The Journal of Systems & Software 186 (2022) 111152

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Considerations and challenges for the adoption of open source components in software-intensive businesses☆

Simon Butler [a,*], Jonas Gamalielsson [a,*], Björn Lundell [a,*], Christoffer Brax [b], Anders Mattsson [c], Tomas Gustavsson [d], Jonas Feist [e], Bengt Kvarnström [f], Erik Lönroth [g]

[a] University of Skövde, Skövde, Sweden
[b] Combitech AB, Linköping, Sweden
[c] Husqvarna AB, Huskvarna, Sweden
[d] PrimeKey Solutions AB, Stockholm, Sweden
[e] RedBridge AB, Stockholm, Sweden
[f] Saab AB, Linköping, Sweden
[g] Scania CV AB, Södertälje, Sweden

ARTICLE INFO

Article history:
Received 17 September 2020
Received in revised form 30 September 2021
Accepted 19 November 2021
Available online 24 December 2021

Keywords:
Component-based software development
Software adoption
Open source software

ABSTRACT

Component-Based Software Development is a conventional way of working for software-intensive businesses and Open Source Software (OSS) components are frequently considered by businesses for adoption and inclusion in software products. Previous research has found a variety of practices used to support the adoption of OSS components, including formally specified processes and less formal, developer-led approaches, and that the practices used continue to develop. Evolutionary pressures identified include the proliferation of available OSS components and increases in the pace of software development as businesses move towards continuous integration and delivery. We investigate work practices used in six software-intensive businesses in the primary and secondary software sectors to understand current approaches to OSS component adoption and the challenges businesses face establishing effective work practices to evaluate OSS components. We find businesses have established processes for evaluating OSS components and communities that support more complex and nuanced considerations of the cost and risks of component adoption alongside matters such as licence compliance and functional requirements. We also found that the increasing pace and volume of software development within some businesses provides pressure to continue to evolve software evaluation processes.

1. Introduction

> "If most of the code comprising your product or service isn't open source software, it's highly likely that you're wasting effort and cash reinventing the wheel." Spinellis (2019)

Component-Based Software Development (CBSD) is a dominant paradigm in software development with more than 90% of business software projects incorporating Open Source Software (OSS) components (Synopsys, 2020; Tidelift, 2019; Szulik, 2018). Managers and practitioners in software-intensive businesses using CBSD frequently make decisions concerning the adoption of OSS components to be integrated into their tools and products (Franch et al., 2015; López L. Costal et al., 2016; Hauge et al., 2009, 2010; Stol and Ali Babar, 2010a; Ayala et al., 2011a; Lundell et al., 2017; Lenarduzzi et al., 2019; Spinellis, 2019; Kazimierczak et al., 2020). A key economic motivation for many businesses is the use of OSS as a means of short-cutting resource intensive software development processes (Spinellis, 2019; European Commission, 2017; Kazimierczak et al., 2020; Petersen et al., 2018; Badampudi et al., 2016).

During the first ten years of the twentieth century, a number of formalised processes were proposed for the evaluation of OSS components (Stol and Ali Babar, 2010b). In general, however, practitioners and companies developed their own approaches to support the evaluation and adoption of OSS components (Ayala et al., 2011a; Hauge et al., 2010). During the second decade, the

☆ Editor: Neil Ernst.

* Corresponding authors.
E-mail addresses: simon.butler@his.se (S. Butler), jonas.gamalielsson@his.se (J. Gamalielsson), bjorn.lundell@his.se (B. Lundell), christoffer.brax@combitech.com (C. Brax), anders.mattsson@husqvarnagroup.com (A. Mattsson), tomas.gustavsson@primekey.com (T. Gustavsson), jonas.feist@redbridge.se (J. Feist), bengt.kvarnstrom@saabgroup.com (B. Kvarnström), erik.lonroth@scania.com (E. Lönroth).

https://doi.org/10.1016/j.jss.2021.111152

motivations of software developers to adopt OSS components have evolved (Lenarduzzi et al., 2019). Developers moved from seeing OSS components as a cost free alternative, to something that requires investment of effort for businesses to adopt and use. Accordingly they prioritised the ease of software customisation and the availability of support from either community or commercial sources as key factors when evaluating software (Lenarduzzi et al., 2019).

The challenge of OSS component adoption is more complex and nuanced than simply identifying functionally suitable software (Spinellis, 2019). Businesses need to consider multiple additional factors, for example, the software licence of the component in the context of their own licensing policy (Stol and Ali Babar, 2010a; van der Burg et al., 2014; Spinellis, 2019; Petersen et al., 2018), and also the viability and stability of the OSS project community that develops the component (Stol and Ali Babar, 2010a; TODO Goup, 2018; Franch et al., 2015; López L. Costal et al., 2015, 2016).

Some factors for software adoption have been found to be tractable to software repository mining methods and the application of metrics. For example, the RISCOSS project developed tools to assess activity in GitHub projects, which can then contribute to a business's evaluation of the software (Franch et al., 2015). Ongoing licence compliance checks for the software bill of materials (SBoM) are being automated through the use of SPDX (SPDX. Workgroup, 2020); examples include systems developed by Siemens AG (Fendt and Jaeger, 2019), the Open Source Tooling Group (OSTG) (Geyer-Blaumeiser, 2019), and the Linux Foundation (ACT, 2020).

Much recent research has focused on aspects of the process of OSS component adoption. Meanwhile the extent to which companies adopt and use OSS components has increased (Synopsys, 2020; Tidelift, 2019) with only limited accounts of the practices used within companies to support component adoption. More formalised schemes for evaluating OSS software described 10 years ago by Stol and Ali Babar (2010b), among others, no longer feature in the academic and practitioner literature and more recent research has shown that attitudes towards OSS components within businesses are changing (Lenarduzzi et al., 2019). In the absence of formalised methods, and given changes in the extent and scale of OSS component adoption, what work practices are businesses using to evaluate OSS components for adoption? To provide an answer to this question we use two research objectives.

O1 To identify and analyse the work practices used in software-intensive businesses to support the evaluation and possible adoption of OSS components in CBSE.

O2 To understand challenges faced by software-intensive businesses when evaluating OSS components for adoption, and how those challenges contribute to the work practices used.

To meet these objectives we undertake a multi-case study of software-intensive businesses in Sweden. The businesses range in size and operate in the primary and secondary software sectors in a variety of domains including IoT, security, engineering, and the provision of cloud infrastructure.

The following section gives an account of the background to the problem of selecting and evaluating OSS components for adoption in CBSD and reviews the relevant academic research. In Section 3 we give an account of the methodology used for the case study. Section 4 reports findings which are then discussed Section 5. We draw our conclusions and summarise the contribution made by the article in Section 6.

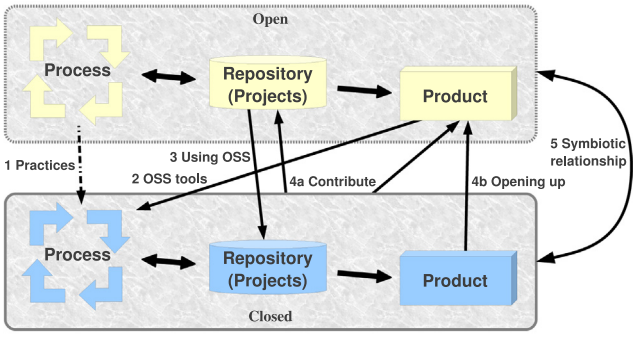

Fig. 1. Strategies for open source software adoption and project engagement. *Source:* Reproduced from Lundell et al. (2017).

2. Background & related work

CBSD has become a conventional way of working for many software-intensive businesses, and OSS is one source of components for integration into tools and products (See path 3 in Fig. 1). Businesses evaluate software components that are candidates for adoption and for OSS components the criteria considered include licensing (Cohn and Spiegel, 2011; Copenhaver, 2010) and the nature of the OSS project (Hauge et al., 2010; Franch et al., 2015), as well as functional requirements (Spinellis, 2019). Business considerations are also a factor when choosing between OSS components, internal development and proprietary software components, because the adoption of OSS may offer cost advantages in the short term, but can require additional resources in the longer term (Petersen et al., 2018) While the selection and adoption of a component is initially a linear process, there is also a need for an iterative, ongoing evaluation of adopted OSS components because of functional changes to the software as it evolves, as well as changes within the supporting OSS community, and, sometimes, changes to the licence used for the software (Copenhaver, 2010; TODO Goup, 2018).

A variety of formal and semi-formal schemes to evaluate OSS components have been proposed. Stol and Ali Babar (2010b) identified twenty evaluation methods and frameworks developed between 2003 and 2009, but their adoption has been limited. Research has instead shown that practice is often less formalised (Hauge et al., 2010; Ayala et al., 2011b; Lenarduzzi et al., 2019; Franch et al., 2015; López L. Costal et al., 2016). A systematic literature review by Hauge et al. (2010) found six distinct approaches recorded by researchers for evaluating OSS software, including components to be used in software products. A key challenge identified was that of estimating the cost of component integration (Hauge et al., 2010). Ayala et al.'s (2011b) extensive study of OSS adoption, including the integration of OSS components, found companies used a range of processes to evaluation components. Some businesses used no formal process, others performed no evaluation, while some used third parties to undertake or lead the evaluation process, and some had established their own process and methods (Ayala et al., 2011b).

A recent study by Lenarduzzi et al. (2019), replicating two earlier surveys (del Bianco et al., 2011; Taibi, 2015), found changes in developers' motivations to adopt OSS components between 2010 and 2016. A key shift identified was from considering OSS components as free to understanding that the adoption of OSS is a cost to the adopting company (Lenarduzzi et al., 2019). Developers prioritised the ease of software customisation and the availability of support from either community or commercial sources as key factors, while their managers prioritised the availability of commercial support. Factors considered almost as important

were quality, flexibility, maturity and reliability (Lenarduzzi et al., 2019). Research has also shown the complexity of the decision-making within businesses when selecting the source of software components for integration (Petersen et al., 2018). There are trade-offs for a business when choosing between OSS components, internal development and proprietary software, each of which requires careful consideration (Petersen et al., 2018). A systematic literature review by Badampudi et al. (2016) identifies a largely common process for evaluating proprietary software and OSS components for adoption in CBSD. The main considerations when selecting external components were found to be purchase cost, cost of maintenance, integration effort and quality, with criteria used differing for commercial-off-the-shelf components and OSS (Badampudi et al., 2016).

Franch et al. (2015) and others have mined OSS project repositories and mailing lists to extract metrics that can be used to evaluate OSS projects in terms of activity levels and the number of individuals involved, amongst other factors. One simple metric that has been explored is the Truck Factor, the number of individuals critical to the functioning and survival of a project (Rigby et al., 2016; Cosentino et al., 2015; Ferreira et al., 2017). Others have looked in more detail at assessing communication within a project community as an indicator of project vitality and risk, considering both the volume of messaging and the number of individuals involved in activities such as bug fixing. The metrics give a wider picture of the health of a project (Yahav et al., 2014). Further work in the RISCOSS project (Franch et al., 2013, 2015) developed a configurable tool for assessing GitHub communities to support risk assessment during the software adoption process.

In addition to risks arising from within project communities, there are also threats arising from decisions made by companies supporting software development in OSS projects. Zhou et al. (2016) studied the impact on developer participation of commercial backing for three Java projects, including Apache Geronimo. The study found that company backing has both positive and negative effects on the OSS projects and can lead to an abrupt cessation of development work when company support is withdrawn (Zhou et al., 2016).

Research related to RISCOSS has developed models of the processes of OSS software evaluation and OSS project adoption, including the adoption of OSS components for inclusion in products (López L. Costal et al., 2015, 2016).

Other researchers have considered the challenges of licence compliance, particularly in large software products containing a mixture of proprietary and OSS components (Harutyunyan et al., 2019; Riehle and Harutyunyan, 2019; Fendt and Jaeger, 2019). Interactions between OSS licences are complex (German and Hassan, 2009; van der Burg et al., 2014), and can be a challenge that is not always resolved through manual inspection (van der Burg et al., 2014). The Linux Foundation's OpenChain project (The Linux Foundation, 2019) has created standards such as SPDX (SPDX. Workgroup, 2020), for example, that can be used to support automated compliance checking. Fendt and Jaeger (2019) and Harutyunyan et al. (2019) examine the challenge of managing OSS licensed components in large software projects. Fendt and Jaeger (2019) describes work at Siemens AG to implement tool chains for licence compliance checking in CI/CD. A key concern is that the process of *licence clearance* or establishing the licensing of source code as opposed to accepting what the packager declares the licence to be, is an expensive task, and one that in a complex SBoM should only be completed once for each package (Fendt and Jaeger, 2019). Riehle and Harutyunyan (2019) summarise the problem, some potential solutions, and identify some unanswered research questions. Automation can be used, but there are limitations to current solutions that require additional tooling to be developed (Harutyunyan et al., 2019).

In summary, OSS component evaluation practices are evolving in response to changes in software development methods (Fendt and Jaeger, 2019; Harutyunyan et al., 2019; Riehle and Harutyunyan, 2019), and changes have also been observed in software developers' motivation to adopt OSS (Lenarduzzi et al., 2019). However, the focus of much recent research is on solving particular challenges in OSS component adoption rather than considering the wider process. In this study we seek to understand the work practices businesses use to manage entire process of OSS component adoption.

3. Research approach

In this work we undertake a descriptive case study (Gerring, 2017; Yin, 2018) of the challenges faced and work practices used for OSS component adoption in purposefully sampled (Patton, 2015) software-intensive businesses in Sweden.

3.1. Case selection and setting

The subjects for the case study (see Table 1) were selected to be part of the LIM-IT project[1] (Lundell et al., 2017) to reflect a variety of companies of different sizes working in multiple domains in the primary and secondary software sectors.

Combitech AB is computer consultancy with around 30 offices in Sweden that works in a variety of domains in both the private and public sectors. Husqvarna AB develops and manufactures forestry and gardening tools for professional and domestic use. PrimeKey Solutions AB specialise in public key cryptography applications and manages OSS projects including EJBCA and SignServer. RedBridge AB is a computer consultancy specialising in the delivery of cloud solutions in Sweden. Saab AB is a large defence, aeronautics, and security company. And Scania CV AB develops and manufactures commercial vehicles.

3.2. Data collection and analysis

Software developers and managers in each of the companies represented by the coauthors were informed about the study through emails sent internally by the respective coauthors and invited to participate in the research. Each prospective participant received an explanation of the purpose of the research and the method, written by the first author, which also emphasised the voluntary nature of participation, and that participants were able to withdraw from the research prior to the anonymisation of responses.

Written questions were provided to respondents in both Swedish and English. Questions were drafted in English by the first author, a native English speaker, and translated into Swedish by the second author, a native Swedish speaker. The translation process was iterated where necessary to ensure the questions were asked as consistently as possible in both languages. Both English and Swedish were used because a small proportion of technical staff working in Sweden speak English as a first or second language and have limited knowledge of Swedish. Respondents were able to give answers in their preferred language.

Each respondent was asked to answer general questions to provide background information about their company role, the role that they play in the software adoption process, and to give an overview of the adoption process used. Respondents were also asked to give an account of the process of selecting and evaluating between one and four OSS components considered for adoption and use in products. The questions about the software evaluation

[1] https://www.his.se/en/research/informatics/software-systems-research-group/lim-it/.

Table 1
The businesses participating in the study.

Business	Main areas of activity
Combitech AB	Computer consultancy
Husqvarna AB	Engineering, IoT
PrimeKey Solutions AB	Security applications
RedBridge AB	Cloud services
Saab AB	Engineering, Safety-critical Systems
Scania CV AB	Engineering, Safety-critical Systems

and selection process were designed to allow respondents to answer expansively. Respondents were also asked, where possible, to include an account for at least one OSS component that had been rejected with the intention of obtaining richer insights. (The questions used are given in Appendix.)

The distribution of questions and collection of responses within each business was coordinated by the author working in that company. For each company, responses were reviewed to ensure that sensitive company information was not disclosed, and that the responses were anonymised, before being returned to the first author.

We received a total of 13 responses from companies represented by the authors (See Table 2). In addition, the first three authors also had limited access to documents supporting the software component evaluation processes used by Saab and Scania. Access to documentation was constrained for reasons of business confidentiality and security.

The responses were analysed by the first author using semantic thematic analysis (Braun and Clarke, 2006; Braun et al., 2018). Responses in Swedish were translated to English by the first author and the translation reviewed, revised if necessary, and approved by the second author. The English responses and translations were the primary input for analysis, and the original Swedish responses were referred to as required to confirm the analysis.

The academic literature — particularly Ayala et al. (2011a,b), Hauge et al. (2010) and Stol and Ali Babar (2010a) — was used to inform the thematic analysis of the responses. Summaries of the results and analysis were discussed iteratively by the authors over a two month period. The discussions served two purposes. The first was to refine the categorisation developed during the semantic analysis (McDonald et al., 2019). The second was to identify and analyse points made by respondents that could then be drawn together into topics for the discussion. Discussions took place by phone, video conferencing, and email.

4. Findings

In this section we first report findings related to the work practices used for software component adoption, namely who is doing the work and how the evaluation and adoption process is organised. Subsequently, we report findings related to the challenges of OSS adoption and how they are addressed.

4.1. O1: Work practices used for OSS component evaluation

Table 2 summarises the work roles for each respondent and shows that those evaluating software components all have some level of responsibility within the business. Table 2 also indicates whether the respondent works in a group when evaluating software components, and the type of role that they have in the evaluation process. Not all respondents gave sufficient information for them to be assigned to a category. Most respondents undertake evaluation of software components in groups and only two work alone. The supervisory roles reported by respondents fell into two categories. One was the perspective of an experienced practitioner providing oversight to ensure that processes had been followed. The other was to provide oversight from a different role, in the case of R_4 a more business oriented perspective that further informs the evaluation and adoption process. In two cases, respondents also stated that they had designed evaluation processes.

Two respondents were explicit that a formalised process of evaluation is being used within the respective businesses. In both cases the business has a set of requirements for OSS components that support the software evaluation process, and in one case (R_{11}) have led to the design of a process that is applied to the evaluation and adoption of both OSS and proprietary components. A respondent in a third business (R_5) drew attention to a decision-making process where the size and relative importance of the software being evaluated allowed for a choice of evaluation process. R_5 identified a relatively light-touch developer-led process for smaller libraries, particularly for prototyping, and a more formal process at the team and departmental level for larger components.

Only two respondents reported working alone when evaluating software components. Both had supervisory or oversight roles where they reviewed evaluations made by others which accounted for much of their lone working.

Processes for continuing evaluation and compliance checking were also considered by respondents. Activities included monitoring activity in the OSS project, and checking licences remain suitable; specific details are reported in the following subsection. One respondent advocated change to the business's current way of working: R_{13} expressed the need to automate routine compliance checks, such as monitoring source code for licence changes. R_{13} manages a number of products with the business and the development teams under their supervision work at a scale and pace that makes using the business's formalised evaluation process extremely difficult and time-consuming.

As well as the responses from individuals, we had some constrained access to documentation related to the evaluation processes used within Saab and Scania, the two largest companies involved in the study. In both cases, practitioners and managers are provided with a framework within which to evaluate OSS components. While both facilitate a diligent evaluation of software components, there are differences in scope and features. For example, Saab apply a single framework to evaluate both proprietary and OSS software. Scania provide a list of commonly used OSS licences with "traffic light" colours so that practitioners may easily recognise which licences are safe to use (green), those that require deeper consideration (yellow), and some that are not approved for use (red).

4.2. O2: Challenges of OSS component evaluation

The evaluation of software components for potential adoption is multi-dimensional. As R_4 summarises the process:

> "Evaluation of software in general includes stating requirements from different perspectives and then narrowing the selection of available options down."

For reporting, the challenges of evaluating OSS components are divided into four categories. Three categories – technical, compliance and OSS project attributes – are largely separate, and a fourth category, risk, is in part a cross-cutting concern for the first three categories as well as identifying some additional factors considered by respondents for component evaluation. The aspects of OSS components considered by the respondents in each category are summarised in Tables 3, 4, 5 and 6.

Table 2
Respondents' work roles and responsibilities for OSS component evaluation.

	Respondent												
	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}
Work Role													
Manager						✓		✓		✓	✓		✓
Architect	✓		✓	✓	✓				✓			✓	
Development team manager		✓											✓
Development team leader	✓						✓						
Developer	✓				✓								
Evaluates													
Alone	✓									✓			
Group	✓	✓			✓		✓	✓	✓			✓	✓
Additional Responsibility													
Designer						✓					✓		
Reviewer		✓				✓							
Supervisor				✓							✓		✓

Table 3
Technical aspects of OSS components considered by respondents during evaluation.

Category	Description
Feasibility	Is it feasible to use the component being considered, i.e. can it be integrated?
Good-Fit	A potential component must be a good fit with existing components, i.e. limited reworking of existing software, or the component, would be required to integrate the component.
Limited Alternatives	That an OSS component is one of a few (if not the only) components that meet the functional requirements.
Long-Term Maintenance	A need for, or expectation of, or assessment of long-term maintenance of a component. Minimum threshold considered by some respondents is 10 years or so.
Maintenance	That the software component is well-maintained.
Maintenance In-House	The action of taking over a proportion of the maintenance of an OSS dependency. This might, for example, be the creation of a fork that is then maintained internally, or it can be that a business takes over a leading role in the OSS project.
Non-Trivial	The requirement that an OSS component should solve a non-trivial problem.
Requirements	That the OSS components meets functional requirements.
Safety-Critical	That the software meets safety-critical requirements.
Security	That adopting the software creates no security issues, and that the software is secure in use.
Software Adaptation	The overhead of adapting the software for integration or for the purpose it is being used for.
Standards	That the software component implements a standard.
Time-Saver	Adoption of an OSS component must save in-house development time.

4.2.1. Technical factors

The majority of components evaluated by respondents were software libraries. The GNU compiler collection (GCC) and Java were also discussed, as were middleware libraries and other components used in distributed systems and product delivery systems such as containers. GCC, for example, needs to be considered as a component because it requires runtime libraries to be distributed with compiled binaries. Consequently, any business using the GCC compiler suite has to consider the compiler runtime libraries to be a component of the product. Similarly, Java requires a virtual machine to be installed to support the product at runtime. While the licensing of runtime components permits their use, suppliers do sometimes change licences and there are, for example, multiple Java platforms available (e.g. openJDK, Amazon Corretto, Oracle) with some differences in licence model.

One of the first questions in an evaluation process reported by respondents focused on the need for the component and an early assessment of its value to the business. R_1 summarised this part of the process in the requirement, "A component must solve a non-trivial problem or save a lot of time". R_7 also identified time-saving as a factor saying that using the Apache Commons libraries reduced development time by eliminating a need for routine coding. (The technical aspects of an OSS component considered by respondents are summarised in Table 3.)

Components need to meet, or get close to meeting, the functional requirements of the business. R_{12} expressed this requirement as, "Does the component solve my problem?" and R_8 poses the question, "… does it have functionality we are looking for[?]", and R_9 also specifies that a component "should, so far as possible, follow standards". R_5 also said that smaller components were "Selected only if a given library solves the functional requirements".

Non-functional requirements were also considered during evaluation. R_6 and R_{11} both evaluate the security of a component. R_{11} expressed concerns about whether a given component was secure in operation, and R_6 considered security risks more generally. R_9 also considered performance:

> "The library or product should, either through our own or another's evaluation, show that it is stable and has good performance for the use case it is intended to be used for."

That R_6 values the evaluations of others is related to the wider use of the OSS component, which is considered further in Section 4.2.3.

Table 4
OSS licence considerations during component evaluation.

Category	Description
Compatibility	That the licence of the software being evaluated is compatible with the licence used in the product/tool it will be incorporated in. Includes understanding of interaction of the licence with that of non-OSS components, and whether the OSS licence permits commercial use.
Revision	That a company might ask that an OSS project revise the licence to make it compatible with that required by a business.
Obligations	Whether the licence used include obligations for the company, the product (e.g. copyleft licences), or obligations to the upstream project.

Table 5
OSS project attributes considered during evaluation.

Category	Description
Active Development	That there is software development and maintenance activity in the OSS project. Also in the case of an adopted component, it may be that a decrease in development is an indicator that it is time to move away from the project
Active Support	That OSS community mailing lists and other support forums are active.
Community Activity	The there is activity within the OSS project community, e.g. that there are multiple, regular participants in a range of software development and community activities, that there is discussion of development priorities.
Documentation	That the software is well documented.
Project Structure	The ownership, governance, community and so on.
Support Options	A consideration of whether there is the opportunity to pay for support.
Supported Project	That the OSS project is organised or managed by a (larger) company or organisation such as a foundation.
Wide Use	That the software is widely used.

Table 6
Risks assessed by respondents during OSS component evaluation.

Category	Description
Commercial Terms	That software may have some licensing terms that allows commercial use, but there is a risk that the licensor may withdraw those terms, and change the licence for commercial use.
Dynamic	That OSS projects and components are dynamic, as is software development within the business and thus risk is also changing.
Financial	That there are financial costs associated with adopting a component. Usually expressed in terms of labour required to adapt software prior to integration, or costs of paid support.
Lock-In	The risk of a lock-in effect.
Maintenance	A concern that the software may or will cease to be maintained, or the realisation of that risk.
Obsolescence	The risk that a software component will become obsolete through external changes or through the developing organisation (company or OSS project) stopping to support the software
Project Supported	That the organisation behind an OSS component considered for adoption may itself be considered to be a risk.
Reputation	A risk related to the business's reputation by association with the component, which includes potential issues related to marketing.
Security	Security risks as a consequence of using the component
Skills	The concern that the company has and retains staff with the necessary competence to use the software, and perhaps contribute upstream.

While a software component may meet functional and non-functional requirements, additional work can be required to integrate it with an existing product, and indeed integration with a company's software may be impossible. Integrating a component into a company's software product can be an expensive and time-consuming task, and the extent of any required integration work needs to be understood as part of the component evaluation. In some cases, such as middleware which implements a standard API, R_9 observed that the overhead of integrating the new or a replacement component is minimal. Other components require modifications either to the component or to the company's software. Reasons include incomplete functionality or architectural incompatibilities. R_2 highlighted this factor in the evaluation of one component where:

> "… we rejected several candidates … because we assessed that it would take too much time (if it was even possible) to integrate them with our system."

R_1 emphasised the need for good documentation, saying "A component must be well-documented and well maintained". R_9 also saw documentation as a necessity:

> "… in the case of complex components where the functionality cannot easily be read from the code there is also an emphasis on documentation being available."

Documentation supports component integration, future maintenance and possible extension of the component to meet the

business's needs. A lack of documentation is potentially an additional cost for a business and R_{12} reported having rejected a component because it was "sporadically" documented.

A further challenge for software development teams is the long-term maintenance of components. Having made an investment within the business to incorporate a component, the overhead of switching to a different component is a cost to be avoided. A second factor is the long-term maintenance of the product itself. R_{12} identified about 10–15 years of maintenance being required for the product, for example. Accordingly, the evaluation process must consider the prospects for the long-term maintenance of any component on a comparable timescale to the anticipated lifespan of the product. R_1 also implied the importance of stability of supply for maintenance saying:

> "A component must come from a larger, well-known organisation if the component is to form part of the solution …".

R_{12} highlighted the value of paid support for OSS projects to supply long-term maintenance of components. For some OSS projects the option of paid support is not available, and development within the project can stop, regardless of whether a business or large organisation supports project development. The challenge of such events for businesses adopting software components is considered further in Section 4.2.4.

4.2.2. Compliance

Licensing of software, and OSS licence and legal compliance are important aspects of the evaluation and adoption process considered by all respondents. Respondents referred mostly to licence compatibility between the licence of the software being evaluated and the licence used in the product or tool it will be incorporated in (See Table 4). R_5 highlighted a key condition for licences for components to be used in a business's products:

> "Important that the licence agreement permits inclusion in a product where not all components are open source …"

Some respondents also reviewed licences from the perspective of obligations. R_2 said that they consider:

> "The licence model and the requirements it places on us as an organisation and for our products."

While R_4 used more pragmatic terms to explain the purpose of reviewing the licence:

> "… to secure that we are allowed to do what we want and need to …"

R_5 was similarly pragmatic, saying:

> "… and that the licensing model does not impose any restrictions on how the library can be used."

Typically the evaluations reported included concerns about the implications of copyleft licences on the businesses' products, and also whether the licence might include obligations to the OSS project. Similarly, part of the oversight of component evaluation described by R_{10} includes a long-term perspective of the licence:

> "… to understand what implications the licence can have for the project [now] and in the future …"

While R_{10} was not explicit, it is clear that should the business' needs or use of the component change, then the licence needs to be appropriate in those cases.

Licenses were not always seen as inflexible. R_7 described an occasion where the business asked an OSS project to revise the licence used in a library.

> "We found an open source project (development library) that had the functionality we required. However, the licence was not compatible with the licence that we use. We asked the developers of the library about it, and they changed the licence to one that is compatible with our licence. We chose to use the library based on the functionality, limited complexity of the library, and active developers."

The requested licence revision – in this case from GPL v3 to LGPL v3 – retains the copyleft aspect of the licence for modifications to the library, but allows the library to be incorporated into products without a copyleft obligation applying to the product itself.

4.2.3. OSS project attributes

Attributes of the OSS project developing the component were also identified as important inputs to the evaluation process (See Table 5). Most respondents stated that the software should be actively developed and that the project community should be active. R_2 and R_5 also specified that the mailing lists and support forums should also be active. R_5 expressed this aspect of evaluation as follows, "Activity in the project, most recently updated, activity in user forum etc" and R_{10} characterised key attributes in the questions, "Is the community active, are there commercial actors that support the development?" R_{12} described a component evaluation process where an OSS library was available, but it was found that the software lacked maturity and had a relatively small community developing and supporting the software; the component was rejected.

As well as active support in the OSS project community, respondents also emphasised that good documentation of the software was a factor for evaluation, as noted above. R_6, R_9 and R_{10} further considered opportunities for paid support as part of their evaluation processes. Learning to use software libraries is a cost in the adoption of a component. Support in the form of good documentation reduces cost for the business, and paid support may further reduce costs in the longer term.

Wide use of a component was a consideration for some respondents as an indicator of expectation of continuing development and availability of support in the future. R_{12} characterised this consideration as the question, 'Is the component "popular"?' and R_{10} asks the question, "How common is the product?" R_8 selected one component, in part, on the grounds that "An active community and widespread use gave us confidence that it was a long-term solution." However, respondents also report that while popularity is a desirable factor, it was not always decisive, and this topic is considered further below.

4.2.4. Assessing risks

Another key challenge for businesses is assessing the risk involved in the adoption of a specific OSS component. Indeed, respondents, as noted above, often acknowledge change, and consequently risk, as an inherent aspect of component evaluation. Some of the risk areas identified by respondents are related to the initial adoption of the software component, while others are longer-term, dynamic risks that require vigilance and reassessment during the period the adopted component is used for. Assessment of the longer-term risks will also inform the initial assessment, in part at least. Some risks are characterisations of factors already identified in terms of risk. Others, however, were only described by respondents as risks. The areas of risk considered by respondents are summarised in Table 6.

R_6 described the evaluation process in terms of risk:

> "When requirements arise and a suitable alternative is identified from a technical perspective, a risk analysis is undertaken that focuses on the licence type, the structure of [project]

ownership and support opportunities, [software] lifecycle (activity level and longevity) and other risks (commercial, such as marketing related and possible cost).''

Other respondents' description of risk also fall into similar technical, project, and compliance areas described by R_6, as well as threats or risks to the business itself.

While software licences are not changed often, a change may have consequences for some users. For a business it may mean that it becomes impossible to use particular versions of software, and that a replacement component may need to be evaluated and adopted. R_{13} reported the business ceasing to use MongoDB when the software licence was changed to the MongoDB Server Side Public License in 2018. R_8 also discussed undertaking risk analyses where components are licenced with additional terms that permit commercial use. Where such a clause exists the licensor may choose to withdraw the permission for commercial use at some point, with consequences for the business. Similarly, R_{10}, as noted previously, identified future implications of licences as something to consider early in the evaluation process.

Activity within the OSS project developing the software can be seen in terms of risk, because, for example, levels of activity might be open to interpretation. R_2 outlines the need for judgement when assessing project community activity.

> ''How active the community around the component is and how mature the solution seems to be. The combination of these is of interest, depending on what the component is. A mature solution with few updates can indicate stability, but also that it is abandoned. Depending on what the component is and how good a fit it is currently you must consider the risks in each case.''

A similar idea was expressed by R_8 as a series of questions to assess risk, ''[is there] visible and expected maintenance, will the project continue, or will we need to take care of maintenance ourselves?''

The concerns expressed by R_2 and R_8 are, in part, evaluations of the risk of the business facing additional costs by adopting the component concerned that may arise from change in the OSS project. The initial assessment is one of the likelihood of additional costs and avoiding them. However, to paraphrase Heraclitus, change is the only constant and there is a need to protect the business from the consequences of external change, where possible, such as respondents' remarks above on ensuring paid support is available for the duration of the use of the component. Technologies nonetheless continue to develop, and markets and OSS communities change, consequently the evaluation of OSS components is a continuous process.

A technical risk identified by R_{11} is that of lock-in effects. Lock-in may arise for technical reasons alone, for example through adopting a specific standard or technology. Other respondents' descriptions of integration overhead recognise elements of lock-in effects that arise as a consequence of the cost to the business of integration. The concern that then arises, regardless of the source of the lock-in effect, is the cost of migrating to alternative component, should it become necessary.

The business risks identified by the respondents concern direct costs to the business, in terms of the expense of adopting a component. The overhead of integration has been mentioned previously, and there are additional sources of risk. For example, R_6 identifies potential reputational risks which may arise from association with a specific component, or a company that supports it.

Another risk is that the OSS project will stop developing the software; reasons might include a supporting company withdrawing funding, or the core developers changing their focus. Respondents discussed two specific instances. R_1 commented on Hystrix (Netflix, 2020), software that adds resilience to distributed systems, which is no longer actively developed. Hystrix was developed by Netflix, who have since changed their approach to network resilience, and, at the time of writing, Hystrix is maintained. R_1 said that the business has a period within which to select and adopt a replacement component. In a case reported by R_7, the company used a library that implemented an IETF specification. The OSS project that created the component stopped developing the software. The business decided to take maintenance of the library in-house, because the implementation was mature, and they had the necessary knowledge and skills to maintain it. However, it was not a long-term solution, and another OSS library, already used in R_7's product, later implemented the functionality.

A challenge faced within the business is the that of obtaining and retaining competence, both in the short and long terms. R_4 highlighted that the business needed to ensure:

> ''… we have people skilled enough to use and contribute to the development.''

How this can be achieved in the long term was not addressed in detail by respondents, but where a component requires integration there will always be a need to retain skills to manage the integration, just as there is a need for long-term maintenance in the upstream OSS project. R_8 commented on expertise within the business being a factor in the adoption of a component, and implied that the more than one individual had the requisite skills.

Returning to R_4's observation that the evaluation process considers a number of perspectives and narrows down the available options, some respondents observed that the selection of a specific library can involve making judgements about trade-offs. R_5 identified one such case, where there appear to be reasons not to use a particular library:

> ''Despite complex integration and the relatively low activity in the project, this component was selected because it closely matched the functional requirements and there was synergy with other projects''

R_2's comments above on assessing project activity reflect a similar judgement of risk and illustrates that while some factors considered during component evaluation and adoption are simple choices, others are considerably more nuanced, and related to the circumstances of businesses and trade-offs in the form the acceptance of some risks and costs.

5. Analysis

5.1. Discussion

The greater part of the process of evaluating an OSS component for adoption described by respondents might be viewed as one of risk assessment. There are clear criteria that can be seen as potential 'dealbreakers', as Spinellis (2019) describes them, such as an incompatible licence or absent functionality, that may make it easy to reject a particular component. Otherwise the process of component evaluation is less clear and was reported as more a process of appraisal and judgement than one of applying clear cut rules; although a business may have firm lines of the kinds of risks are acceptable. The process might be likened to that of purchasing a house, where a survey gives an understanding of the positive and negative physical attributes of the property, but is only one of a number of factors contributing to the decision to buy the property.

The challenges for businesses of OSS adoption fall largely into the categories identified by Stol and Ali Babar (2010a) and

reiterated more recently by Spinellis (2019), among others. What appears to have changed is the volume of OSS that is available and the extent to which it is used in products and to provide services, and that businesses and organisations have greater experience of OSS adoption for CBSD. The businesses whose staff participated in the study have developed processes that are appropriate for their structure and capabilities. Further there is an awareness that software development is evolving, both within and outside the business, and that the process for OSS adoption develops alongside technological change.

There is also a relationship between integration overhead and a form of lock-in implied in some responses. Given the overhead of integrating a given library, replacing it will also require some sort of overhead also. For drop-in replacements, such as the middleware examples, where there is a standardised API then integration overhead is zero, and, so long as there is a replacement, there is no lock-in. Where there is integration overhead, then there is a degree of commitment to the component that is costly to undo, and thus contributes to a lock-in to the component. Effectively then, the process of assessing the long-term maintenance of the software and viability of a project can be seen as mitigation of the consequences of this form of lock-in by trying to ensure that the software component will have a lifespan similar to the product in which it is incorporated.

A recurring theme amongst the responses is that of change. Change is inevitable and businesses when evaluating software components for use in tools products are trying to identify solutions that are less likely to change in ways that are unhelpful for the business. Or to have some form of mitigation in place, for example in the form of contracted long-term maintenance. Many respondents considered a large company or organisation backing an OSS project to be a desirable attribute in their component evaluation process. The implication in responses is that such OSS projects are more stable, and that the software will continue to be developed and maintained. However, as one respondent identified, it is not always a guarantee and a large supporting company may develop new technological needs and stop development, or change direction of development, in a way that is incompatible with the adopting business' needs.

One respondent also talked about a further dimension of change: the way in which the pace and scale at which they were working is making aspects of the company's software evaluation process difficult to follow. An aspect of continuing vigilance for adopted software components is that of compliance checks. There are methodologies for assessing compliance such as OpenChain and solutions for automated licence checking and component management are being developed (e.g. by Siemens AG (Fendt and Jaeger, 2019), the Linux Foundation (ACT, 2020), (HERE Europe B.V., 2020) and OSTG (Geyer-Blaumeiser, 2019)). Such automated approaches are not a replacement for the initial evaluation of licence compliance in a software component, but support the continuing compliance checks.

Questions for future work concern the type and extent of vigilance needed by a business to evaluate an adopted software component continuously. The introduction of automation has limitations in that it can currently detect changes made to licences, perhaps, and variations in activity in OSS communities. However, given the subtle and nuanced decisions made in the component adoption processes identified in this article, how vigilant does a company need to be in order to protect itself from potentially detrimental changes within the OSS project or software? And, what form should that vigilance take?

5.2. Threats to validity

As with any empirical study, there are threats to the validity of this work. We consider threats to construct validity and external validity. We do not consider threats to internal validity because no claims for causality are made, and statistical conclusion validity is not discussed because no statistical inference is used. There is a threat to *construct validity* from the semantic analysis of the interviews be being performed by a single author. The threat is mitigated in two ways. Firstly, the thematic analysis was informed by the academic literature and, secondly, iterative discussions between the authors of interview summaries and analysis were used to refine the thematic analysis.

Threats to *external validity* arise from the relatively small number of participants and that the authors and participants are based in a single European country. The threats are mitigated by the diversity of size of the software-intensive businesses represented by the authors, as well as the range of industries within which they operate, and the variety of domains in which the businesses develop software. In addition, while the companies are based in Sweden, they operate in an international marketplace with operational offices, collaborators, and partners in other countries and on other continents. Accordingly, given the contexts within which the respondents work, the research provides a rich account of practitioner and business approaches to software evaluation and adoption which may be generalised for businesses of similar sizes evaluating and adopting OSS components.

A further consideration is the possible impact of study design and subject recruitment on the responses received. The study design is focused on the practices used for OSS component adoption within companies and individual respondents were invited to participate through their employer. The intention of this approach was twofold: first to focus recruitment of potential respondents within each company on relevant individuals; and, second, to give each company the confidence to ensure sensitive information was not disclosed. However, individuals are potentially exposed to coercion to participate by their employer and may also bias their responses towards those that the respondent perceives to be "right" from the employer's perspective (Vinson and Singer, 2008). Furthermore, the privacy of the participation of individual respondents (Vinson and Singer, 2008) is compromised to an extent, and, accordingly, there may have been reluctance or unwillingness on the part of some individuals to participate on these grounds.

6. Conclusions

In this article we have reported and analysed results from a study of the experiences of managers and practitioners in six businesses in the primary and secondary software sectors in Sweden involved in the selection and evaluation of Open Source Software (OSS) components for use in products and internal tools.

The study focused on two research objectives:

- **O1** To identify and analyse the work practices used in software-intensive businesses to support the evaluation and possible adoption of OSS components in CBSE.
- **O2** To understand challenges faced by software-intensive businesses when evaluating OSS components for adoption, and how those challenges contribute to the work practices used.

We found that the businesses each adopt a pragmatic approach to the evaluation of OSS components. The processes used in larger companies are informed by guidelines or frameworks that practitioners can follow. In smaller companies, practitioners tend to take decisions about software adoption in groups, with managerial oversight.

Assessment and evaluation of individual components is driven by technical need in the first instance, and informed by judgements that consider multiple factors related to licence compliance and expected longevity of the OSS product and project. Further consideration is given to business factors, such as costs of integration and benefits gained from use, as well as the potential risks of adopting a specific OSS component. While the businesses are generally risk averse there are factors, such as the scarcity of alternatives and goodness of fit for individual components, that mean companies may trade-off increased risk for potential commercial advantages. Additionally, the evaluation processes used by businesses are under pressure to evolve because of changes in the pace and scale of software development, as well as changes in hardware and software technologies.

CRediT authorship contribution statement

Simon Butler: Conceptualisation, Methodology, Investigation, Writing – original draft, Writing – review & editing, Project administration. **Jonas Gamalielsson:** Conceptualisation, Methodology, Writing – review & editing. **Björn Lundell:** Conceptualisation, Methodology, Writing – review & editing, Supervision, Funding acquisition. **Christoffer Brax:** Investigation, Writing – review & editing. **Anders Mattsson:** Investigation, Writing – review & editing. **Tomas Gustavsson:** Investigation, Writing – review & editing. **Jonas Feist:** Investigation, Writing – review & editing. **Bengt Kvarnström:** Investigation, Writing – review & editing. **Erik Lönroth:** Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research has been financially supported by the Swedish Knowledge Foundation (KK-stiftelsen) and participating partner organisations in the LIM-IT project. The authors are grateful for the stimulating collaboration and support from colleagues and partner organisations. We are also grateful for the work of Tomas Bigun and Stefan Landemoo of Saab AB, and Jonas Öberg of Scania CV AB in support of this research.

Appendix. Questions sent to respondents

The English version of the questions used in the study is as follows:

1. Which of the following role descriptions applies to your job? (mark more than one if necessary)
    - manager
    - manager of multiple software development teams
    - software development team leader (multiple teams)
    - software development team leader (one team)
    - software developer
    - other - please add a brief description
2. Describe how are you involved in the process of OSS component adoption as part of your job? (e.g. do you select or evaluate software alone or in collaboration with others? Do you review or supervise how others select and evaluate software? Do you design processes for the selection and adoption of components?)
3. Give a brief description of the processes, and functional and non-functional criteria that you use to select and evaluate OSS components for possible adoption.
4. For each OSS component that you have evaluated or contributed to the evaluation of (up to a total of four) please give a brief account of how the component was selected for evaluation, and the functional and non-functional criteria used to evaluate it. If you are able to give two or more examples, please include one where the software was rejected with a clear statement of the reasons for doing so.

References

ACT, 2020. Automated compliance tooling. The linux foundation. URL https://automatecompliance.org/. (accessed: 30 September 2021).

Ayala, C., Cruzes, D.S., Franch, X., Conradi, R., 2011a. Towards improving OSS products selection – matching selectors and OSS communities perspectives. In: Open Source Systems: Grounding Research - Proceedings of the 7th IFIP WG 2.13 International Conference on Open Source Systems. OSS 2011, Springer, pp. 244–258. http://dx.doi.org/10.1007/978-3-642-24418-6_17.

Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J., 2011b. Selection of third party software in off-the-shelf-based software development — an interview study with industrial practitioners. J. Syst. Softw. 84, 620–637. http://dx.doi.org/10.1016/j.jss.2010.10.019.

Badampudi, D., Wohlin, C., Petersen, K., 2016. Software component decision-making: In-house, OSS, COTS or outsourcing – a systematic literature review. J. Syst. Softw. 121, 105–124.

del Bianco, V., Lavazza, L., Morasca, S., Taibi, D., 2011. A survey on open source software trustworthiness. IEEE Softw. 28, 67–75. http://dx.doi.org/10.1109/MS.2011.93.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3, 77–101. http://dx.doi.org/10.1191/1478088706qp063oa.

Braun, V., Clarke, V., Hayfield, N., Terry, G., 2018. Thematic Analysis. Springer Singapore, Singapore, pp. 1–18. http://dx.doi.org/10.1007/978-981-10-2779-6_103-1.

van der Burg, S., Dolstra, E., McIntosh, S., Davies, J., German, D.M., Hemel, A., 2014. Tracing software build processes to uncover license compliance inconsistencies. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering. Association for Computing Machinery, New York, NY, USA, pp. 731–742. http://dx.doi.org/10.1145/2642937.2643013.

Cohn, A.G., Spiegel, G., 2011. Effective open source development business practices. Comput. Internet Lawyer 28, 1–17.

Copenhaver, K., 2010. Open source policies and processes for in-bound software. Int. Free Open Sour. Softw. Law Rev. 1, 143–154. http://dx.doi.org/10.5033/ifosslr.v1i2.27.

Cosentino, V., Cánovas Izquierdo, J.L., Cabot, J., 2015. Assessing the bus factor of git repositories. In: 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering. SANER, IEEE, Piscataway, NJ, USA, pp. 499–503.

European Commission, 2017. The Economic and Social Impact of Software & Services on Competitiveness and Innovation. European Union, http://dx.doi.org/10.2759/949874.

Fendt, O., Jaeger, M.C., 2019. Open source for open source license compliance. In: Proceedings of the 15th IFIP WG 2.13 International Conference on Open Source Systems. OSS 2019, Springer International Publishing, Cham, pp. 133–138. http://dx.doi.org/10.1007/978-3-030-20883-7_12.

Ferreira, M., Valente, M.T., Ferreira, K., 2017. A comparison of three algorithms for computing truck factors. In: Proceedings of the 25th International Conference on Program Comprehension. IEEE Press, pp. 207–217. http://dx.doi.org/10.1109/ICPC.2017.35.

Franch, X., Kenett, R.S., Mancinelli, F., Susi, A., Ameller, D., Annosi, M.C., Ben-Jacob, Y., Franco, O.H., Gross, D., López, M., Oriol, M., Siena, A., 2015. The RISCOSS platform for risk management in open source software adoption. In: Open Source Systems: Adoption and Impact - Proceedings of the 11th IFIP WG 2.13 International Conference on Open Source Systems. OSS 2015, Springer, pp. 124–133. http://dx.doi.org/10.1007/978-3-319-17837-0_12.

Franch, X., Susi, A., Annosi, M.C., Ayala, C.P., Glott, R., Gross, D., Kenett, R.S., Mancinelli, F., Ramsamy, P., Thomas, C., Ameller, D., Bannier, S., Bergida, N., Blumenfeld, Y., Bouzereau, O., Costal, D., Dominguez, M., Haaland, K., López, M., Siena, A., 2013. Managing risk in open source software

adoption. In: Proceedings of the 8th International Joint Conference on Software Technologies. ICSOFT, pp. 258–264. http://dx.doi.org/10.5220/0004592802580264.

German, D.M., Hassan, A.E., 2009. License integration patterns: Addressing license mismatches in component-based development. In: 2009 IEEE 31st International Conference on Software Engineering. pp. 188–198. http://dx.doi.org/10.1109/ICSE.2009.5070520.

Gerring, J., 2017. Case Study Research: Principles and Practices, second ed. Cambrige University Press, Cambridge, UK.

Geyer-Blaumeiser, L., 2019. Ensuring Open Source Compliance using Eclipse Foundation Technology. Bosch Software Innovations Gmbh, URL https://github.com/Open-Source-Compliance/Sharing-creates-value/blob/master/Presentations/2019_10_22_EclipseConEurope_EnsuringOpenSourceCompliance.pdf. (accessed: 30 September 2021).

Harutyunyan, N., Bauer, A., Riehle, D., 2019. Industry requirements for FLOSS governance tools to facilitate the use of open source software in commercial products. J. Syst. Softw. 158, 110390. http://dx.doi.org/10.1016/j.jss.2019.08.001.

Hauge, Ø., Ayala, C., Conradi, R., 2010. Adoption of open source software in software-intensive organizations – a systematic literature review. Inf. Softw. Technol. 52, 1133–1154. http://dx.doi.org/10.1016/j.infsof.2010.05.008.

Hauge, Ø., Østerlie, T., Sørensen, C.F., Gerea, M., 2009. An empirical study on selection of open source software - preliminary results. In: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE Computer Society, USA, pp. 42–47. http://dx.doi.org/10.1109/FLOSS.2009.5071359.

HERE Europe B.V., 2020. OSS Review toolkit. HERE Europe B.V. URL https://github.com/oss-review-toolkit/ort. (accessed: 30 September 2021).

Kazimierczak, M., Breckwoldt Jurado, A., Wajsman, N., 2020. Open-Source Software in the European Union. European Union Intellectual Property Office, http://dx.doi.org/10.2814/866548.

Lenarduzzi, V., Tosi, D., Lavazza, L., Morasca, S., 2019. Why do developers adopt open source software? Past, present and future. In: Proceedings of the 15th IFIP WG 2.13 International Conference on Open Source Systems. OSS 2019, Springer International Publishing, Cham, pp. 104–115. http://dx.doi.org/10.1007/978-3-030-20883-7_10.

López L. Costal, D., Ayala, C.P., Franch, X., Annosi, M.C., Glott, R., Haaland, K., 2015. Adoption of OSS components: A goal-oriented approach. Data Knowl. Eng. 99, 17–38. http://dx.doi.org/10.1016/j.datak.2015.06.007, selected Papers from the 33rd International Conference on Conceptual Modeling (ER 2014).

López L. Costal, D., Ralyté, X., Méndez, M.C., 2016. OSSAP - A situational method for defining open source software adoption processes. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (Eds.), Proceedings of the 28th International Conference on Advanced Information Systems Engineering. CAiSE 2016, Springer, pp. 524–539. http://dx.doi.org/10.1007/978-3-319-39696-5_32.

Lundell, B., Gamalielsson, J., Tengblad, S., Yousefi, B.H., Fischer, T., Johansson, G., Rodung, B., Mattsson, A., Oppmark, J., Gustavsson, T., Feist, J., Landemoo, S., Lönroth, E., 2017. Addressing lock-in, interoperability, and long-term maintenance challenges through open source: How can companies strategically use open source? In: Open Source Systems: Towards Robust Practices - Proceedings of the 13th IFIP WG 2.13 International Conference on Open Source Systems. OSS 2017, Springer, pp. 80–88. http://dx.doi.org/10.1007/978-3-319-57735-7_9.

McDonald, N., Schoenebeck, S., Forte, A., 2019. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. In: Proceedings of the ACM on Human-Computer Interaction 3. http://dx.doi.org/10.1145/3359174.

Netflix, 2020. Hystrix: Latency and fault tolerance for distributed systems. Netflix. URL https://github.com/Netflix/Hystrix. (accessed: 12 September 2020).

Patton, M.Q., 2015. Qualitative Research and Evaluation Methods, fourth ed. Sage Publications Inc., Thousand Oaks, California, USA.

Petersen, K., Badampudi, D., Shah, S.M.A., Wnuk, K., Gorschek, T., Papatheocharous, E., Axelsson, S., Crnković, I., Cicchetti, A., 2018. Choosing component origins for software intensive systems: In-house, COTS, OSS or outsourcing? — a case survey. IEEE Trans. Softw. Eng. 44, 237–261. http://dx.doi.org/10.1109/TSE.2017.2677909.

Riehle, D., Harutyunyan, N., 2019. Open-source license compliance in software supply chains. In: Fitzgerald, B., Mockus, A., Zhou, M. (Eds.), Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability: Communications of NII Shonan Meetings. Springer Singapore, Singapore, pp. 83–95. http://dx.doi.org/10.1007/978-981-13-7099-1_5, Communications of NII Shonan Meetings (chapter 5).

Rigby, P.C., Zhu, Y.C., Donadelli, S.M., Mockus, A., 2016. Quantifying and mitigating turnover-induced knowledge loss: Case studies of chrome and a project at avaya. In: Proceedings of the 38th International Conference on Software Engineering. ICSE 2016, ACM, New York, NY, USA, pp. 1006–1016. http://dx.doi.org/10.1145/2884781.2884851.

SPDX. Workgroup, 2020. Software Package Data Exchange. The Linux Foundation, URL https://spdx.org/. (30 September accessed).

Spinellis, D., 2019. How to select open source components. IEEE Comput. 52, 103–106. http://dx.doi.org/10.1109/MC.2019.2940809.

Stol, K.J., Ali Babar, M., 2010a. Challenges in using open source software in product development: A review of the literature. In: Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. Association for Computing Machinery, New York, NY, USA, pp. 17–22. http://dx.doi.org/10.1145/1833272.1833276.

Stol, K.J., Ali Babar, M., 2010b. A comparison framework for open source software evaluation methods. In: Open Source Software: New Horizons - Proceedings of the 6th International IFIP WG 2.13 Conference on Open Source Systems. OSS 2010, Springer, pp. 389–394. http://dx.doi.org/10.1007/978-3-642-13244-5_36.

Synopsys, 2020. Open source security and risk ananlysis report. Synopsys. URL https://www.synopsys.com/software-integrity/resources/analyst-reports/2020-open-source-security-risk-analysis.html. (accessed: 30 September 2021).

Szulik, K., 2018. Open source is everywhere: Survey results part 1. Tidelift. URL https://blog.tidelift.com/open-source-is-everywhere-survey-results-part-1. (accessed: 30 September 2021).

Taibi, D., 2015. An empirical investigation on the motivations for the adoption of open source software. In: Proceedings of the 10th International Conference on Software Engineering Advances. ICSEA, IARIA, pp. 426–431.

The Linux Foundation, 2019. Openchain. The Linux Foundation, URL https://www.openchainproject.org/. (accessed: 30 September 2021).

Tidelift, 2019. The 2019 Tidelift Managed Open Source Survey Results. Tidelift, Inc., URL https://tidelift.com/subscription/managed-open-source-survey. (accessed: 30 September 2021).

TODO Goup, 2018. TODO Guides: Using Open Source Code. TODO Goup, URL https://github.com/todogroup/guides/blob/master/using-open-source.md. (accessed: 30 September 2021).

Yahav, I., Kenett, R.S., Bai, X., 2014. Risk based testing of open source software (OSS). In: 2014 IEEE 38th International Computer Software and Applications Conference Workshops. pp. 638–643. http://dx.doi.org/10.1109/COMPSACW.2014.107.

Yin, R.K., 2018. Case Study Research and Applications: Design and Methods, 6th ed. Sage Publications, Los Angeles, CA, USA.

Zhou, M., Mockus, A., Ma, X., Zhang, L., Mei, H., 2016. Inflow and retention in OSS communities with commercial involvement: A case study of three hybrid projects. ACM Trans. Softw. Eng. Methodol. 25, 13:1–13::29. http://dx.doi.org/10.1145/2876443.

Simon Butler received a Ph.D. in computing from The Open University in 2016. He is an associate senior lecturer at the University of Skövde and is a member of the Software Systems Research Group. His research interests include software engineering, open source software, program comprehension, software development tools and practices, and software maintenance.

Jonas Gamalielsson received a Ph.D. from Heriot Watt University in 2009. He is a senior lecturer at the University of Skövde and is a member of the Software Systems Research Group. He has conducted research related to free and open source software in a number of projects, and his research is reported in publications in a variety of international journals and conferences.

Professor Björn Lundell received a Ph.D. from the University of Exeter in 2001, and leads the Software Systems Research Group at the University of Skövde. Professor Lundell's research contributes to theory and practice in the software systems domain, in the area of open source and open standards related to the development, use, and procurement of software systems. His research addresses socio-technical challenges concerning software systems, and focuses on lock-in, interoperability, and longevity of systems. Professor Lundell is active in international and national research projects, and has contributed to guidelines and policies at national and EU levels.

Christoffer Brax received the M.Sc. degree from the University of Skövde in 2000, and a Ph.D. from Örebro University in 2011. He is a consultant with Combitech AB working in systems engineering, requirements management, systems design and architecture, and IT security. Christoffer has 18 years experience as a systems engineer.

Anders Mattsson received the M.Sc. degree from Chalmers University of Technology, Sweden, in 1989 and a Ph.D. in software engineering from the University of Limerick, Ireland in 2012. He has almost 30 years experience in software engineering and is currently R&D manager for Information Products and owner of the software development process at Husqvarna AB. Anders is particularly interested in strengthening software engineering practices in organisations. Special interests includes software architecture and model-driven development in the context of embedded real-time systems.

Tomas Gustavsson received the M.Sc. degree in Electrical and Computer Engineering from KTH Royal Institute of Technology in Stockholm in 1994. He is co-founder and current CTO of PrimeKey Solutions AB. Tomas has been researching and implementing public key infrastructure (PKI) systems for more than 24 years, and is founder and developer of the open source enterprise PKI project EJBCA, contributor to numerous open source projects, and a member of the board of Open Source Sweden. His goal is to enhance Internet and corporate security by introducing cost effective, efficient PKI.

Jonas Feist received the M.Sc. degree in Computer Science from the Institute of Technology at Linköping University in 1988. He is senior executive and co-founder of RedBridge AB, a computer consultancy business in Stockholm.

Bengt Kvarnström received the M.Sc. degree in Applied Physics and Electrical Engineering from LiU Institute of Technology in Linköping in 1981. He is a senior systems engineer at Saab Aeronautics and is the current leader of the group responsible of the Saab Processes, Methodology and Tools for software development.

Erik Lönroth holds an M.Sc. in Computer Science and is the Technical Responsible for the high performance computing area at Scania CV AB. He has been leading the technical development of four generations of super computing initiatives at Scania and their supporting subsystems. Erik frequently lectures on development of super computer environments for industry, open source software governance and HPC related topics.