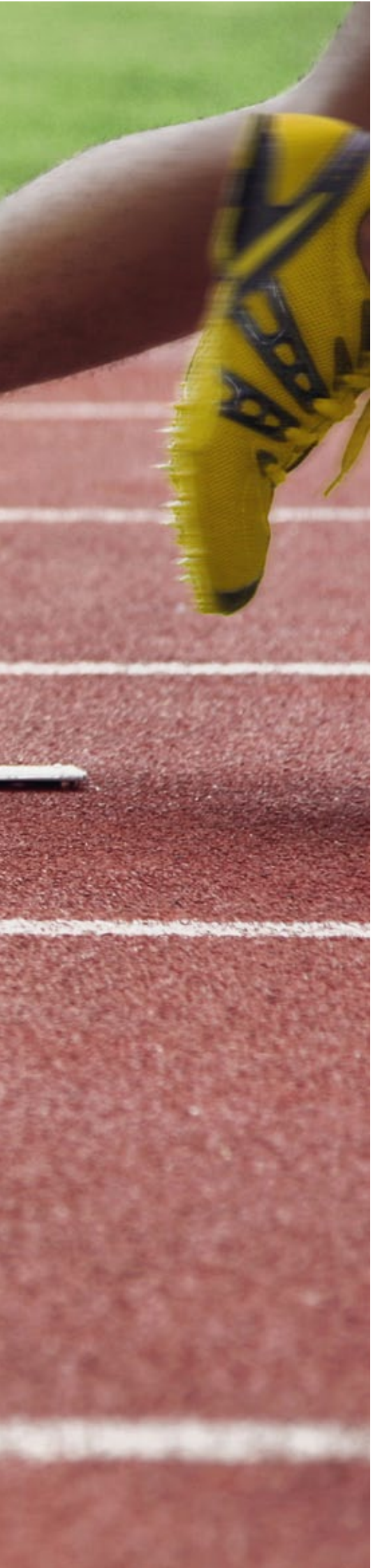
Open source software
in business-critical environments

ERNST & YOUNG
Quality In Everything We Do



Editorial

We fully expect almost all of the world's largest companies to be using open source technologies in mission-critical areas within the next few years. The question is therefore not whether companies and public institutions will use open source software in the future, but rather how they will do so.

As is the case with other fundamental decisions regarding software platforms or equipment, companies would be well advised to analyze and assess the consequences of using open source software in terms of opportunities and risks. Today, there are practical principles for both perspectives that make the controlled use of open source software even in business-critical environments an option worth considering.

Ernst & Young has prepared this publication in order to set out the benefits, risks and good practices relating to open source software and the ways in which it can be used in a professional context, as well as to give a brief look at the issue of licensing and general overview of the field of open source software. Our hope is to promote an active interest in this issue, and to contribute towards a realistic appraisal of it.

Jürg Brun
Partner
Advisory Services

Ferdinand Kobelt
Partner
Advisory Services

Reto Aeberhardt
Senior Manager
Advisory Services

Dr. Matthias Stürmer
Senior
Advisory Services

Contents

1. Executive Summary

Open source is not a question of faith

Even if the issue of open source is often emotionally charged in professional circles and misconceptions still abound, open source software has a clear definition and objective advantages and disadvantages. Software is referred to as open source if it is published under an open source license that is recognized by the Open Source Initiative. These licenses allow the software to be copied for free and the source code to be improved and distributed at will – freedoms that are not possible with proprietary software (please refer to page 20 et seq. for further information). The same software can therefore be integrated and maintained by different companies, reducing dependency on the producer of the software (referred to as "vendor lock-in"). At the same time, technical know-how is required (see section "3. Professional application of open source," page 12), and the obligations attached to the open source license must be complied with exactly (see section "4. Legal aspects of open source," page 15).

Open source software must be managed

Although open source software is rarely marketed, software subject to an open source license can now be found in every operating system and in almost every professional environment. Moreover, in many development departments, open source components and software solutions are still being used in business-critical environments without the knowledge of those responsible. Open source strategies, policies or architectural guidelines are often introduced to aid the targeted management of open source software and improve IT governance (see section "2. Benefits, risks and good practices," page 6).

Exploiting its full potential with an open source strategy

When used without proper management, open source software is rarely able to develop its full potential. A comprehensive open source strategy that is tailored to the organization concerned is therefore necessary in order to utilize unidentified opportunities, identify risks at an early stage and introduce corresponding measures. Whether using complete open source solutions or open source components in the development of software, or even when launching your own open source community, it is always advantageous to think ahead. Aspects of licensing law are just as important as software procurement, total cost analysis, evaluation and the long-term build-up of know-how (see the section "Good practice: open source strategy," page 11).

First-hand knowledge is required

In a dynamic environment such as an open source community, new software projects can quickly develop into disruptive innovations depending on the starting situation. But even established open source solutions can quickly reposition themselves on the market due to significant changes in their communities. New business models and licensing systems, as well as methods of analysis and evaluation, are being developed all the time. There are a range of good practices as this publication will show you. However, the extent to which they can be applied and implemented must always be assessed on a case-by-case basis. This is why you need first-hand knowledge from people with direct access to the important players on the open source market.

2. Benefits, risks and good practices

Cost savings are often a good reason to seriously consider using open source software. However, reducing dependency on software producers, marketing and employer branding, and the rapid distribution of open source software are important strategic advantages.

These benefits must be compared with the risks associated with the use of software in general and open source in particular. The highly complex licensing provisions must be complied with, the use of open source software monitored and support secured. As with the introduction of new proprietary software, there is an inherent migration risk associated with the migration to open source software.

The identified benefits and potential risks allow tried-and-tested measures to be derived that crystallize the benefits and minimize the risks. One well-established good practice is the development of an open source strategy as part of the overall IT strategy. This approach allows the issue to be addressed on a holistic footing with coordination between the various implementing measures.

The diagram below and the explanation that follows show the benefits and risks of using open source software, and the associated good practices.

The benefits of open source software

Cost savings

Good practices:

- Software portfolio assessment
- Total cost of ownership analysis
- Shared maintenance

Control over software and data

Good practice:

- Vendor risk assessment

Boost to your reputation

Good practice:

- Open source marketing

Rapid distribution

Good practice:

- Open source project management

The risks of open source software

Breach of open source licensing provisions

Good practices:

- Software licensing compliance
- Software development guidelines

Unmanaged use of open source software

Good practices:

- Quality assessment framework
- Enterprise architecture governance

Support not secured

Good practice:

- Support assessment

Migration to open source fails

Good practice:

- Integration strategy

Reaping the benefits: good practices

Cost savings

With the right approach, the use of open source software can save IT costs both in the short term and the long term.

On the one hand, replacing proprietary software with open source alternatives eliminates all license fees, leading to immediate savings. For example, an open source solution within a cloud environment is usually freely scalable with no extra expense. On the other hand, open source solutions do not entail any obligations regarding the choice of provider for the maintenance and expansion of the platform. This newly gained independence gives you the freedom to choose, and therefore a strong negotiating position that can save maintenance and development costs in the long term. Improvements to software are also generally available to all users, spreading the recurring costs of maintaining and expanding the software.

However, an uncoordinated approach can mean that the expense of integrating with the open source platform quickly negates or even outweighs the saved license costs. Insufficient planning can also incur subsequent costs, for example for retraining or the migration of data, as well as for unanticipated compatibility issues or a lack of acceptance on the part of users. This is why it is important to have systematic investigations and independent assessments of the projected levels of investment and cost savings carried out in advance.

Software portfolio assessment:
In order to identify potential relevant cost savings, we recommend examining whether there are any suitable open source alternatives to proprietary software solutions available on the market.

Total cost of ownership analysis:
We recommend having independent experts carry out a total cost of ownership analysis in order to comprehensively compare the alternative courses of action.

Shared maintenance:
We recommend considering whether internally developed software (or individual components) can be published as an open source project in order to share maintenance and development costs with other users.

Control over software and data

Control over the software used and internal data is a strategically important aspect of IT governance that can be improved by using open source software.

Using open source software allows you to regain a measure of independence and flexibility with regard to your IT, which these days is often managed by external providers. By releasing the source code, the way in which a software solution functions is open for all to see, and can be examined by independent third parties with regard to security and data protection. Open source software also typically uses open standards and formats, allowing data to be freely accessed via clear interfaces and saved for the long term in a digital format. And if you want to make changes to an open source solution that you are using, this can be directly implemented within the community or dealt with by an external service provider.

However, choosing the wrong open source solution can create an unintended dependency on the expertise of an open source service provider. At the end of the day, it is in the nature of software to create dependencies. It is important to minimize or at least consciously manage reliance on individual suppliers.

Vendor risk assessment:
We recommend thoroughly reviewing your dependency on the publishers of the software that you use or aim to introduce. This applies both to proprietary software and open source solutions.

Boost to your reputation

The professional management of open source software can give your company's image a boost.

The benefits of open source software are known to those in the field, but also increasingly to those in the wider world of business. Tech companies and public institutions receive positive press coverage when they release their own software under an open source license. They are lauded for their exemplary "digital citizenship," which can be communicated as part of their public relations. Public institutions are often expected to use more open source software than most as it promotes local innovation and business, and improves equality of opportunity.

It has also been shown that motivated IT workers and students have a high level of confidence in open source software. It makes companies more attractive places to work as the professional management of open source software creates challenging positions for qualified experts.

Rapid distribution

Under an open source license, software can be distributed much more rapidly than under a proprietary license.

The rapid and unimpeded distribution of open source software aids networking for the originating organization. If a software solution becomes established as the de facto standard, expansions or professional maintenance services can be offered for a charge, for example. The company or public institution that launched the open source platform and contributed materially to its development has the know-how and reputation to provide highly qualified services relating to the software.

Skillfully influencing an open source project can quickly accelerate its distribution and boost market penetration. In this way, a company can establish itself even in a saturated market by initiating and coordinating an open source project. Successful community management can add new users and developers of the open source software, aiding cooperation and supporting the distribution and usefulness of the open source platform (please refer to the section "Successful community management," page 23).

Open source marketing:
We advise companies and public institutions to professionally and broadly communicate open source activities in order to achieve the desired boost to the reputation of your organization.

Open source project management:
We advise companies and institutions to maintain strict control over open source projects that they have started, and to carry out intensive community building activities. It is important to strike a balance between control and openness using suitable community governance.

Minimizing risks: good practices

Breach of open source licensing provisions

Dealing with open source licenses is complicated, but can be managed using suitable guidelines for software developers.

The wide range of licensing provisions currently in place, varying in substantial fields or even simply in nuances, makes it a challenge to comply with all licensing regulations for the use of open source software. If, under a particular license, open source code is integrated into in-house software due to an oversight and then distributed outside the organization, this can result in the mandatory publication of the entire programming code for the in-house software. Otherwise the organization risks going to court for failure to comply with licensing laws.

This is why it is important, when acquiring, developing or distributing software, to know which are the relevant individual provisions and corresponding consequences when opting for a particular open source license.

Unrestricted use of open source software

Although open source products are free of licensing charges, their unrestricted use can lead to unexpected costs.

Because open source software can be downloaded from the internet free of charge, the procurement department is often bypassed, and open source technologies used without being assessed by decision-makers within procurement or checked against architecture specifications. Apart from the legal risk this poses, this can also result in a jumble of products and components that is hard to maintain in the long term and leads to increased costs. This is why it is important to manage the use of open source software in order to ensure compliance with IT governance guidelines and specifications regarding the company's architecture.

There are usually a range of open source alternatives available for each area of application, and these should be compared by means of a systematic evaluation. On the one hand, criteria stemming from traditional software procurement such as functionality, user-friendliness, stability and security play an important role. On the other hand, open source-specific characteristics such as the open source license, community size, developer heterogeneity, support availability, release management and legal ownership are also highly relevant when determining the quality of an open source solution.

Software licensing compliance:
When using, integrating and approving open source software, we recommend having experienced lawyers review the risks associated with the relevant licenses and what is required in order to comply with their provisions.

Quality assessment framework:
We recommend adopting a standardized approach to evaluation that is tailored to the qualities of open source software as well as your organization's requirements when selecting suitable open source solutions.

Software development guidelines:
In order to make the software developers aware of the range of requirements, we recommend creating guidelines regarding the development and integration of open source components.

Enterprise architecture governance:
In order to ensure that your company's IT architecture can be maintained in the long term, we recommend creating guidelines in addition to a target architecture with regard to open source technologies, including a whitelist of assessed open source solutions.

Support not secured

Maintenance and support for an open source product is not always guaranteed.

Although there are many hybrid forms, a fundamental distinction can be drawn between two different kinds of open source project: 1) Those led by organizations and 2) those coordinated by a decentralized group of developers (communities).

1) In the case of institutional open source projects, a company or government agency has released software under an open source license, and publishes updated versions on a regular basis. As the originators of the solution, the organization's employees have the necessary know-how to rectify defects and continue to develop the software. If the software becomes widely distributed, other companies also begin to offer services for it. The provision of support services is a typical business model for open source software. As with proprietary software, service level agreements or subscriptions are used to offer maintenance services with particular response times. You can find out more in section "3. Professional application of open source"on page 12.

2) Community-driven projects follow a different course. These are developed and coordinated by a number of volunteers. Rectifying errors, release management, documentation and other roles are often carried out decentrally by a number of different parties. In the medium term, a foundation or association is typically set up to function as a legal shell organization. The support provided by such communities is cost-efficient and usually prompt. However, no guarantees are given in the absence of service level agreements. Assessing the reliability of support therefore requires in-depth knowledge of the relevant community. One possible way of ensuring support is to hire software developers from the open source community concerned.

Support assessment:
We recommend thoroughly investigating the support and long-term development available for an open source solution, and securing it if necessary by concluding service level agreements or hiring core developers.

Migration to open source fails

Migrations to open source solutions can fail due to a lack of acceptance on the part of users, as well as technical dependencies.

As with transitions to new proprietary platforms, the migration to open source solutions represents a challenge that should not be underestimated. However, introducing open source software involves additional risks over and above the usual obstacles to migration.

On the one hand, today's proprietary platforms are often highly integrated into other, closed systems. These kinds of technical dependencies and incompatibilities, as well as inaccessible data or lacking interfaces can make it hard to integrate open source solutions. As a result, migrations to open source software can be delayed or must be abandoned entirely. One important approach to finding a solution is therefore the hybrid approach, which allows a combination of open source and proprietary software.

On the other hand, uncertainty and a lack of knowledge on the part of the end users constitute additional risks. The users are often accustomed to well-known software brands and proprietary software products, and are therefore not familiar with any open source alternatives. Open source solutions with the same level of functionality are often not well advertised or distributed. This can lead to low levels of user acceptance unless targeted countermeasures are taken to support the introduction of open source software. This makes it important to pursue an intensive information policy and raise awareness of open source software, as well as to conduct special training for power users and other key people.

Integration strategy:
We recommend developing a pragmatic and non-aggressive integration strategy involving pilot trials, hybrid approaches and intensive user training and awareness measures, among other things.

Good practice: open source strategy

An integrated open source strategy promotes long-term IT performance.

The advantages, risks and leading practices discussed here show that introducing and maintaining open source software requires the involvement of many IT functions. Progress in one area can have consequences in others. Decisions regarding migration and solutions often have long-term consequences that must be investigated and planned in order to be successful.

More and more companies and public institutions are developing an explicit open source strategy in order to ensure that all targets, regulations and measures relating to open source software are coordinated and fully integrated into the overall IT strategy. The open source strategy covers all relevant aspects relating to the introduction, use and approval of open source software.

The typical areas dealt with by an open source strategy are set out in the adjacent checklist.

Potential elements of an open source strategy

- ☐ Benefits and risks of open source software to the organization
- ☐ General guidelines and targets relating to open source software
- ☐ Targets for reducing dependency on proprietary software
- ☐ Taking open source solutions into account when procuring software
- ☐ Specifications regarding open source licenses, such as pure open source or open core
- ☐ Criteria for evaluating open source software
- ☐ Regulations governing the approval of open source software
- ☐ Training regarding open source technologies, organization, licenses, etc.
- ☐ Measures for implementing strategic goals, such as studies, pilot projects, centers of excellence, etc.

3. Professional application of open source

Open source software that is already available on the market can be used in three different ways. It can be downloaded from the internet free of charge, installed and used as is (scenario 1). Or a company or public institution can intensively build up its expertise and resources for certain open source products in order to be able to use them in critical functions (scenario 2).

Or an external open source provider can be drafted in to provide professional support for the introduction and maintenance of open source software (scenario 3). Which of these scenarios best suits your purpose must be decided on a case-by-case basis. The following table will help you reach a decision.

	1. Use with no professional support	2. Use with internal support	3. Use by external provider
Scenario	Downloading open source software from the internet for free, and using it as it is	Building up internal expertise and resources for certain open source solutions in order to be able to use them intensively in the long term	Support from an external open source provider in order to swiftly integrate and adapt existing open source software
Field of application	Non-critical functions	Critical functions and technologies that distinguish the company from the competition	Critical functions in which direct in-depth knowledge of the software must be available
Target group	Private, small and medium-sized enterprises, small schools, non-profit organizations	Large companies, government agencies, large institutions	Large companies, government agencies, large institutions
Benefits	▸ Low costs ▸ Rapid implementation	▸ High level of flexibility thanks to internal expertise ▸ No reliance on providers	▸ Direct access to the expertise of the open source developer ▸ Patches and development on a contract basis ▸ Selection of various open source providers ▸ Other benefits in accordance with the service level agreement (see section on "Service level agreements for open source solutions," page 13)
Disadvantages	▸ No guaranteed support ▸ No claims under liability	▸ High level of investment and time required to build up expertise ▸ High fixed internal costs due to increased staffing levels ▸ No certification for hardware and software	▸ External costs due to open source provider ▸ Know-how dependency on open source provider
Risk and hedging	High risk: No support agreements or guarantees	Medium risk: Support depends on the expertise and availability of internal IT	Low risk: Warranty provided in accordance with the terms of the order or service level agreement

Open source providers

Open source providers are companies that have launched an open source solution themselves or at least played a significant role in its development. This can be assessed by clarifying whether the company's software developers have "committer" status. This means that these programmers are able to independently amend the centrally administered source code. It indicates that the company has the necessary know-how to maintain solutions with regular security updates and patches, and to adjust them to meet changing needs. Open source companies offer expert support for their products, implement software launches and migrations, develop upgrades and expansions and conduct employee training. In particular, open source providers guarantee support for a particular software version for a binding period of time via support agreements. This allows the stable operation and long-term maintenance of the open source solution. These and other services allow such companies to continue to develop the open source software, thus ensuring the longevity of the open source project.

Service level agreements for open source solutions

The aspect of security offered by open source providers is an important one. Clearly defined service level agreements and subscriptions give open source solutions the guarantees and hedging of risks required for use in a company's critical functions. The following checklist details the possible services that can be covered by service level agreements with open source providers.

Possible services offered by open source providers

- ☐ Direct access to the expertise of the core developers of the open source software
- ☐ Defined response times for support enquiries
- ☐ Support via a number of channels (internet, VPN, email, chat, telephone, remote desktop, on-site)
- ☐ Provision of professional documentation, training and certification courses
- ☐ Prompt, proactive and user-friendly delivery of security patches
- ☐ Minimum periods of maintenance and support for particular software versions
- ☐ Guaranteed, regular software releases and updates
- ☐ Guaranteed compatibility with other software solutions
- ☐ Certification for particular hardware and proprietary software systems
- ☐ Integration of patches and expansions into the main version
- ☐ Hedging against legal claims based on intellectual property rights (copyright, patents)
- ☐ Liability for disruption and malfunction
- ☐ Provision of additional proprietary expansions and support tools

Open source versus open core

A number of hybrid business models have been established in recent years that combine aspects of open source and proprietary licensing in different ways. One such hybrid model is open core, which mainly differs from traditional proprietary licensing models in the degree of "mixing" and approval of open source software.

In an open source model, the source code for the entire software solution is published, or at least made available to customers, using an official open source license. This means that even once a service level agreement has expired, the customer can continue to use the software solution without restriction and for any purpose. In terms of business models, the open source provider has a number of service options that add value for the customer. These include advice, integration, training, development, maintenance, guarantees, operations and long-term support. At the same time, reliance on a provider ("vendor lock-in") is kept to a minimum as the customer can legally change open source provider at any time. For widely distributed open source software, the open source model allows the customer to change to a different, competent provider at little cost. Successfully changing your open source provider requires that the new provider be very familiar with the software solution, and even have its own developers that can feed changes to the software back into the open source community (see previous section "Open source providers," page 13). But even in the case of rarely used open source products or tailored open source solutions, getting a new open source provider up to speed is usually much more cost-effective than restarting the procurement and migration processes from scratch and retraining employees.

Under the open core model, the producer only publishes part of the software under an open source license, and places the rest under a proprietary license for which it sells corresponding licensing agreements. The proprietary elements usually consist of important functions for the company's operations. The open source version is typically referred to as the "community edition" or similar, with the "enterprise edition" being the version with proprietary software. This business model is essentially identical to other proprietary models and does not meet any commonly accepted definition of open source software. Open core therefore denies customers the main benefits of open source such as independence and openness. This usually makes changing service provider impossible without migrating to a different software solution.

	Open source model	**Open core model**
Community version of the software solution	The complete software source code is published under an open source license, and external contributions can be integrated without impediment.	The core of the software is published under an official open source license, but critical functions and expansions are missing.
Enterprise version of the software solution	A chargeable service level agreement is concluded with a producer for this commercial open source software, but the software source code is available in its entirety under an open source license.	The software is acquired for a fee under a proprietary license, with no access to the source code or any of the rights associated with open source software.

4. Legal aspects of open source

Key issues with regard to open source

In practice, open source often raises a variety of legal questions. What are the differences between the different open source licenses? What legal conditions and consequences are associated with the use of open source software? What effect does open source software have on license management? What kind of license is suitable for publishing your own software?

Answering these questions is key to the professional application and long-term development of open source software. The following section therefore examines the impact of copyright on open source software, gives an overview of the most important open source licenses, provides detailed explanations of delimitations and distinctions, explains the term "dual licensing" and discusses the prevalence of different open source licenses.

Open source and copyright

The term "open source" is not mentioned in law, and only refers to the fact that a program's source code is freely accessible. However, open source software is subject to all provisions of the law without exception. This also applies to copyright. Computer software is generally protected by copyright. The widely held assumption that authors of open source software waive copyright is wrong.

Copyright grants those holding it an exclusive right to use the software, and with it the right to prohibit or allow its use by others. Contractual permission to use copyrighted software is referred to as a "license." Issuing a license is generally associated with rights and responsibilities. In the case of proprietary software, there is usually an obligation to pay a license fee. The copyright holder is at liberty to make granting a license subject to other obligations and conditions. This is typically the case for open source licenses, as they usually contain obligations or conditions regarding one of the following areas:

- Changes to the source code
- Incorporation of the source code into third-party software
- Obligation to publish the changed source code
- Further distribution of the changed source code

Overview of the most important open source licenses

There are many different kinds of open source licenses that differ in terms of their licensing conditions. It is therefore important for users of open source software to know what kind of license the software was published under and what obligations must be met as a result. The original wording of the licenses described here and of many others can be found on the Open Source Initiative's website.

GNU General Public License (GPL)

The GNU General Public License (GPL) is the most common type of open source license. It was first issued in 1989 by the Free Software Foundation, and updated in 1991 and 2007 with versions 2 and 3, respectively. The GPL grants four freedoms. It explicitly allows the unrestricted use of the software and its reproduction free of charge, free access to the source code and the distribution of any changed versions. The GPL is unique because of its viral effect. This means that changed GPL components may only be distributed if their source code is made available and also subjected to a GPL. On the one hand, this mandatory publication represents a "strong safeguard" of the freedom that keeps subsequent versions of software published under a GPL freely available forever. On the other hand, this mechanism (also referred to as "copyleft") represents a legal risk to the company as software developed by the company and combined with GPL code must be published together with its source code in accordance with the GPL. In this respect, the GPL differs from other licenses on account of the fact that it relates to the software as a whole and not just individual components. Its scope includes functional as well as some economic aspects. This makes the GPL "stricter" and its safeguarding of freedoms "stronger."

GNU Library or Lesser General Public License (LGPL)

The GNU Library or Lesser General Public License (LGPL) was also published in 1991 by the Free Software Foundation at the same time as version 2 of the GPL. It was revised slightly in 1999 as version 2.1, and version 3 was then released in 2007 at the same time as the GPL. The LGPL was developed for the use of libraries, a practice that is widespread in the world of software development. These software components are explicitly intended for repeated use in software applications. In order to avoid forcing the publication of the entire source code for the "shell" application, as is the case with the GPL, components licensed under the LGPL may continue to be used in proprietary software. However, one similarity to the GPL is that any changes made to the LGPL library itself must be published under an LGPL. This is why the LGPL is referred to as a "weak safeguard" of freedoms.

GNU Affero General Public License (AGPL)

The GNU Affero General Public License (AGPL) was last published as version 3 by the Free Software Foundation in 2007 in collaboration with the company Affero. The license is based on GPL version 3, and aims to close a "freedom loophole" for application service providers (ASP). This arises from the fact that when software licensed under a GPL is operated on a server, users have access to the application but not to the executable files. The ASP is therefore not obliged to publish its changes to the GPL software because it is not physically distributed to customers. This is not the case for software licensed under an AGPL, which also requires the source code to be disclosed in the case of use via a network connection. Because of this characteristic, the AGPL could become increasingly important in the future for concepts such as cloud computing and software as a service.

MIT-, BSD- und Apache Licenses

Although "liberal" open source licenses such as the MIT License, BSD License and Apache License originate from different open source communities, they differ only in the details. In contrast to the licenses of the Free Software Foundation already described, the liberal licenses allow the source code to be directly incorporated into proprietary software. The only condition is often that mention be made of the copyright, or that the text of the license be supplied, in order to ensure that the open source software libraries are mentioned by name as stand-alone components of the proprietary software.

Mozilla Public License (MPL)

Similarly to the LGPL, the Mozilla Public License (MPL) is a weak safeguard of freedoms. The MPL is the license that is most consistently file-based. However, as one of the less liberal licenses, it is incompatible with the GPL. This means that code licensed under an MPL cannot be integrated into GPL software.

Definition of other categories of software

When discussing open source, other terms such as public domain software, freeware and shareware often crop up, which can lead to misconceptions. It is not always easy to distinguish between them as they do not have any generally binding definitions. The following definitions cover some of the main characteristics of each term.

Public domain software

There are no obligations or conditions tied to the use of public domain software. The term does not indicate whether the software's source code is freely accessible. This can be, but is not necessarily the case.

Freeware

Freeware may be used free of charge or in return for a voluntary payment. However, use can be made subject to obligations and conditions. The term does not indicate whether the software's source code is freely accessible. This can be, but is not necessarily the case.

Shareware

Shareware usually refers to proprietary software that can be used free of charge with limited functionality for the purpose of testing. The source code of shareware is not usually freely accessible.

Distinctions between the most important open source licenses

The table below gives an overview of the characteristics of the licenses mentioned.

They tend to be less restrictive towards the top and towards the right of the table.

	Strong protection of freedoms		**Weak protection of freedoms**	**"Liberal" open source licenses**	
		GPL (v2 und v3)	*LGPLv3*	*Apache License 2.0*	MIT License and BSD License
Source code freely accessible	**yes**	**yes**	**yes**	**yes**	**yes**
Source code may be changed and combined at will with other software within the same legal entity.	**yes**	**yes**	**yes**	**yes**	**yes**
Source code may be kept on web servers where it is inaccessible: If the software is not physically distributed to customers or partners, but is instead exclusively made available to users on a server, for example, then there is no obligation to publish the source code.	**no**	**yes**	**yes**	**yes**	**yes**
Source code may be distributed with proprietary software: Provided LGPL-licensed software is exclusively used externally, for example as a software library, it may be distributed together with proprietary software.	**no**	**no**	**yes**	**yes**	**yes**
Changes may remain inaccessible: The main difference between liberal licenses and the licenses of the Free Software Foundation (AGPL, GPL and LGPL) is that they allow the source code to be closely integrated into proprietary software. Improvements and additions to the source code therefore no longer have to be released, but can be kept inaccessible.	**no**	**no**	**no**	**yes**	**yes**
The only obligation is to include a copyright notice and exclusion of liability in the source code.	**no**	**no**	**no**	**no**	**yes**

License compatibility (see next section for more details)

Interaction between open source licenses: compatibility and dual licensing

The table on page 17 can be used to determine how compatible open source licenses are with each other. For example, software with an Apache license can be integrated into GPL software but not vice versa, because the GPL is more restrictive than the Apache license.

In the case of dual licensing, a producer publishes the same software under two different licenses. A restrictive license (particularly a GPL) and a proprietary license are typically used. This gives the producer the best of both worlds. On the one hand, the software enjoys more rapid distribution thanks to the free open source version. On the other hand, selling the software under a proprietary license generates revenue. Meanwhile, the proprietary license is in the customers' interests if they wish to use the software within their own proprietary software.

Successful examples of such dual licensing models can often be found in the field of databases. The database producer offers its software under the two licenses previously mentioned. This gives the platform a high number of users, which aids testing and the search for errors and in doing so ultimately boosts stability. If a company in the field of business applications wishes to integrate the database into one of its own software products, it can acquire a proprietary license from the producer of the database. This gives the producer of the application a legal way to bypass the restrictions of a GPL license, integrate the database into its own solution and sell said solution under a new proprietary license.

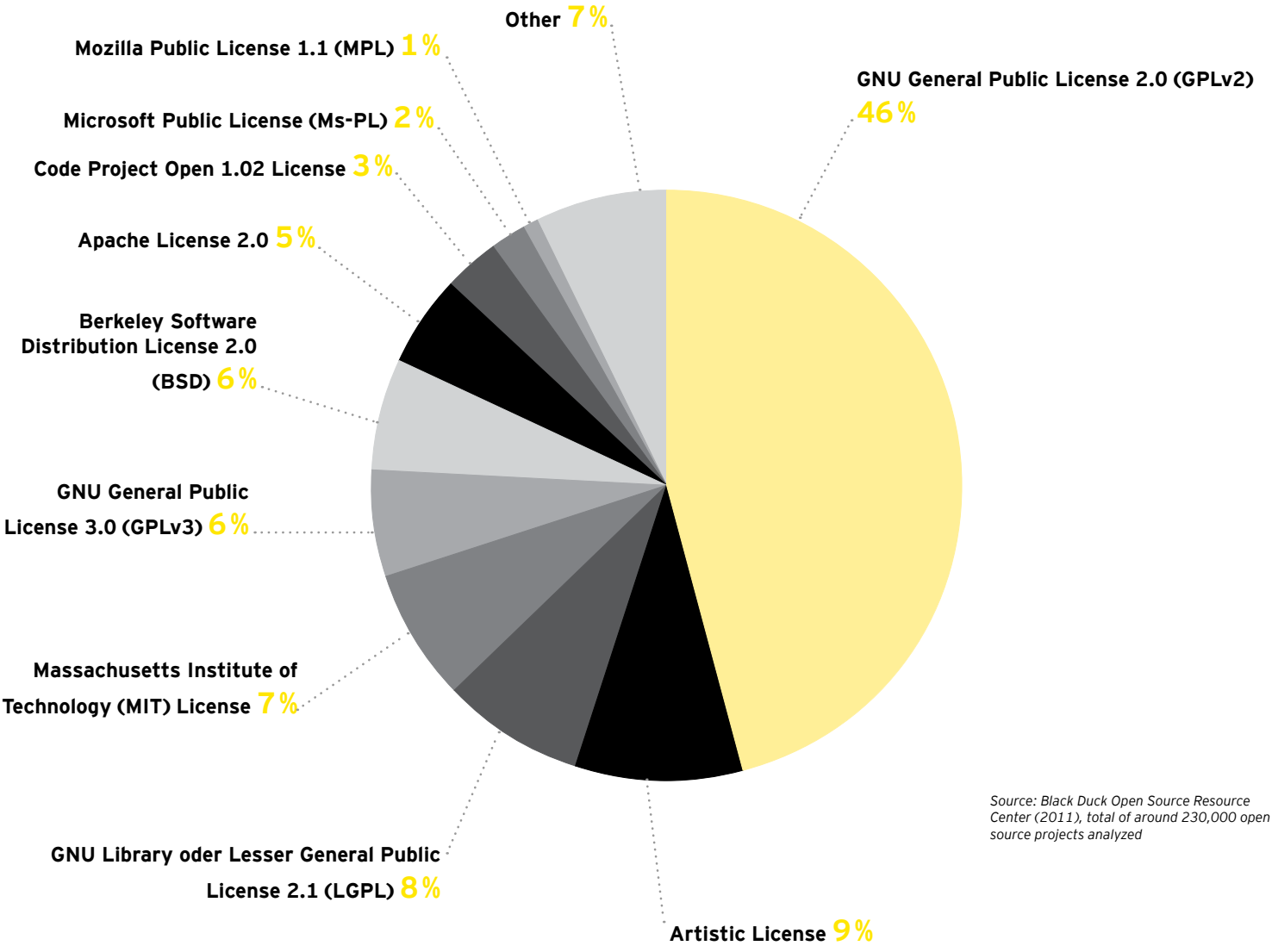
License compliance checklist for open source software

- ☐ We know whether open source software is used at our organization
- ☐ We know what license the open source software concerned is subject to
- ☐ We have identified whether and when we can use GPL software in development
- ☐ Our programmers are aware of the consequences of integrating GPL components into our software
- ☐ When we release our own software, we carefully consider which open source license to use
- ☐ Regular monitoring ensures that our internal regulations governing the use of open source software are adhered to

Prevalence of open source licenses

According to Black Duck Software Inc., a company specializing in open source compliance management, over half of the approximately 230,000 open source projects in existence today use the GNU General Public License Version 2 (GPLv2). Around one in ten projects are published under the artistic license typical for Perl software, and about the same number use the LGPL designed for software components. The MIT license is in fourth place with around 7%, while version 3 of the GPL comes in fifth but is gaining ground. Other well-known open source licenses include the BSD License, the Apache License and the Mozilla Public License.

Although they only make up a small percentage of the total number of licensed software projects, many commonly used open source solutions such as the Apache web server or the Mozilla Firefox web browser are published under these "liberal" licenses. Recent years have seen an increase in the number of open source projects based on the Microsoft Public License, as this also promotes the development of open source software by producers better known for their proprietary software.



Source: Black Duck Open Source Resource Center (2011), total of around 230,000 open source projects analyzed

5. Background information on open source

A look at software development

Understanding the qualities of open source software requires a brief introduction to the field of software development. Software is usually developed in the form of a source text that can be read by people. Most programming languages use a "compiler" to convert this source code into high-performance binary code that can only be read by processors. The operating system, standard programs and specialist applications that run on a desktop computer or server is made up of executable files such as these.

Dependency on the producers of proprietary software

In the software industry, user licenses are generally sold for software. Customers are only provided with the binary code with a number of restrictions on how it may be used. The software's real "bricks and mortar," the source code, remains the restricted property of the software company. The user is therefore dependent on the company and its strategy for how and whether the software product should develop in the future. This is not the case with open source software, where both the binary code and the corresponding source code are provided, and the reproduction and amendment of the software is explicitly allowed. It is important to remember, however, that the producer of the software retains copyright, allowing it to also sell open source software under a traditional license. This is referred to as dual licensing, as already described.

Definition of open source software

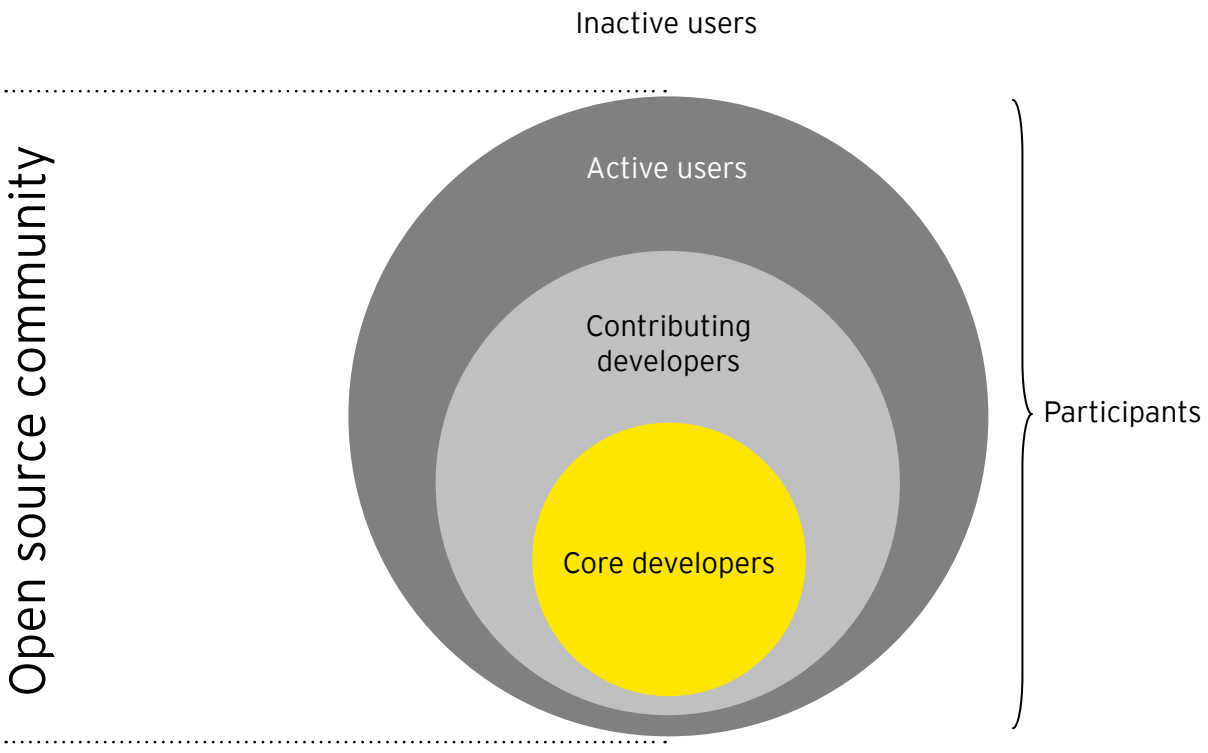
A software product is referred to as open source software if it has been published under one of the around 70 licenses approved by the Open Source Initiative (OSI, www.opensource.org) (see section "4. Legal aspects of open source," page 15). Open source software therefore defines the key characteristics of the respective software license rather than a particular technology or business model. All official open source licenses prescribe the following characteristics for software published using them:

1. The software's source text must be comprehensible for people.
2. The software may be copied, distributed and used as desired.
3. The software may be amended and distributed in its amended form.

Open source projects and communities

If a software solution is published under an open source license, it is often referred to as an open source "project." This indicates that apart from the software itself, the open source development model depends on the formation of an active community to develop the source code according to its own rules and structures. It is worth noting that an open source project does not have a start or end date, and is referred to as active for as long as the software continues to be developed.

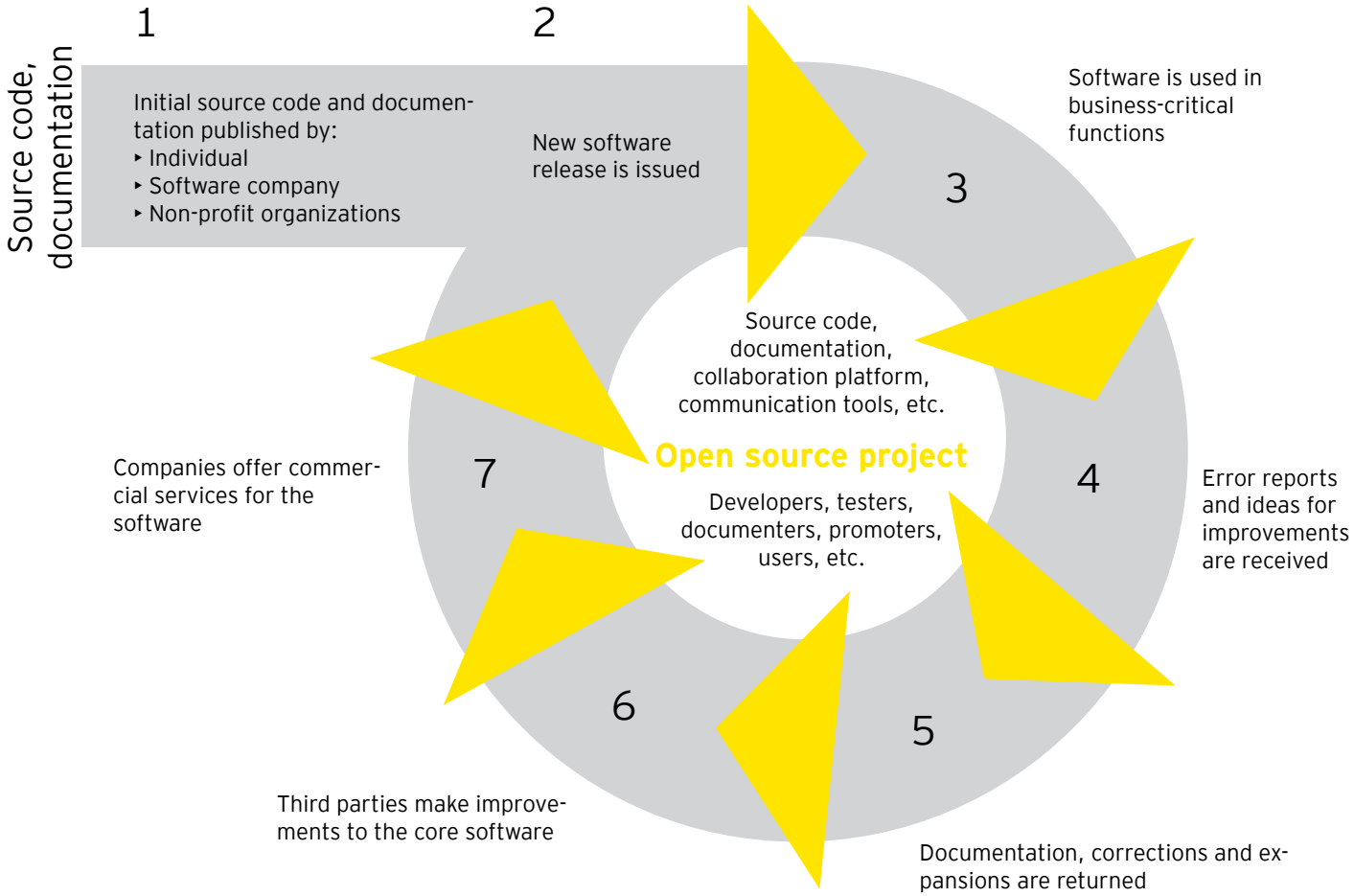
Open source communities typically consist of a number of layers. At the center are the core developers, often a small group of between 10 and 20 software developers that have programmed the majority of the software and push forward the underlying development. They have committer rights for the open source project, and can therefore change any aspect of the source code at any time. The next layer comprises the contributing software developers. They are responsible for particular peripheral software components, program new expansions or correct errors and security gaps in the source code. These people can usually change small sections of the central source code. Even further out we find the active users, who use the open source solution for themselves while at the same time playing an active role in the community, for example by contributing documentation, translations, reporting on errors or responding to requests for support. These users do not generally have authorization to access the central repository in order to change the source code contained therein. Outside of the open source community per se are the inactive users, who use the software without making any contributions or giving feedback. These people are also unable to alter the source code in the repository, and are limited to downloading the software.



Community building process

The following cycle shows the typical, recurring stages of the community building process for an open source project:

Every open source project begins with the publication of the initial source code by an individual, a software company, public institution or other non-profit organization. Active open source projects only issue new software releases within short cycles in order to improve both stability and functionality. If the open source solution is widely used and applied to companies' critical functions, users often send requests for improvements and error reports to those responsible for the open source project. As the community grows, users begin to help improve the documentation or even the quality and functionality of the open source software by rectifying errors and expanding applications. Once external developers have become more familiar with the source code, they can start to contribute to the core software. As demand for support, maintenance and development grows, software companies generally appear offering commercial services for that open source product. These additional developers and companies add to the open source community, thus contributing towards the longevity of the open source software. However, a community can become divided as described in the following section, "Successful community management," on page 23.



Successful community management

Open source projects are always launched by an individual, a software company or non-profit organization such as a public institution or research department. If the community building process is successful, the number of users and developers grows and the software is constantly developed. The initiator must then decide whether they would like to retain full control over the open source project or whether they would like to involve other people and organizations in the decision-making processes.

If the initiator cedes more influence to others, they can expect the software to be more widely distributed with more third-party improvements and expansions to the source code. However, they lose some control in the process, as the other stakeholders can also have their say regarding technical or organizational issues in the future.

If the initiator largely retains control, they alone can shape the ongoing development of the software in order to implement their business strategy unimpeded. However, this comes at a cost to acceptance on the part of users and developers, increasing the risk that third-party investment and contributions will always remain low. If the software is very widely used and the community believes that the initiator is repeatedly making serious errors, the community can split, which is referred to as a "fork." Thanks to the open source license, disenchanted people and organizations can copy the source code and develop it independently on a new collaboration platform under a new name. Whether the fork is successful in the long term, the original open source project survives or perhaps even the two develop in parallel depends on the situation.

Checklist for successful community management

- ☐ Care is taken to ensure transparency and comprehensible decision-making processes
- ☐ Important contributors from the community are included in decision-making processes
- ☐ Thought has been given to setting up an independent foundation or association to best include all major stakeholders in the management of the community
- ☐ Even critical voices from within the community are treated with respect

Open source versus free software

Although these terms are often used as synonyms in IT, and are in fact identical from a technical perspective, the difference between open source software and free software is important to those in the know, and is often the subject of heated debate. This is due in part to historical reasons, as the term "free software" was introduced in 1983 while open source did not appear until 1998. On the other hand, the two approaches have a somewhat different focus as explained in the following.

Please note: this publication exclusively uses the term "open source." This is done in the knowledge that free software provided the roots and underlying concepts behind the entire trend.

Free software as a synonym for free knowledge

Free software emphasizes the guaranteed, long-term availability of the source code in order to keep the knowledge freely available in the interest of its benefit to society. The wordplay "copyleft" was introduced in this regard. This term requires that amended and distributed versions of the source code must be made freely available themselves. The widely distributed GNU General Public License (GPL) includes a mechanism requiring software, once published under a GPL, to always be made available under a GPL. This is why the GPL is often referred to as a "viral" license because it can "infect" other software components. In the eyes of the Free Software Foundation, which coined the term free software and also published the GPL, only this viral effect can safeguard the freedoms of software users.

Open source as a synonym for openness

The concept behind open source software focuses on the open source code and the many uses to which it can be put. This approach was consciously chosen at the end of the 1990s as a pragmatic response to the term "free software," which was not seen as a suitable starting position for business use in terms of its meaning. The Open Source Initiative which was founded around the same time compiled a list of ten criteria that borrows heavily from the definition of free software. However, the term "open source" explicitly covers licenses that allow open source software to be reused in proprietary software.

Brief historical overview

The following events, grouped into three periods, show stages in the development and spread of open source software.

The pioneer period

1985 Richard Stallman establishes the Free Software Foundation (FSF) and coins the term "free software"
1989 The FSF publishes version 1 and, in 1991, version 2 of the GNU General Public License (GPL)
1991 Linux Torvalds launches the development of the Linux core under the GPL version 2
1993 Linux distributor Debian is founded

The business period

1998 Eric Raymond, Bruce Perens and Tim O'Reilly establish the Open Source Initiative (OSI) and coin the term "open source"
1999 Netscape is the first major company to release the source code for its Netscape Navigator, which had previously been proprietary, under an open source license
1999 Linux service provider Red Hat goes public on the NASDAQ
2000 IBM announces that it is to invest one billion US dollars in the development of Linux
2001 IBM publishes the software development platform Eclipse, with an estimated value of USD 40 million, under an open source license

The mainstream period

2004 Canonical launches Ubuntu, a Linux desktop aimed at end users that today is used by around 12 million people
2007 The Free Software Foundation publishes version 3 of the GNU General Public License (GPL)
2007 Sun Microsystems publishes the Java Development Kit under a GPL, and names the software platform OpenJDK
2008 Google launches the Linux-based Android operating system for smartphones, which is already used in around 20 million devices by 2011
2008 The French Gendarmerie migrates 70,000 desktops from Microsoft to the Linux distribution Ubuntu
2009 Linux is installed on more than 300,000 virtual workspaces at Brazilian schools
2011 In the Spanish region of Andalusia, Linux's own distribution Guadalinux is installed on around 500,000 school computers
2011 Insurer LVM migrates 10,000 workplaces to the Linux distribution Ubuntu

Common misconceptions

"There is no support for open source software"

Not true. As seen in the historical overview, an active and innovative business landscape of providers arose for open source software towards the end of the 1990s, some of which were responsible for major open source launches. Today, there are software companies offering service level agreements for open source solutions that are identical to those for proprietary software (see section "3. Professional application of open source," page 12).

"The legal situation concerning open source software is ambiguous"

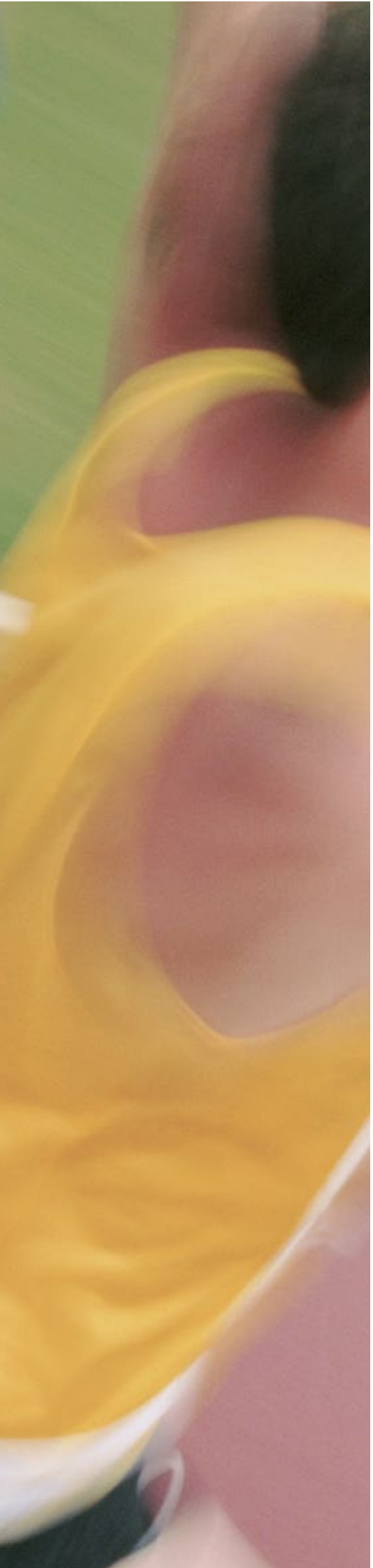
Not true. Open source software is always released under a clearly defined license that has been certified by the Open Source Initiative (OSI) and specifies how the software may or may not be used. An individual expert report should be obtained for specific questions relating to licensing laws in order to ensure compatibility between licenses and compliance with their provisions (see section "4. Legal aspects of open source," page 15).

"Open source software is free"

Simply downloading open source software does indeed cost nothing. However, for professional use, service level agreements or subscriptions are required to ensure reliable development, maintenance and support (see section "3. Professional application of open source," page 12). The key issue is that the source code of open source software is freely accessible and can be changed or reproduced. This reduces dependency on the producer of the software, saving costs in the long term.

"Open source software is not widespread"

Not true. Open source software has many applications, and can now be found on millions of desktops, servers, mobile devices and embedded systems. Even major companies and banks use open source in critical functions. As these products are no single company's property, no serious effort is made to provide information regarding the use of open source software. As a result, actual usage levels of open source software are often underestimated. Many well-known internet companies would never have been as successful as they are without the large-scale use of open source software.



6. Conclusion

Open source software has become an important aspect of computing. When used in the right way, open source solutions can lead to substantial savings in both the short and the long term. Open source also contributes towards strategic advantages such as independence, security and transparency. This boosts investment confidence and digital sustainability.

However, managing open source software requires know-how and experience in order to be able to make the most of its advantages and successfully avoid its risks. The challenges of using and distributing open source software can be overcome with the help of targeted measures. The good practices discussed in this brochure will help you learn from the tried-and-tested approaches of other companies and public institutions, and to realize the full potential of open source technologies through proper management.

Ernst & Young and its qualified Open Source Advisory Team are happy to help you with the implementation of good practices, licensing issues and other issues relating to open source software for business-critical environments.

Ernst & Young

Assurance | Tax | Legal |
Transactions | Advisory

About Ernst & Young
Ernst & Young is a global leader in assurance, tax, transaction and advisory services. Worldwide, our 141,000 people are united by our shared values and an unwavering commitment to quality. We make a difference by helping our people, our clients and our wider communities achieve their potential.

Ernst & Young refers to the global organization of member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients. For more information about our organization, please visit www.ey.com.

In Switzerland, Ernst & Young is a leading accounting and advisory firm, employing more than 2,000 people at 10 locations, which is also offering services in the fields of tax, legal and transaction.

Contact

Advisory Services		
Jürg Brun (Partner)	+41 58 289 32 03	juerg.brun@ch.ey.com
Ferdinand Kobelt (Partner)	+41 58 289 69 31	ferdinand.kobelt@ch.ey.com
Reto Aeberhardt (Senior Manager)	+41 58 289 67 40	reto.aeberhardt@ch.ey.com
Dr. Matthias Stürmer (Senior)	+41 58 289 61 97	matthias.stuermer@ch.ey.com