

Effective Strategies for Using Open Source Software and Open Standards in Organizational Contexts

Experiences From the Primary and Secondary Software Sectors

Björn Lundell, Simon Butler, Thomas Fischer, and Jonas Gamalielsson, University of Skövde

Christoffer Brax, Combitech

Jonas Feist, RedBridge

Tomas Gustavsson, PrimeKey Solutions

Andrew Katz, University of Skövde and Moorcrofts

Bengt Kvarnström, Saab Aeronautics

Erik Lönroth, Scania CV

Anders Mattsson, Husqvarna

// Many companies seek to engage with open source software (OSS) projects. Based on insights and experience from practice, we present seven strategies for organizations to leverage long-term involvement with OSS projects. //

THROUGH THE YEARS, individuals and organizations have contributed to and witnessed more widespread development, procurement, use, and deployment of complex software systems that involve data processing and the maintenance of associated digital assets in a range of domains. This has caused a number of challenges for private and public sector organizations in different usage contexts and domains.¹⁻⁹

Companies in the primary and secondary software sectors need to deal with an increasing amount of software provided through industrial products and innovative applications and services. Several primary software sector⁶ companies have extensive experience and business offerings as service providers related to

the development and deployment of solutions, whereas many companies in the secondary sector¹⁰ create software as part of their product and service offerings in different domains, such as automotive, avionics, and outdoor power products. Many organizations have long recognized that “only a small part (5 to 10%) of the software is differentiating” and that involvement with nondifferentiating software projects in open collaboration has many potential benefits.¹¹ For example, when developers paid by a company contribute to internal software projects that constitute a commodity (i.e., nondifferentiating software projects) with respect to a company’s business, there is a significant risk of reinventing the wheel and wasting valuable resources. This, in turn, may inhibit innovation and have a demoralizing effect on the company’s own developers.

Open source software (OSS) is released under a license that complies with the open source definition (www.opensource.org/osd). Widely used OSS has been provided by projects under a small set of software licenses approved by the Open Source Initiative (www.opensource.org). A standard that complies with the definition presented by European Interoperability Framework version 1.0 constitutes an open standard.⁶ Since such standards permit implementation under different licenses and thereby enable competition, the Swedish National Procurement Services stipulates that when organizations use their framework agreements, they are allowed only to reference open standards when expressing a mandatory requirement which refer to a standard in public procurement projects.^{6,12}

There are complex relationships between the development of standards and OSS projects, something

that has received attention among policy makers and organizations involved with standards development.¹³ Formal standards may impose legal and technical challenges for OSS projects,⁵ and widely used software applications that deviate from a technical specification when implementing a standard may also impose interoperability challenges.¹⁴ Many organizations seek to utilize OSS and open standards to address challenges related to lock-in, interoperability, and long-term maintenance to process and sustain associated digital assets through compliant file formats and protocols. Such standards can be implemented through projects that provide and deploy software under different conditions, including various closed source software licenses and all OSS licenses.⁵

Several OSS projects have gained significant commercial interest and attracted contributions from individuals employed by different companies, and it is clear that many successful OSS initiatives are supported by some form of nonprofit organization.¹² For example, among globally recognized foundations that support OSS projects, there are the Linux Foundation (governing the Linux kernel and many other projects), the Eclipse Foundation (providing governance for many projects including development tools), and the Apache Foundation (providing governance for the Apache HTTP Server Project and hundreds of others). These and other foundations have played important roles in the governance, nursing, and promotion of a large set of well-known OSS projects in conjunction with associated communities.

In addition, there are numerous other foundations and charities that support and govern a few (or even

a single) OSS projects that provide widely used products. For example, the MariaDB foundation supports MariaDB Server, the Legion of the Bouncy Castle supports Bouncy Castle, and the Document Foundation supports LibreOffice (and some other OSS projects, such as Document Liberation). It should be noted that there are independently governed OSS projects (e.g., Curl) that are (and have been) used by many companies for decades.¹²

Based on findings from a four-year research project,⁶ this article presents seven practically grounded strategies that have evolved from collaborative research studies involving researchers and practitioners representing large and small companies in the primary and secondary software sectors. Specifically, the overarching goal is to understand how companies seeking to engage with OSS projects can establish strategies for the development, procurement, and deployment of software systems that can achieve long-term strategic benefits through open standards and their implementations.

The Strategic Use of OSS Projects

Today, OSS is widely used in the private and public sectors. An increasing number of individuals and organizations engage with OSS projects for a variety of reasons. Several countries, governments, companies, and public sector organizations have undertaken strategic initiatives and established policies that detail recommendations for how to engage with OSS.¹² For example, in 2019, the Swedish Agency for Digital Government (DIGG) introduced a strategy for the use and development of software, detailing recommendations for different OSS licenses. On 20 May 2020, DIGG presented an independent review (provided by the first

author of this article) of its recommendations,¹² one of which, applying to all private and public sector organizations seeking to establish OSS projects, is to use copyleft licenses from the General Public License (GPL) family [e.g., the Lesser GPL (LGPL)]. In particular, research shows that when a new OSS project needs to implement standards that are important for promoting interoperability and avoiding lock-in, it is often very sensible to use a license from the GPL family.⁵

Depending on the business models they use, some companies may be uneasy with establishing a project that will provide OSS under a license with a strong copyleft effect, such as GPL 3.0 and the Affero GPL (AGPL) 3.0, whereas licenses that have a weak copyleft effect (e.g., LGPL 2.1 and LGPL 3.0) would be fine for almost all relevant scenarios. We find that using a license with a copyleft effect protects continued OSS project openness, something some companies consider a prerequisite for contributing to an existing project.¹⁰ Further, in a scenario where a company considers a project to be nondifferentiating, a strong copyleft license (e.g., GPL 2.0, GPL 3.0, and AGPL 3.0) may be preferable in the event that the organization that controls the source code relicenses the project and instead provide future releases of the software under a closed source software

licence. One very successful example of a project that provides OSS under the GPL 2.0 license is the Linux kernel, which has established a vibrant community and attracted contributions from individuals representing many different companies.

Business models in assorted companies, domains, and scenarios may result in varying preferences and attitudes toward open collaboration. It is clear that many professionals representing numerous companies have contributed to OSS projects through the years.² For a variety of reasons, individuals may have strong preferences for different governance models and wish to engage only with OSS initiatives that fulfill their aims. OSS licenses are often categorized along a copyleft dimension (i.e., licenses that have a copyleft effect and those that lack such an effect, often referred to as *permissive licenses*). Further, OSS licenses can be categorized into a dimension related to patent clauses (i.e., licenses that have an explicit patent clause and those that lack such a clause). See Figure 1 for a presentation of commonly used licenses in each category that are also recommended in a review of a Swedish policy for OSS.¹²

Studies show that many professionals have contributed to and are engaged with projects that provide OSS under different versions of GPL licenses.^{1,2,7,10,11} Among projects that

provide OSS under GPL version 2, we find well-known products, such as VLC Media Player (LGPL 2.1) and the Linux kernel (GPL 2.0). Further, projects that provide OSS under GPL version 3 include PeaZip (LGPL 3.0), a file and archiving application; the GNU Compiler Collection (GPL 3.0); and Nextcloud (AGPL 3.0) for file sharing and collaboration. Projects that provide widely deployed OSS under permissive licenses that lack explicit patent clauses include the Nginx (the BSD-2-Clause license) web server project, the Contiki-NG (the BSD-3-Clause license) operating system project for the Internet of Things, the Bouncy Castle (the MIT license) cryptographic library project, and the X-Road (MIT License) project for interorganizational interoperability. Further, among projects that provide widely deployed OSS under a permissive license that contains explicit license clauses, we find Apache CloudStack (Apache License 2.0) and OpenStack (Apache License 2.0), both for cloud computing.

Based on interviews with 22 experts that have more than 10 years of experience in leading roles in widely deployed OSS projects, we identified a number of key factors that contribute to successful OSS project governance. Among the respondents, we find stark support for transparency and clear rules concerning what to expect when engaging with an OSS project. For example, as stated by one respondent, “One key factor is that the contributors understand what they’re getting into.” Several respondents emphasized that most successful governance “seems to be controlled by the inbound and outbound licenses.” We note that “inbound licenses” refer to those granted by project contributors, whereas “outbound licenses” are granted by a project (e.g., rights for

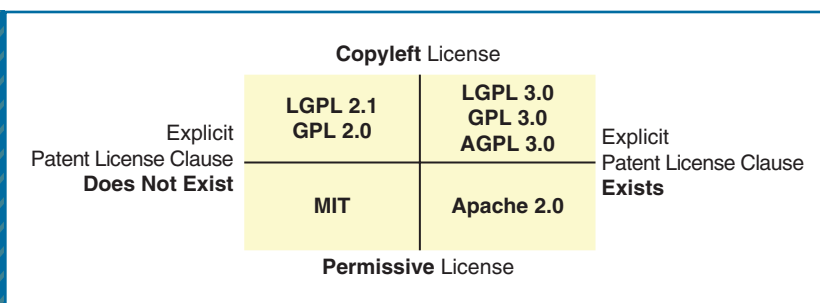


FIGURE 1. Different types of OSS licenses (see <https://opensource.org/licenses>).

code users and other recipients, including contributors who make use of a project as a whole). Under the concept of “inbound equals outbound,” there is a legal assumption concerning licenses that are “fully symmetric,” something used by many OSS projects that have a copyleft license. Further, we note that some experts express “great qualms about contributing to a project with an asymmetric agreement.”

These findings confirm previous research results from the embedded systems area, where consultants have expressed clear preferences for OSS provided under the GPL copyleft license,¹⁰ which is commonly used in projects that promote “fully symmetric” rights among contributors (the most prominent of which being the Linux kernel). On the other hand, if a company contributes to OSS that includes asymmetric licenses and rights grants, it follows that it (or any other contributor) will convey rights to the project (for example, under the Apache Contributor License Agreement) that are broader and less restrictive than the license it may receive from the project. This means that a contributor who assumes that a project respects software freedom (because, for example, it has an outbound GPL license) could find that the software is later made available under a non-free proprietary license (in the sense of respecting the four freedoms of software; see www.fsf.org) if all contributions are made under a permissive agreement. This is not to say that there is anything wrong with permissive contribution agreements per se, but it may be perceived as unfair if a project owner sets expectations that one form of outbound licensing model will be followed and then changes to another model without involving contributors.

We find that recognizing individual contributors is more important than identifying organizations. One expert stressed that successful governance “happens when you concentrate on the technical reasons [and] the technicalities of the contribution and maintainability.” As stated by another respondent, “The Golden Rule is the key: don’t do to others what you wouldn’t have done to yourself.” As the most important aspect of successful OSS project governance and management, one expert emphasized “human factors” and basic human kindness. Similarly, another stressed that leadership with “a little too much rock star” can do a lot of damage.

Strategies for Company Engagement With OSS Projects

Bridging the gap between industrial and OSS development practices has been an issue for many companies for more than a decade,¹¹ and it imposes challenges for enterprises in the primary and secondary software sectors. Individuals, companies, and other types of organizations may benefit from and engage with OSS projects in a variety of ways. Figure 2 presents a conceptual model for how companies (blue) can benefit from and contribute to OSS projects (yellow) through five principle strategies.⁶

Adopting OSS project work practices within closed contexts (strategy 1), sometimes referred to as *immersource development*,¹¹ is seen as beneficial by many companies for improving procedures in intra- and intercompany development scenarios. Further, using OSS products and OSS development tools (strategy 2) is common in numerous organizations. Incorporating OSS components from external projects (strategy 3) provides opportunities to reuse and integrate

valuable modules. In particular, companies may benefit from engaging with developments in relevant external OSS projects. They may gain from integrating OSS in collaboration with partner enterprises that have established credibility, saving time and leveraging expertise. Externally developed OSS may need adaptation and technical and legal reviews before it can be integrated with code being written internally.

For companies that have incorporated OSS from external initiatives, it is advantageous, for a number of reasons, to establish a presence in associated project communities and to provide contributions in return. Contributions can take the form of bug fixes, code, and participation in discussions about improvements [strategy 4(a)]. Companies may also provide direct (or indirect) financial contributions, something which may be seen as a form of outsourcing the long-term maintenance of strategically important OSS components.¹¹ If, on the other hand, enterprises do not contribute bug fixes, they will face additional costs to modify the internal code base for each new project release, which may be very expensive.^{2,6} Further, in some scenarios, it may be advantageous to open internally developed (closed source) applications and release them as OSS on collaborative platforms to establish a community [strategy 4(b)]. If an application has become a commodity, it may be advantageous to share development and maintenance costs while seeking business opportunities (e.g., support and related services) related to the software.¹⁵

For companies that seek to benefit from leveraging nondifferentiating external OSS, it is essential to engage with and build credibility in selected projects and their associated communities (strategy 5). To be successful,

this requires long-term engagement and strategic considerations.² Further, we note that it is far from uncommon that competing firms engage in non-differentiating software development activities through open collaboration via OSS projects.^{10,11} Figure 2 illuminates complex (many-to-many) relationships between internal software (in blue) and OSS projects (in yellow).

For example, an initiative may provide OSS that is used internally by hundreds of companies (strategy 3). Similarly, a company may, through its paid developers, provide code [strategy 4(a)] to hundreds of OSS projects.

Based on findings from collaborative research and practical experience, we have evolved a set of tips for companies to leverage opportunities through engagement with OSS projects and participation in associated communities. The seven tips specifically relate to three of the five principle strategies (strategies 3–5) in Figure 2. In relation, we elaborate on experiences in primary and secondary software sector contexts. Companies’ ability to innovate and address future challenges can be promoted through initiatives for ensuring long-term competence and skills development among its developers. They should consider investing time for employees to participate in OSS projects that will have long-term importance and thereby build a presence in initiatives and communities. Based on this, we recommend that organizations consider practical tip 1:

- *Practical tip 1:* An enterprise’s interests may be protected by giving developers time to contribute their expertise to OSS standards-based

implementations on which the business depends. Contributions may take the form of participating in discussions about the direction of development and adherence to a standard as well as donating code to improve a product.

When developers participate in OSS projects, there are opportunities for companies to influence future decisions. Strategic involvement promotes insights into the future direction of an important project. This, in turn, may significantly improve opportunities for congruence with a company’s internal goals. Further, contributions to external OSS projects may reduce an organization’s maintenance burden. Participation should be seen as a long-term investment (see practical tip 2), and various

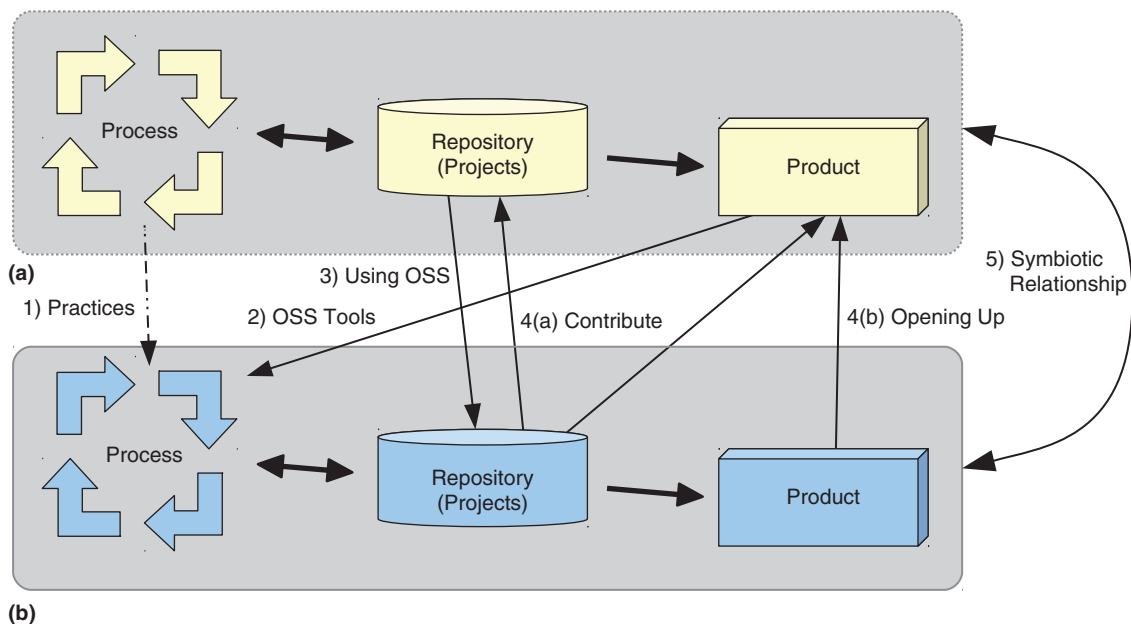


FIGURE 2. Leveraging opportunities with OSS projects.⁶ (a) OSS projects undertaken in open collaborative contexts. (b) Software projects executed in closed company contexts.

activities need to be considered across a long horizon to better understand potentially different goals and priorities (see practical tip 3). Hence, companies must be selective and carefully consider their involvement with OSS projects. Based on this, we recommend that they consider practical tips 2 and 3:

- *Practical tip 2:* OSS project structures can provide opportunities for organizations to participate in governance processes to support their business aims. Organizations should ensure that their involvement is seen as a cost or investment for which there is a return.
- *Practical tip 3:* OSS projects are not obliged to deliver new functionality and bug fixes to match businesses' time requirements, and they will not always share enterprises' priorities. When resolving OSS bugs or adding functionality, consider the time frame within which a matter must be resolved. If a solution is required more quickly than a project can deliver, you should resolve the problem yourself and try to minimize the long-term consequences by reporting the problem and solution to the OSS project.

For companies and their representatives to fully appreciate the evolution of OSS projects, it is essential to participate in associated communities to gain a deeper understanding of the code being developed. Governance and work practices differ among OSS initiatives, and there is a need to engage with members of communities, as important information may be discussed at various meetings (see practical tip 4). Company representatives need to build trust among other

community members to facilitate effective interaction and gain a deeper knowledge of the code (see practical tip 5). Based on this, we recommend practical tips 4 and 5:

- *Practical tip 4:* Although much development planning takes place openly in OSS projects, some plans may not be clearly documented. To avoid unnecessary work when considering feature requests, search project documentation, issue trackers, and mailing lists for similar proposals. If necessary, reevaluate and revise suggestions accordingly. Ask core developers whether a proposed feature is already being developed or would be accepted, and learn their preferences for how a feature should be delivered.
- *Practical tip 5:* Core OSS developers often request specific forms of evidence so that they are able to investigate reported behaviors. Learning to create this results in a deeper understanding of the software and leads to more efficient problem resolution.

Companies that seek to promote interoperability and the long-term maintenance of digital assets while avoiding lock-in must engage with OSS projects that implement open information and communications technology (ICT) standards (see practical tip 6). However, using closed file formats and data is often unavoidable. Just because a file format adheres to a particular standard does not mean the standard is freely available to use or will remain free. The specifications of some standards are provided under conditions that, for legal and technical reasons,

may inhibit implementation in OSS projects. Before companies can include data and documents in closed-format files, they must procure all necessary rights (including all patent licenses) to enable implementation in sustainable OSS projects (see practical tip 7). Based on this, we recommend that companies consider practical tips 6 and 7:

- *Practical tip 6:* ICT standards and their implementation in OSS are of strategic importance to any organization wishing to address challenges related to lock-in, interoperability, and long-term maintenance. Organizations developing and providing standards-based technologies with an aim to implement an OSS strategy need to engage with OSS projects that implement standards.
- *Practical tip 7:* To manage data and documents in closed formats, acquire before procurement all necessary rights (including all patent licenses) so files can be implemented in software that can be used and distributed under different licenses (including all licenses for OSS).

It is widely recognized that open standards, especially when implemented in OSS, contribute to interoperability by ensuring that data and systems can be interpreted independently of the tools that generated them. Such standards are also important for avoiding problematic lock-in effects in many scenarios for private and public sector organizations. The guidelines (in the form of seven practical tips) presented in this article provide a potentially valuable resource for any organization that needs to develop, use, and deploy software in different



BJÖRN LUNDELL is a professor at the University of Skövde, 541 28, Skövde, Sweden, where he leads the Software Systems Research Group. His research interests include fundamental sociotechnical challenges concerning software systems, focusing on different aspects of lock-in, interoperability, and longevity of systems. Lundell received his Ph.D. from the University of Exeter. He is a Member of IEEE and the Association for Computing Machinery. Contact him at bjorn.lundell@his.se.



CHRISTOFFER BRAX is a consultant at Combitech, Linköping, 583 30, Linköping, Sweden, working in systems engineering, requirements management, systems design and architecture, and IT security. Brax received his Ph.D. from Örebro University. Contact him at christoffer.brax@combitech.com.



SIMON BUTLER is a researcher in the Software Systems Research Group, University of Skövde, 541 28, Skövde, Sweden. His research interests include software engineering, open source software, and program comprehension. Butler received his Ph.D. in computing from the Open University. He is a Member of IEEE, the Association for Computing Machinery, and the British Computer Society. Contact him at simon.butler@his.se.



JONAS FEIST is a cofounder of RedBridge, 111 20, Stockholm, Sweden. His research interests include business models related to OSS, and in cloud computing, particularly the development and application of container technologies. Feist received his M.Sc. in computer science from the Institute of Technology, Linköping University. Contact him at jonas.feist@redbridge.se.



THOMAS FISCHER is a senior lecturer at the University of Skövde, 541 28, Skövde, Sweden, where he is a member of the Software Systems Research Group. His research interests include open source software and open standards, in particular, file formats, lock-in, and interoperability. Fischer received his Ph.D. from the Technical University of Kaiserslautern. He is a member of a number of open source and open data communities. Contact him at thomas.fischer@his.se.



TOMAS GUSTAVSSON is a cofounder and the chief technology officer of PrimeKey Solutions, 171 73, Solna, Sweden. Gustavsson received his M.Sc. in electrical and computer engineering from KTH Royal Institute of Technology. The founder of the open source enterprise public key infrastructure project EJBKA, he contributes to numerous open source projects and is an Open Source Sweden board member. Contact him at <https://www.linkedin.com/in/tgustavsson/> or tomas.gustavsson@primekey.com.



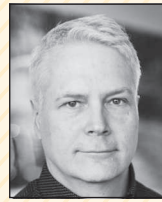
JONAS GAMALIELSSON is a senior lecturer at the University of Skövde, 541 28, Skövde, Sweden, where he is a member of the Software Systems Research Group. His research interests include free and open source software. Gamalielsson received his Ph.D. from Heriot Watt University. Contact him at jonas.gamalielsson@his.se.



ANDREW KATZ is a visiting researcher in the Software Systems Research Group, University of Skövde, 541 28, Skövde, Sweden, and a partner at the law firm Moorcrofts, Marlow SL7 1PB, U.K. His research focuses on technology law with a particular interest in open design (including hardware), development, and licensing. Katz received his M.A. in law from Cambridge University, U.K. and is qualified as a barrister at the Inns of Court School of Law in London, U.K. (nonpracticing). Contact him at <https://moorcrofts.com/team/andrew-katz/> or andrew.katz@moorcrofts.com.




BENGT KVARNSTRÖM is a senior systems engineer at Saab Aeronautics, 581 88, Linköping, Sweden, where he leads the group responsible for software development processes, methodology, and tools. His research interests include systems and software development, with a specific focus on strategies for adoption and use of third party components provided under different conditions. Kvarnström received his M.Sc. in applied physics and electrical engineering from the Institute of Technology, Linköping University. Contact him at bengt.kvarnstrom@saabgroup.com.



ANDERS MATTSSON is the lead architect for Internet of Things systems at Husqvarna, 561 82, Huskvarna, Sweden. His research interests include strengthening software engineering practices in organizations, software architecture, and model-driven development of embedded real-time systems. Mattsson received his Ph.D. in software engineering from the University of Limerick. He is a Member of IEEE. Contact him at <https://www.linkedin.com/in/andersmattsson/or/anders.mattsson@husqvarnagroup.com>.



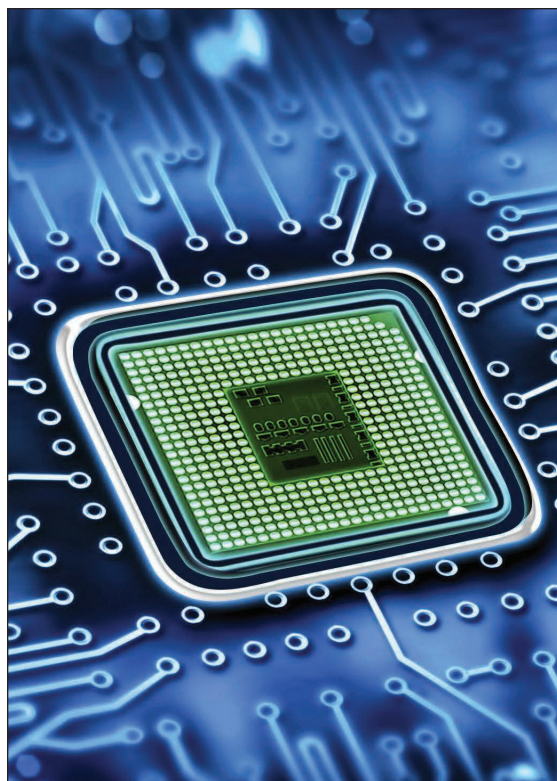
ERIK LÖNROTH leads supercomputing technical initiatives at Scania CV, 151 32, Södertälje, Sweden. His research interests include the development of supercomputer environments for industry, open source software governance, and high-performance computing. Lönroth received his M.Sc. in computer science from Linköping University. Contact him at erik.lonroth@scania.com.

contexts. This article proposed and elaborated on effective organizational strategies and guidelines for using OSS and open standards, based on experience from individuals and organizations in the primary and secondary software sectors. The strategies and guidelines highlight how enterprises can leverage opportunities through engagement with OSS projects and associated communities. 

References

1. B. M. Brosgoi, “How to succeed in the software business while giving away the source code: The AdaCore experience,” *IEEE Softw.*, vol. 36, no. 6, pp. 17–22, 2019. doi: 10.1109/MS.2019.2934044.
2. S. Butler et al., “On company contributions to community open source software projects,” *IEEE Trans. Softw. Eng.*, 2019. doi: 10.1109/TSE.2019.2919305.
3. J. Gamalielsson and B. Lundell, “On influences between ICT standards and their implementation in open source software projects: The case of H.264,” in *The Past, 20/20 and FUTURE of ICT Standardisation, Booklet of Papers of the 11th International Conference on Standardisation and Innovation in Information Technology (SIIT 2020)*, EURAS Contributions to Standardisation Research, K. Jakobs, Ed. Aachen: Mainz Publishers, 2020, vol. 16, pp. 1–12.
4. J. Linåker and B. Regnell, “What to share, when, and where: Balancing the objectives and complexities of open source software contributions,” *Empirical Softw. Eng.*, vol. 25, no. 5, pp. 3799–3840, 2020. doi: 10.1007/s10664-020-09855-2.
5. B. Lundell, J. Gamalielsson, and A. Katz, “Implementing IT standards in software: Challenges and recommendations for organisations planning software development covering IT standards,” *European J. Law Technol.*, vol. 10, no. 2, 2019. [Online]. Available: <https://ejlt.org/index.php/ejlt/article/view/709/>
6. B. Lundell et al., “Addressing lock-in, interoperability, and long-term maintenance challenges through Open Source: How can companies strategically use Open Source?” in *Proc. 13th Int. Conf. Open Source Syst. (OSS 2017)*, IFIP AICT 496, 2017, pp. 80–88.

7. H. Mäenpää, S. Mäkinen, T. Kilamo, T. Mikkonen, T. Männistö, and P. Ritala, "Organizing for openness: Six models for developer involvement in hybrid OSS projects," *J. Internet Services Appl.*, vol. 9, no. 1, 2018, Art. no. 17. doi: 10.1186/s13174-018-0088-1.
8. D. Mueller and D. Izquierdo-Cortazar, "From art to science: The evolution of community development," *IEEE Softw.*, vol. 36, no. 6, pp. 23–28, 2019. doi: 10.1109/MS.2019.2936177.
9. Y. Zhang, M. Zhou, A. Mockus, and Z. Jin, "Companies' Participation in OSS development—An empirical study of OpenStack," *IEEE Trans. Softw. Eng.*, 2019. doi: 10.1109/TSE.2019.2946156.
10. B. Lundell, B. Lings, and A. Syberfeldt, "The practitioner perceptions of open source software in the embedded systems area," *J. Syst. Softw.*, vol. 84, no. 9, pp. 1540–1549, 2011. doi: 10.1016/j.jss.2011.03.020.
11. F. van der Linden, B. Lundell, and P. Marttiin, "Commodification of industrial software: A case for open source," *IEEE Softw.*, vol. 26, no. 4, pp. 77–83, 2009. doi: 10.1109/MS.2009.88.
12. B. Lundell, "Analys av DIGG:s policy för utveckling av programvara, version 1.0, 20 May," Skövde University Studies in Informatics 2020:1, Univ. of Skövde, Skövde, Sweden, 2020. [Online]. Available: <http://urn.kb.se/resolve?urn=nbn:se:his:diva-18895>
13. K. Blind and M. Böhm, "The relationship between open source software and standard setting," in *EUR 29867 EN, JRC (Joint Research Centre) Science for Policy Report*, N. Thumm, Ed. Luxembourg: Publications Office of the European Union, 2019. doi: 10.2760/163594.
14. S. Butler et al., "Maintaining interoperability in open source software: A case study of the Apache PDFBox project," *J. Syst. Softw.*, vol. 159, p. 1,1045, Jan. 2020. doi: 10.1016/j.jss.2019.110452.
15. P. S. Kochhar, E. Kalliamvakou, N. Nagappan, T. Zimmermann, and C. Bird, "Moving from closed to open source: Observations from six transitioned projects to GitHub," *IEEE Trans. Softw. Eng.*, 2019. doi: 10.1109/TSE.2019.2937025.



IEEE TRANSACTIONS ON

COMPUTERS

Call for Papers: *IEEE Transactions on Computers*

Publish your work in the IEEE Computer Society's flagship journal, *IEEE Transactions on Computers*. The journal seeks papers on everything from computer architecture and software systems to machine learning and quantum computing.

Learn about calls for papers
and submission details at
www.computer.org/tc.



Digital Object Identifier 10.1109/MS.2021.3130452