Open Source Software

Note: this paper was written collectively by Partners of the NESSI ETP. However, any statement should not be considered the views of any specific NESSI Partner, or attributed to them, unless explicitly stated by that Partner.

Introduction

Open source[1] is a way of developing and delivering software (OSS) which can be modified and shared under a certain license of distribution and/or use. Open source means you have access to the source code (description) compared to closed software (proprietary assets) where you mostly get the binaries (products). The term does not refer to or imply anything about functionality, applicability, (connected) fees, legality, quality, or maintainability. Using OSS does not necessarily come without cost, it is not always free[2], and it does not always fit into all business models. The nature of most OSS is that it serves some general purpose, such as a graphical library, a set of machine learning algorithms, data management tools, etc. Hence, OSS is usually not developed for specific solutions, but can be used across application domains and sectors.

Open source-based software gives a good baseline to avoid starting a development from scratch or to develop some functionality by yourself. Most open source communities encourage the modifications and improvements of their software by its users, publishing back the source code under the same license. But it is necessary to know how to deal with it. The license terms must be followed, and it may not fit completely to your needs. Legal checks are required to avoid infringement of license terms, because license regimes are many, and some can even lock business operation if a company does not check the terms carefully. Before using the code, the quality must be assessed - though this requires appropriate expertise, and may be challenging when the code is large or complex. There may be similar issues with proprietary software: there is a similar need for ensuring license compliance, and for checking that the software fits the purpose, but code quality cannot be assessed, unless the code provider grants the necessary access.

There are often more options for maintenance of OSS compared to proprietary software. Maintenance of proprietary software usually requires a contract with the supplier, whereas with OSS potential choices include building up know-how internally, contracting a third party, or relying on an active community.

NESSI[3] is comprised of many diverse organisations with different views on open source. Therefore, this paper does not give a single coherent position about OSS, but explores the broad spectrum of positions.

The vision about open source must consider both OSS communities and the business interests of software companies, in such a way that everyone may win. Open source and proprietary software are non-exclusive options, and in many situations we see a combination of both. For example, it is quite frequent that an IT company uses OSS for basic and general-purpose components (operating system, cloud infrastructure, middleware functionality, etc.) and leverages on top of them high-value specific proprietary or open source solutions. This is especially the case for integrators, whose main business is to provide an end-to-end solution to the customer at the best price, minimizing vendor lock-in and covering all the customer's needs with the best available technology, either open or proprietary. The OSS community wins as its goal is to spread the use of software and evolving it thanks to massive use; and the companies win as they can concentrate on added-value development, while relying on OSS for their baseline functionality.

[1] https://opensource.com/resources/what-open-source.

[2] Open source software is often called "Free software" where free is related to the freedom allowed by the licenses and not related to the costs or fees associated with the software.

[3] NESSI (Networked European Software and Services Initiative), the European Technology Platform (ETP) dedicated to software, data and services; http://www.nessi.eu/.

The main drivers of open source are:

- fostering the extensive reusability of code;
- democratizing access to essential and basic software for everyone, by removing barriers to access technology, empowering people;
- speeding up ICT innovation by allowing others to build up on prior work (this is very similar to the process of science, where researchers base their work on earlier findings);
- sharing knowledge and practices to speed up software development (open and proprietary);
- fostering interoperability by using standard and open technologies (when more and more software systems are built on many different technologies); and
- optimizing the cost of an ICT solution.

This paper is addressed to:

- policy makers, to be aware of the relevance of open source, not only for fostering businesses but also to make access to software more democratic and by that to promote digital inclusion
- NESSI Members, to promulgate a common understanding that NESSI is a plural community and does not champion one particular position on this issue; and
- all external stakeholders, to communicate the position of NESSI on open source.

The paper shows the relevance of open source in the current software context from the NESSI perspective, then it sets out the position of NESSI with regards to main features of open source, and continues with the intellectual property and business considerations of open source use and development. The paper is focused on open source software; open data and open services are outside of the scope of this paper.

OSS ecosystems: essential innovation engines for the software industry

It is evident that innovative software solutions will be more and more the result of a continuous interaction among companies and open source communities, and especially those communities whose interest in software is motivated by the digitalisation of society. This fruitful symbiosis is easily demonstrated by recent trends in software engineering, which are clearly influenced by OSS communities. The software architecture of modern systems has been inspired/promoted by open source projects. Open source operating systems are now the most used for high-end computers and for embedded devices like smartphones, TVs, routers, etc. Commercial software companies are necessarily considering these facts in designing their strategies.

Open source business models

There are many ways in which the different actors in the OSS arena generate revenue and gain benefits from OSS. The key players in the OSS field are foundations, distributors, integrators, cloud actors and software companies which are using OSS to build an ecosystem around their products and services, as well as individual developers contributing to OSS projects as employees of an organization or as contributors motivated to develop technical expertise or to gain recognition from the OSS community.

OSS foundations are usually non-profit organisations which provide the legal and economic platform for open and collaborative software development[4]. They are financed by membership fees, donations, or even crowdfunding or crowdsourcing campaigns. Some of the most well-known foundations are:

[4] e.g. https://livablesoftware.com/study-open-source-foundations/

- The Free Software Foundation (https://www.fsf.org/), which funds the GNU project, has defined the GNU GPL license, and provides educational material about open source software development. It is funded mainly by donations.
- The Apache Software Foundation (http://apache.org/), which supports projects such as the Apache HTTP server, Perl, and Tomcat, and the Apache License. It is funded by donations and sponsorships (platinum $125k/year, gold $50k/year, silver $25k/year, and bronze $6k/year).
- The Eclipse foundation (http://www.eclipse.org/org/foundation/) is known for the Eclipse IDE. The projects are licensed under the EPL license. It is financed by sponsorships and yearly membership fees ranging between $250k and $5k.

Distributors focus on packaging OSS to end users. They earn revenue by providing customer support and professional services such as consulting, training, and maintenance. For example, RedHat's business model is subscription-based, offering different levels of support and trainings for its Linux distributions. RedHat's revenue in 2018 reached $3.36 billion. Beside the commercial distributors, there are also non-profit distributors such as CentOS or Debian. They are financed mainly by donations.

Integrators provide customized system solutions and services. They use and integrate OSS components, but also contribute to OSS projects with changes that ease integration. In this way, they can save on software license costs and improve the overall cost position of their products and service offerings.[5]

Hosting an OSS project in the cloud and making it available as a service provides another way of monetizing OSS, i.e. through customers paying for using the service. Examples are Databricks offering a service based on Apache Sparks (https://databricks.com/) and Acquia offering a content management service based on Drupal (https://www.acquia.com). In some cases, when service providers are generating enough revenue from advertising or data monetization, even the service is provided for free. For example, WordPress.com is very successful in providing the WordPress Content Management System as a Service for its customers.

Some companies are developing and publishing OSS to promote closely-related commercial software offerings. There are many licensing models supporting this kind of business model. Most popular are dual licensing models and license schemes supporting the open core model. In case of dual licensing, a company releases software under two different licenses, a copyleft license like GPL and a proprietary license the software user has to pay for. The copyleft license stipulates that anyone who modifies or integrates the software has to distribute the resulting software freely under the same terms and conditions. This means that anybody who integrated this OSS into a commercial software would violate the copyleft license, and therefore has to pay for the proprietary license that allows commercial use. MySQL is a well-known example of applying dual licensing (https://www.mysql.com/de/about/legal/licensing/oem/). In the open core model, a 'core' part of the software is released under an OSS license and an additional part, usually comprising features that are essential for commercial use of the software, is released under a proprietary license the user has to pay for.

Developing and publishing open source software can be used as a tool to gain competitive advantage, to promote the commoditisation of solutions, or to change a value chain. It has been used to create new ecosystems for platform-oriented businesses, to weaken the importance of suppliers, and to facilitate the introduction of new generations of technologies. The value created by open source may rely on the opening of new market opportunities, new partnerships, independence from third parties, or cost reductions. Web-scale players such as Google, Facebook or Netflix often open source software components of their services or infrastructure (TensorFlow, Android, Kubernetes, Hystrix, React…) for such strategic reasons.

[5] Riehle, D. (2007, April). The Economic Motivation of Open Source: Stakeholder Perspectives. *IEEE Computer* vol. 40, no. 4, pp. 25-32.

Relevance of open source

Due to digitalization, the importance of software has grown dramatically. Open source has revolutionized the software industry. According to Open Source Zone[6], a survey conducted by Black Duck Software and North Bridge revealed that a majority of businesses now rely on open source software for increased security and the lack of license fees. It is estimated that the use of open source saves businesses $60 billion annually.

Most of the biggest software companies in the world are radically changing their approach to open source. An example is Microsoft, whose change of strategy demonstrates that open source software is no longer the province of revolutionary hackers but an essential marketing element especially for large software companies. Another example is IBM's acquisition of Red Hat. More than 4550 Microsoft employees contributed to different open source projects on GitHub in 2017, Google to 2267 and Red Hat to 2057[7]. Open source is an essential element of any modern software production strategy.

Today's rise of AI is driven by open source and adoption has been accelerated. CB Insights research report on AI Trends 2019[8] notes that "The barrier to entry in AI is lower than ever before, thanks to open-source software" … and Yoshua Bengio commented (in a Mila announcement) that "The software ecosystem supporting deep learning research has been evolving quickly, and has now reached a healthy state: open-source software is the norm; a variety of frameworks are available, satisfying needs spanning from exploring novel ideas to deploying them into production; and strong industrial players are backing different software stacks in a stimulating competition."

Most modern proprietary software contains open source elements or is built on top or connected to larger open source parts, because a lot of innovation is in open source built by communities. Developments by the larger open source communities and new business models around open source have changed the software market and landscape. The Apache Foundation, Eclipse Foundation, Linux Foundation and other such consortia are the home of countless communities developing software in an open source model. Big data as we know it today and most commercial offerings for big data are based on open source projects from the Apache Foundation for distributed computation, distributed storage and other essential functionalities.

While software providers will have to decide which third party tools, frameworks, and software packages they might include into their software solutions, the users of software more and more rely on SaaS offerings or Apps in the mobile world. In these scenarios, they typically do not care whether the software is open source or not, but primarily consider the costs and benefits of the software usage.

The traditional consideration of total cost of ownership, which includes the price of the software, hardware investments to run the software, administration and maintenance efforts etc. is replaced by just recurring cost. In this case, the issue of open or closed source is typically not relevant. On the other hand, the usage of appliances (hardware with pre-packaged software) becomes more relevant. This holds for the internet of things, but also for AI (e.g. Tensorflow Processing Units).

[6] https://dzone.com/articles/benefits-of-open-source-vs-proprietary-software
[7] https://resources.whitesourcesoftware.com/blog-whitesource/git-much-the-top-10-companies-contributing-to-open-source
[8] https://www.cbinsights.com/research/report/ai-trends-2019/

Open Source considerations in software production

Reproducibility

The myth of the infinite reproducibility of digital goods, that could apply for example to multimedia contents, may not apply for some software which may be intrinsically complex, require a high degree of expertise and relatively long periods of apprentice for being used/reused. Nevertheless, most of the effort of the software industry, addressed at simplifying software production, is also simplifying the reuse/customization of open source packages. Recommenders are an example of this trend: they are based on artificial intelligence (AI) algorithms which suggest solutions to developers all along the software production process. It is easy to foresee that the more AI techniques/tooling will be adopted by the software industry, the easier it will be to select the best software solution given specific requirements, including total cost of ownership , to reuse software , and to adopt OSS packages as a whole. The same could apply the other way around: open source is also changing the way industries produce software. DevOps is an example: open source production processes are at the root of the last innovation wave in software production. Another good example is so-called 'containers': the most innovative solutions come from open source communities (e.g. Docker). Very heavyweight J2EE containers are going to be substituted by lightweight components based on standard interfaces and OSS containers. Another example is IT integrators, who used to prefer integration of proprietary and closed software packages to provide solutions to customers. The major cost in these projects was the purchase of licenses, and integrators were quite limited in the adaptation they could make to the original software to customize it to customer needs. Nowadays, integrators are keener to reuse existing open source software, which is adaptable and usually free; they have to spend more in adaptation, but the solutions are more flexible and reusable, revenue and margin are improved, and customers like the openness of the solutions and freedom of choice of providers.

The cost of open source

OSS is not free (as in "free beer", different from free as in "freedom"). It has some costs. Since there are usually no license fees and maintenance fees to pay, open source is at first glance cheaper than proprietary software. It should thus lower the barriers for companies - especially for SMEs - to invest in digital technologies. However, what should be considered is the Total Cost of Ownership. When software is used, indirect costs are incurred in addition to direct costs on licenses and maintenance. There is for example a cost associated to learn about the software you want to reuse. Users of OSS either rely on the community to evolve the software, or invest the money for supporting and maintaining software themselves, or pay a company for service and support. There are costs to enhance OSS if the functionality is insufficient. It is essential to check the licensing conditions when creating a derived work; e.g. some licenses mandate providing source code on redistribution. Another possible cost is ensuring compatibility when installing a new version of some OSS (although major open source projects use state of the art continuous integration with intensive regression tests). On the other hand, open source users are not forced to migrate to new versions as is sometimes the case for proprietary software. Finally, there is the cost to integrate OSS into an overall solution or product, though this arises when integrating any software from an external provider.

User engagement

Many companies are trying to engage not only developers but also other citizens in the development of software solutions. A social engagement corresponds to the need to identify software requirements and address correctly real user needs. While in many domains it is difficult to involve end-users, this is not true for middleware, where the target audience is the developer community itself. In this case, it is evident that direct involvement of many developer communities makes a difference. Many companies, including US giants, are engaging communities and creating awareness. Thus open source is a good marketing and product

development strategy, especially when the company culture is able to welcome, organize, lead, select and motivate contributions from the large audience of software developers. Open source communities use the same crowd sourcing mechanisms that are used in other communities like Wikipedia or OpenStreetMap although with more constraints related to the technical aspects of software. Participating in open source communities is also a great way to develop technical and relational skills, in multicultural environments. Moreover, it can contribute to the education of students in technical or sectorial curricula in universities and engineering schools, or of professionals for continuous skills development during their careers. For example, when hiring, some IT companies now ask for the open source portfolio of the candidates.

Open source communities

Open source only means the software is available under an open source compliant license. Therefore, there is a lot of open source software that is created in very different ways. Some software is created by one person or by one company and therefore it does not have a huge community around it. This kind of software does not necessarily enjoy all of the perceived benefits of open source software. Many software companies are mostly concentrating on open source software that has a community around it, which is more likely to get the benefits advertised. Android is probably a good example of open source that is controlled by a single entity without a real community - and even there, Google seems to have moved a lot of functionality e.g. to Play Store APIs to protect its own business. Creating a community for an old product or legacy technology might prove challenging. For accelerating new technology areas, it works much better. Communities are only successful when four characteristics are present: an active community; alignment with industry; a well-timed activity; and good governance of the project.

Security of open source

When source code is open to its users, bugs or vulnerabilities can be discovered and fixed quickly by the community. This ‘self-healing’ mechanism can relieve users of the fear about securing their sensitive data and thus may help considerably in the adaptation of digital solutions. But consider the Heartbleed Bug, when more than 2 years passed until the malicious code lines in the OpenSSL library were discovered, during which time passwords and data traffic could be accessed unnoticed by third parties. The self-healing mechanism can backfire: hackers can study the source code and analyse it for vulnerabilities. In the worst case, they could even infiltrate the community and implement backdoors into the source code. Open source of itself therefore does not offer a higher level of security than proprietary software: it basically depends on how well-managed the OSS project is and how vigilant the community is.

User-adaptable software

Since the source code can be modified by everyone for any purpose, open source can allow flexible, tailor-made solutions for the user. This possibility must be balanced against the effort that goes along with not only making modifications but understanding the code in the first place. In order to make the right changes in the right place, a deep understanding of the source code is required, along with the necessary programming skills. In addition, individual modifications may not flow back into the community code. This makes it more difficult to upgrade to later versions, since the individual modifications made in the old version have to be implemented in each new version all over again (‘backporting’). In continuous integration settings, updates and testing for new versions can be automated, facilitating the task of remaining up-to-date. On the other hand, a large part of developers’ time is spent on debugging the code. When building on proprietary code the developer has limited ways to study what happens in the functions behind the developer API and almost no way to deal with bugs in the proprietary code. In open source the developer can trace the software execution within the open source code, which helps a lot in fixing bugs.

User independence

The freedom to use and modify the code accorded by open source licenses prevents so-called vendor lock-in, i.e. dependence on a single vendor. However, this independence can only be achieved if the user has the competence to implement the open source in its IT landscape. In general, most users rely on external service providers to provide missing skills, additional competence, more resources, or cost-optimization. One common OSS business model is downstream value creation: the money is not earned from the software itself, but from software-related services. The user is recommended to assess the activity and vibrancy of the community to ensure that the OSS in question is being maintained, improved and optimized by third parties.

Open source, IPR and business development

Open source introduces alternative business models, and some open source communities are a paradigm for social innovation; recall that open source was originally conceived as a lifestyle, a way to democratize access to technology and avoid power concentration. It could be said that software has paved the way for social innovation initiatives all over the world. While volunteers have changed the way we 'crystalize' knowledge (Wikipedia), many productive communities have been inspired by open source success stories: the Peer-to-Peer economy (P2P) is based on the same principles - openness, reusability, and a completely different approach to IPR. The open hardware movement and the open innovation laboratories (e.g. fablabs) are other examples of the open source way of thinking. Open source is going a step further: there are examples of communities created around open source products, adopting completely new organizational models. The Enspiral community in New Zealand is an example of this trend: one of their sponsored products, Loomio, is a solution that helps communities to organize themselves and take decisions collectively, introducing new horizontal management strategies.

Due to the voluntary basis of many open source communities, there is the inherent risk of the withdrawal of a key community member, or abandoned software rapidly becoming obsolete. These risks should be evaluated alongside other risk factors by any would-be OSS user. Having a clear and public governance model and an active developers' forum might mitigate such risks. However, many major open source projects are backed by companies providing professional developers to support the community as these projects are part of their innovation and business strategy (for example cloud oriented middleware and tools).

The use of OSS in business applications requires a good knowledge of open source licensing, as not all licenses have same implications in terms of requirements and consequences. Normally, OSS licenses are among those recognized by the Open Source Initiative[9]. There are two main types of licenses: the copyleft and permissive. The spirit of the permissive licenses is to spread the software as much as possible, avoiding any control over its distribution and reuse, so few if any restrictions are applied to use, modification and commercialization. Permissive licenses are preferred in research and academic environments. Copyleft licenses are more oriented to protect the rights of the original owner and the community, with restrictions on licensing for redistribution and to prevent modifications being kept private.

A more detailed discussion about licensing is beyond the scope of this paper. However, it should be stressed that a thorough awareness of original code licensing and deep analysis of which open source license should be used is needed before including any such software in a solution or product. Tools (open source or proprietary) are available to assist in the assessment of open source components and to identify the licenses they need to comply to.

[9] https://opensource.org/licenses

Open source is ideally suited for the execution of exploratory pilots to test a technology, due to the low direct costs, the unbureaucratic usage and the possibility for individual modifications. At this early stage, it is important to test new applications, services or business models as simply and as fast as possible. Here open source can fully exploit its potential.

Open source cooperation and methodology is great when interests align around a project. Doing shared R&D together is a great way to drive efficiency and innovation in areas where parties align their interests. This is beneficial for everybody.

Relevance of open source for ICT research and innovation

Open source is an important component in speeding up ICT research and innovation. New code can be built on top of prior work and researchers can experiment with alternative algorithms and implementations easily. For instance, in modern machine learning research it is typical that researchers release their code, validate their results with publicly available test datasets, and publish their findings in open scientific forums, typically ArXiv. The end result of this is faster progress and better verifiability of new ideas, and in the longer run new novel business opportunities – whether they ultimately result in OSS or proprietary software.

Conclusions

Open source has lovers, haters and sceptics. The positive effects of open source in open innovation processes and research environments are undeniable. It is of especial interest for the piloting and experimental phases of innovation, accelerating the development phase and reducing the prototyping time. However, it is essential to have the technical skills for reusing, adapting and extending the source code, and knowledge about licensing. Although originally in many open source communities the work was done on voluntary basis, the reality is that behind many open source initiatives, such as MySQL and Android, there are business interests and companies driving the development strategy. Open source is not incompatible with proprietary software. Depending on the need, one or the other may be more convenient, and in some case, they co-exist for the same solution. In this paper we have discussed the different considerations which need to be addressed when open source is going to be used, and different benefits and risks that need to be known and assessed before use.

Recommendations

NESSI recommends that the European Commission:

- continues supporting the open source model as one way to facilitate easy access to fundamental technology under equal conditions to all organizations across Europe; and
- does not preclude other models of software licensing (e.g. SaaS, proprietary), especially if this helps exploitation.