On Company Contributions to Community Open Source Software Projects

Simon Butler, *Member, IEEE*, Jonas Gamalielsson, Björn Lundell, *Member, IEEE*, Christoffer Brax, Johan Sjöberg, Anders Mattsson, *Member, IEEE*, Tomas Gustavsson, Jonas Feist, and Erik Lönroth

Abstract—The majority of contributions to community open source software (OSS) projects are made by practitioners acting on behalf of companies and other organisations. Previous research has addressed the motivations of both individuals and companies to engage with OSS projects. However, limited research has been undertaken that examines and explains the practical mechanisms or work practices used by companies and their developers to pursue their commercial and technical objectives when engaging with OSS projects. This research investigates the variety of work practices used in public communication channels by company contributors to engage with and contribute to eight community OSS projects. Through interviews with contributors to the eight projects we draw on their experiences and insights to explore the motivations to use particular methods of contribution. We find that companies utilise work practices for contributing to community projects which are congruent with the circumstances and their capabilities that support their short- and long-term needs. We also find that companies contribute to community OSS projects in ways that may not always be apparent from public sources, such as employing core project developers, making donations, and joining project steering committees in order to advance strategic interests. The factors influencing contributor work practices can be complex and are often dynamic arising from considerations such as company and project structure, as well as technical concerns and commercial strategies. The business context in which software created by the OSS project is deployed is also found to influence contributor work practices.

Index Terms—Open source software, company contribution, work practices

1 Introduction

Open source software (OSS) is widely deployed in commercial software products and services [1], [2], [3] and within companies [4], [5], [6], and is used to support open innovation processes between companies [7], [8]. Given the level of integration into company products, processes and services, company software developers have long contributed to OSS projects for many reasons, including improvement of the quality of the software they use and a desire to influence the direction in which the software is developed [1], [8], [9], [10], [11].

Research on developers' contributions to OSS projects has focused on the motivation and behaviour of individuals [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], as well as the challenges of using the tools available to make technical contributions [22], [23], [24], [25], [26], [27]. A wide range of research on company engagement with and contribution to OSS projects has provided an understanding of the motivations of companies to use OSS and to work with projects, and their ways of working [8], [10], [11], [28], [29], [30], [31]. However, while some research illuminates company strategies when engaging with OSS projects [8], [28], [30], [31], [32], [33] often it is in the context of projects where the company has a controlling influence over the community and the direction of software development. In this work, we focus on engagement by companies and their employees and contractors with *community OSS projects*. By community OSS project we mean an OSS project managed by a foundation or otherwise collectively organised [34], where many contributors are professional practitioners directed by companies and other organisations who collaborate to create high quality software [1].

Decisions on how companies engage with community OSS projects are taken both by managers within each company and individual developers, and the majority of contributions are made directly by developers or other employees, who decide how to conduct each individual interaction with the project [28], [31], [34], [35], [36]. It can be inferred that many contributions made by companies are motivated by software development needs driven by business requirements. Further, we conjecture that practitioners commissioned by companies are working with technical, fiscal and temporal constraints within the business context that may not be apparent which also motivate contributions and how they are made. The context for decisions about

- *S. Butler, J. Gamalielsson, and B. Lundell are with the University of Skövde, Skövde 541 28, Sweden. E-mail: {simon.butler, jonas.gamalielsson, bjorn.lundell}@his.se.*
- *C. Brax is with Combitech AB, Linköping SE-580 15, Sweden. E-mail: christoffer.brax@combitech.se.*
- *J. Sjöberg is with Findwise AB, Göteborg 411 40, Sweden. E-mail: johan.sjoberg@findwise.com.*
- *A. Mattsson is with Husqvarna AB, Huskvarna SE-561 82, Sweden. E-mail: anders.mattsson@husqvarnagroup.com.*
- *T. Gustavsson is with PrimeKey Solutions AB, Stockholm 171 73, Sweden. E-mail: tomas.gustavsson@primekey.com.*
- *J. Feist is with RedBridge AB, Stockholm 111 20, Sweden. E-mail: jonas.feist@redbridge.se.*
- *E. Lönroth is with Scania IT AB, Södertälje 151 32, Sweden. E-mail: erik.lonroth@scania.com.*

Manuscript received 31 Aug. 2018; revised 14 May 2019; accepted 19 May 2019. Date of publication 17 June 2019; date of current version 16 July 2021. (Corresponding author: Simon Butler.)
Recommended for acceptance by E. Murphy-Hill.
Digital Object Identifier no. 10.1109/TSE.2019.2919305

engagement and contributions is also determined by the governance [37], [38] and licensing [39], [40] of the OSS project as well as the strategic interests of other participants in the project [31], [33]. Previous research has shown that when a company and individuals (owners, employees and contractors) affiliated with the company engage with an OSS project governed outside the companys control and specific development context, it is critical to adhere to established "work practices that are appreciated by community members" [41].

This article seeks to illuminate how companies engage with and contribute to OSS projects independent of company control and collaborate with other companies and organisations to achieve their own and common aims. To that end we ask the first research question:

RQ1: *How do companies contribute to community OSS projects?*

The context in which a company contributes to a community OSS project is framed by many factors, including the business and technical interests of the contributing business, as well as those of others in the community, and the governance systems of the OSS project. Less clear is how those factors combine to motivate the use of particular *work practices*, i.e., the specific methods or approaches used to collaborate in community OSS projects. For example, why a particular method of interaction might be chosen to achieve a given outcome. To explore the motivations of companies and practitioners working on their behalf to adopt specific work practices we ask a second research question:

RQ2: *What factors inform the selection of specific work practices used by companies to contribute to community OSS projects?*

To answer RQ1 we undertook an investigation of the public online records of eight community OSS projects (including mailing lists and issue trackers) to identify the opportunities to make contributions to each project and the work practices used to make the contributions. In addition we interviewed practitioners from companies engaged with the eight community projects investigated to obtain further information about the types of contributions made, particularly those that may not be apparent from public records. The interviews also explored the practitioners' motivations to use particular forms of contribution in order to answer RQ2. The interviews examined both the more strategic or policy level decisions about why a company might engage with an OSS project in a particular way, and the choices individuals working for companies make about how to make a contribution to an OSS project.

This research extends previous work [42] that investigated RQ1 in five community OSS projects. In this research the scope of the investigation is increased to include *all* contributions made to a project, expands the number of OSS projects studied from five to eight, and adds a second research question to examine motivation for the observed behaviour.

The remainder of the article is structured as follows. We first present background information and a review of the academic literature (Section 2). In Section 3 we outline the research methodology and give details of the selected projects in Section 4, including the governance mechanisms that frame project activity. In Section 5 we present results, which detail interactions between companies and OSS projects as well as the reasons given by developers for the approaches used. In Section 6 we analyse and discuss the results, and present the conclusions from the study in Section 7.

2 Background and Related Work

In a keynote presentation at ICSE 2017, the Executive Director of the Eclipse Foundation, Mike Milinkovich, explained how many companies strategically engage with OSS projects and claimed that "every software company is an open source company" [43]. Research shows that many companies in different sectors, utilise software that is developed and maintained by OSS projects external to the company [44]. Software created by OSS projects supports business activities, including software development, as part of revenue generating products and services, and as part of open innovation processes [7], [8]. Practitioners engaged with software deployed from well-known community OSS projects (including Linux and the Apache web server) and open source foundations (e.g., Eclipse Foundation and MariaDB Foundation) experience many business benefits, and at the same time encounter a number of challenges that companies need to overcome in order to engage successfully with and contribute to OSS projects [43], [45].

Community OSS projects, like Linux and the Apache web server, are organised for the mutual benefit of participants, in contrast to single vendor, or company controlled, OSS projects, where the OSS project is intended to benefit the controlling business [46]. The development of software by companies in open innovation processes mediated by community OSS projects has been described as "OSS 2.0" [1]. The benefits to businesses of contributing to OSS projects extend beyond the creation of software and include the acquisition of marketable knowledge and expertise [47], and organisational learning [8], [9], [48]. Individual contributors also benefit in terms of their careers [21] and their careers within projects [49], which also has value for the employer [21], [31].

2.1 Business and OSS Projects

Research on the interaction between companies, practitioners, and OSS projects has been undertaken from a variety of perspectives (See Table 1). Fitzgerald [1] articulated the idea that OSS development had evolved from being dominated by individuals to a process where businesses, and, particularly, professional software developers undertaking paid work on behalf of businesses, collaborate to develop software, and characterised it as OSS 2.0. The continuing growth and development of OSS 2.0 is exemplified by very large scale community OSS projects like OpenStack. Ågerfalk and Fitzgerald noted that businesses using and contributing to OSS projects take part in external collaboration with an "unknown workforce" [50]. A further observation made by Ågerfalk and Fitzgerald, and also by

TABLE 1
Previous Research Related to Business and Practitioner Engagement with OSS Projects

Research Topic	Synopsis	Publications
Business and OSS projects (Section 2.1)	Challenges and opportunities for businesses collaborating in OSS projects.	Dahlander et al. (2008) [30], Fitzgerald (2006) [1], Germonprez et al. (2013) [36], Lundell et al. (2017) [44], Riehle (2010) [46], Riehle (2011) [34], Ågerfalk et al. (2008) [50]
OSS Project Governance (Section 2.2)	Mechanisms used to organise OSS projects and foundations.	Alves et al. (2017) [51], Germonprez et al. (2014) [52], Markus (2007) [37], Shaikh et al. (2017) [38]
Business: Activity and Motivation (Section 2.3)	Examinations of business motivation for participation in OSS projects.	Andersen-Gott et al. (2012) [47], Bonaccorsi et al. (2006) [10], Germonprez et al. (2013) [36], Lakhani et al. (2003) [48], Lundell et al. (2010) [9], Munir et al.(2018) [8], Mäenpää et al. (2018) [53], Schaarschmidt (2015) [31]
	Factors limiting business contribution to OSS.	Linåker et al. (2018) [7], Lundell et al. (2013) [54], Morgan et al. (2014) [55], Zhang et al.(2017) [56], Zhang et al.(2018) [57]
Practitioners: Activity and Motivation (Section 2.4)	Studies of how and why individual contributors work with OSS projects.	Lerner and Tirole (2002) [12], Lin et al. (2017) [18], Pinto et al. (2016) [19], Riehle et al. (2015) [21], Schaarschmidt (2015) [31], van Wesel et al. (2017) [49], Zhou et al. (2015) [58]

Dahlander and Magnusson [30], was that the working relationships between companies within OSS projects are not governed by contracts that, for example, formally specify deliverables and delivery dates. Consequently, companies need to give careful consideration to the relationship between internal software development processes and engagement with strategically important community OSS projects. The strategic concerns, as Lundell et al. [44] argued, include the sustainability of both external OSS projects important to the business and the business itself.

Riehle [34], [46] identified the principles of the business models used by participants in community OSS projects, arguing that the software developed is often restricted to core non-differentiating functionality to which companies then add value to generate revenue. Further, Germonprez et al. [36] highlighted that community OSS projects are, sometimes unplanned, collaborations between competitors to create core software platforms.

2.2 OSS Project Governance

In the absence of contracts, governance processes used by OSS projects provide the basis for collaboration between businesses involved in OSS projects; regulating financial contributions, where they are permitted, and contributions from individuals working on behalf of the businesses. Research has examined forms of governance, and how governance facilitates both contributions and the activities of contributors. Markus [37] synthesised a set of core functions a governance system fulfils from a review of the academic literature. Informal aspects of governance found as norms within OSS projects are recognised, in addition to the formal aspects of governance recorded as rules. Four broad types of governance are identified by Germonprez et al. [52] that include meritocracies, as well as more flexible systems, and that these forms of governance can coexist within the same project. In a case study of the Linux kernel, Shaikh and Henfridsson [38] found that governance evolves within an OSS project and may also manifest itself as different co-existent forms. Furthermore, Shaikh and Henfridsson found evidence that the tools used to manage software development contribute to the project governance process [38]. Alves et al.'s [51] systematic review of the literature on governance systems supporting both open and proprietary software ecosystems identified three key aspects of governance systems: how the participants are able to create value from their contributions, the coordination of the activities of contributing organisations, and mechanisms used to balance between openness and control.

2.3 Business: Activity and Motivation

Bonaccorsi and Rossi [10] found that companies were motivated to contribute to OSS projects for economic and technological reasons, rather than the more altruistic motives sometimes ascribed to individuals. Germonprez et al. [36] described business motivations to use community OSS as a means of open collaboration to create core, non-differentiating software. The authors also identified that the process of collaboration is not always easy, but that participants gain clear economic benefits [36]. Activity in two community OSS projects governed by foundations and four OSS projects each controlled by a single company was compared by Mäenpää et al. [53] who found the foundation governed projects facilitated greater external collaboration through increased openness.

Businesses can also be motivated to participate actively in a project for strategic purposes. Schaarschmidt et al. found *resource deployment*—the deployment of company developers in the OSS project—was a common strategy in the community OSS projects investigated, particularly where company developers acquire *committer* or *core developer*[1] status [31].

The acquisition of knowledge is also an important benefit for businesses participating in OSS projects. The contribution to organisational learning was identified by practitioners interviewed by Lundell et al. [9]. The finding is supported by Munir et al. [8] who observed the value to a

1. We use the term *core developer* to refer to contributors who have *commit privileges* to the project version control system and therefore act as a gatekeeper.

company of knowledge interchange with OSS projects in an open innovation process. Andersen-Gott et al. [47] also highlighted that technical knowledge gained through contribution to an OSS project can be monetised by the business through the provision of complementary services. The finding is aligned with earlier work by Lakhani and von Hippel [48] which concluded that the main beneficiaries of help-giving in the Apache web server project were the help-givers themselves, who derived direct learning benefits from the experience.

Businesses can also be cautious about contributing to community OSS projects. A study of OpenStack by Zhang et al. [56], [57] considered the importance of dominance of the software development process by particular companies and the consequences for both the software created by the OSS project and the community. The authors concluded that there are advantages to single company dominance for software development, but that some smaller companies become reluctant to contribute to the development process because of concerns that they are providing free labour to support the efforts of the dominant companies [56], [57]. An investigation by Morgan and Finnegan [55] found tensions between senior managers' desire for the company to be self-reliant and the opportunity to derive business value from OSS. Lundell et al. [54] also found differing attitudes towards collaboration with OSS projects held by technical staff and management. Caution also extends to technical contributions. Linåker et al. [7] studied a model, developed within Sony Mobile, that supports decision-making concerning the contribution of internally developed enhancements to OSS projects. The concern for the company is to distinguish between source code that represents functionality that has business value, and code that can be contributed to the upstream OSS project.

2.4 Practitioners: Activity and Motivation

The motivation of individual developers to contribute to OSS projects has been studied extensively. Many authors, including Lerner and Tirole [12], have found that developers are motivated by "career considerations and ego gratification" [12]. More recent studies of developers working in community OSS projects continue to support this perspective. The motivating factor of a "career path" for company developers working on OpenStack was found by van Wesel et al. to be a strong influence on their activity [49], while Riehle [21] highlighted the value to developers' careers of contributing to OSS projects. Furthermore, Riehle [21] provided evidence of the value of core developers to businesses also identified by Schaarschmidt et al. [31]. However, the studies focused on the motivation of developers to work with OSS projects, and did not examine either how developers work to support the aims of the business they work for, or the commercial pressures and constraints on their activity.

The retention of contributors by OSS projects has also been a concern for researchers, particularly in the first few months of a contributor's activity. Zhou et al.'s study of company developers contributing to Gnome and Mozilla [58], for example, examined the characteristics of the behaviour of those who became long-term contributors, rather than making a small number of contributions at most. However, the study did not consider the motivations of company developers whose involvement with the project might be intended to complete a specific task for commercial reasons.

Further evidence of short-term or intermittent contribution by individuals is found in the research literature. For example, Pinto et al. [19] investigated contributors to GitHub projects that make single or very limited contributions, sometimes referred to as *drive-by*, or casual, contributions. Although the authors considered the motivations of contributors they did not consider commercial aspects in detail. However, they observed that some casual contributors were known to be longer-term contributors to other OSS projects. An investigation by Lin et al. [18] found contributors to five community OSS projects who created source code tended to have briefer relationships with projects than those who edited the project source code. The focus of the work was on developer turnover in projects and did not examine the reasons behind short-term contribution. A speculative explanation might be that some developers contribute source code to resolve an issue of significance to their employer, and once the task has been completed there is no business case to contribute further code.

To summarise, the academic literature identifies the business incentives to contribute to community OSS projects, and the principles of the business models used to generate revenue from participation. There is also evidence that businesses gain from participation in OSS projects in other ways, such as knowledge acquisition. Researchers have also identified that participation in OSS projects can be challenging for businesses. Much research acknowledges that contributors to OSS projects are mainly acting on behalf of businesses, and documents the career incentives for developers. However, few studies examine how such developers interact with OSS projects, particularly community projects, to undertake tasks in order to achieve business goals.

3 Research Design

In this article we report on a descriptive multi-case study [59] of a purposeful sample [60] of eight community OSS projects. We initially compiled a list of software created by OSS projects that is of strategic importance to the businesses represented by six of the authors. From that list, we selected eight OSS projects to investigate according to three fundamental criteria. First, the projects investigated are community OSS projects; that is they are neither exclusively controlled nor maintained by a single commercial entity, but are maintained independently, by independent foundations, or under the aegis of the Apache Software Foundation (ASF) and the Eclipse Foundation. Second, the projects are widely used in that they are deployed in, or support the development or provision of, products and services in multiple business; i.e., the projects studied provide software recognised by many businesses as appropriate for use in commercial contexts. Third, the projects have active communities of contributors, and histories measured in years. In addition, the software solutions implemented by the projects are not concentrated in a single domain and represent a variety of project types including open innovation and large-scale industry collaborations (See Table 2). Furthermore, the eight projects are also ones that seven of

TABLE 2
The Eight OSS Projects Investigated

	Description	Control	Licence	OSS Project Established
Apache CloudStack	A cloud computing system	Apache Software Foundation	Apache-2.0	2010
Apache Solr	A search engine platform	Apache Software Foundation	Apache-2.0	2006
Bouncy Castle	A cryptographic library	Legion of the Bouncy Castle Inc.	MIT	2000
Contiki-NG	An operating system for the Internet of things (IoT)	*independent*	BSD-3-Clause	2003
Leshan	A lightweight machine to machine (LWM2M) client and server for IoT	Eclipse Foundation	EPL-1.0 & EDL-1.0	2014
MariaDB Server	A database management system	MariaDB Foundation	GPL-2.0	1995
OpenStack	A compute component of a cloud computing system	OpenStack Foundation	Apache-2.0	2010
Papyrus	A UML and SysML modelling tool	Eclipse Foundation	EPL-1.0	2008

the authors have first hand experience of as users in commercial deployment contexts and as contributors. In summary, in this work we investigate company participation in OSS projects where there is a non-exclusive relationship between the contributor and the project, and the company must interact with other contributors—commercial entities and individuals—to achieve its goals.

3.1 Archival Investigation

We examined the public archival records [61] available for each of the eight projects to investigate the first research question. Beginning with the project website and any project pages on foundation websites, we identified the online resources that define the project and how it works, as well as the systems used to contribute to the project including mailing lists, changelogs, and bug tracking information (see Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2019.2919305 for details).

The interactions recorded in publicly available online resources for each project were analysed. First, to identify the forms of communication used to contribute to the projects and the types of contribution made; and, second, for evidence of both the manner in which interactions were conducted and their outcomes. The resources used by each project vary and those examined include mailing lists, online forums, and bug and issue trackers.

Three categories of activity in OSS projects reported in the academic literature were used as an *a priori* framework to identify the wider purpose of the contribution: *bug reports* [62], *feature requests* [62], and *support messages* or help requests [48], [63]. A fourth category was also used to capture events outside the scope of the other three categories, including project governance activity.

The characteristics of the work practices used by individuals to pursue their aims in each contribution were also identified and captured using an open coding process, informed by Glaser's ideas [64], [65]. The first author took main responsibility for the coding process and emerging codes were discussed and scrutinised by the first three authors as the coding progressed. A key principle adopted is the notion that, contingent on context, any unit of coding is acceptable. Accordingly codes are applied to email threads, issue tracker tickets, individual comments, emails, and to sentences to develop and refine abstract concepts grounded in the data to support reporting of observations. Evolving observations were discussed by all authors.

We followed an iterative coding process. Initially, one month of activity on the Apache Solr issue tracker and the four project mailing lists was considered in the analysis. Eleven more months of activity on Solr was subsequently considered so that the period between April 2017 and the end of March 2018 was analysed. Thereafter activity in the public communication channels of each project was considered for the same period, with the exception of OpenStack Nova. OpenStack Nova has a volume of activity and number of communication channels that is considerably greater than the other projects analysed. Three months of activity (May and October 2017, and January 2018) were selected at random for analysis.

3.2 Practitioner Interviews

To contribute detail to the first research question and to investigate the second research question we conducted open interviews with contributors to each project. Email invitations were sent to individuals identified as having contributed to each of the eight OSS projects in public communication channels as part of their employment. In total, seventeen respondents were interviewed; nine and eight respectively from the primary and secondary software sectors [66]. The majority of interviewees were employed to work in multiple roles, including non-technical roles. Sixteen interviewees, for example, spent at least part of their working week as a software developer, and some of those also had a role as a core developer in an OSS project as part of their paid work. Some interviewees had consultancy roles; both technical consultancy working with a specific OSS project, and providing bespoke solutions where the OSS formed part of the solution. Just less than half also had non-technical roles, for example in business management and in practitioner training. As well as experience of project governance as core developers, a few interviewees also had wider experience of foundation and project governance having had roles on steering committees and in foundation administration. Interviewees with software development roles all have several years of industry experience. Some have sought roles where they can contribute to OSS projects, while others work with OSS because of strategic decisions made by their employer.

TABLE 3
Governance Characteristics of the Investigated OSS Projects

	Ownership of Assets	Community Management	Software Development Processes	Conflict Resolution and Rule Changing	Use of Information and Tools
Apache CloudStack	ASF	Apache Way [67]	JIRA and developer mailing list	Bylaws of the ASF [68], ASF Code of Conduct [69] & Apache Way	Mailing lists, project wiki, JIRA and IRC
Apache Solr	ASF	Apache Way	JIRA and developer mailing list	Bylaws of the ASF, ASF Code of Conduct & Apache Way	Mailing lists, project wiki, JIRA and IRC
Bouncy Castle	Legion of the Bouncy Castle Inc.	*Not explicitly documented*	JIRA	*Not explicitly documented*	Mailing lists, JIRA, GitHub, and project wiki
Contiki-NG	The project	*Not explicitly documented*	GitHub	*Not explicitly documented*	Project wiki, GitHub, Gitter
Leshan	The project	Eclipse Community Code of Conduct [70]	GitHub and project mailing list	Bylaws of Eclipse Foundation [71] & Eclipse Community Code of Conduct	Mailing lists, GitHub and project wiki
MariaDB Server	MariaDB Foundation	Ubuntu Code of Conduct [72]	Maria Captains mailing list and JIRA	*Not explicitly documented*	Mailing lists, JIRA and GitHub
OpenStack	OpenStack Foundation	OpenStack Foundation Community Code of Conduct [73]	IRC, project mailing lists, Git, Gerrit, and Launchpad Blueprints	OpenStack Foundation Bylaws [74], Code of Conduct [75] and Community Code of Conduct	Mailing lists, IRC, Gerrit, Git, LaunchPad blueprints & bug tracker
Papyrus	The project	Eclipse Community Code of Conduct	Developer mailing list and Bugzilla	Bylaws of Eclipse Foundation & Eclipse Community Code of Conduct	Mailing lists, Bugzilla, Gerrit and Git, project wiki and forum

Interviews were conducted in English by the first author of which fourteen were conducted by telephone and three by email. English is the native language of the first author and four interviewees (all telephone interviewees), and a working language for thirteen interviewees. Interviewees were informed that reporting of the research will preserve anonymity for both the individuals and their employer so that they were able to discuss their experiences and motivations more freely.

Interviews initially probed for a generic understanding of the interviewee's work context through an opening question of the form: "*Please describe your involvement with [`OSS project`] and how that activity relates to your employment?*" For each interview a number of follow up questions were prepared to explore observed activities in which the interviewee had participated.

The telephone interviews were recorded and transcribed. Each interviewee was sent the transcript of the interview to check the transcription of the conversation and correct any misunderstandings that may have occurred during the conduct of the interview. Interviewees were also invited to expand on any points made during the interview, if they wished.

The approved interview transcripts were analysed using an open coding process (using the same open coding strategy as used in the archival investigation, see Section 3.1). A coding scheme was developed through analysis of the first five interview transcripts by the first author and evolved iteratively thereafter. A coding dictionary was maintained and the coding of interviews was reviewed following each interview through scrutiny by the first three authors as the systematic coding process progressed. Anonymised synopses of the interviews, including quotes, were discussed by all authors during a four month period to allow time for reflection. In retrospect, we found that after twelve interviews saturation was being reached because subsequent interviews gave very limited additional material apart from further examples supporting the evidence already gathered.

4 Cases and Case Characteristics

The eight OSS projects investigated are: Apache CloudStack, Apache Solr, Bouncy Castle, Contiki-NG, Eclipse Leshan, MariaDB Server, OpenStack and Papyrus (See Table 2). As well as having a specific technical mission, each project provides opportunities for both companies and individuals acting on their behalf to interact with the project. The governance model for each project defines the intellectual property mechanisms and communication systems through which contributions are made (See Table 3). In addition, contributions may be made to the governing foundation depending on its financial structure (See Table 4).

Apache CloudStack provides a cloud computing platform. Initially developed as a proprietary licensed product, CloudStack became an OSS project in 2010 and came under the control of the ASF in 2012 [76]. Apache Solr was initially proprietary licensed software that became an OSS project governed by the ASF in 2006. The close relationship with the Apache Lucene project led to the integration of the two projects in 2010. Bouncy Castle is a widely used

TABLE 4
The Legal Status of Foundations Controlling the Eight Projects Studied

Foundation	Established	Incorporation	Type	Financial Contribution
Apache Software Foundation	1999	US Title 26 Section 501(c)(3) [85]	For the public good	Donation, Sponsorship
Legion of the Bouncy Castle Inc.	2013	Australian Charity Law [86]	Charity	Donation
Eclipse Foundation	2004	US Title 26 Section 501(c)(1)	For mutual benefit	Sponsorship, Membership,
MariaDBFoundation	2012	Non-stock not-for-profit and incorporated in Delaware, USA [87]	Not for profit	Donation, Sponsorship
OpenStackFoundation	2012	US Title 26 Section 501(c)(1)	For mutual benefit	Membership

cryptographic library first released as an OSS project in 2000. Contiki-NG provides an operating system for small, low-powered devices [77]. Contiki-NG was forked[2] in 2017 from Contiki-OS, which was established as an OSS project in 2003. Contiki-OS remains an active OSS project, but has not released a version of the software since 2015. The Contiki-NG project is independent of company control and managed by its core developers. MariaDB Server is a fork of MySQL. Originally a closed source database management system, MySQL was established as an OSS project in 1995. MariaDB Server was forked from MySQL in 2009 and the MariaDB Foundation was created in 2012 to preserve the project's independence. Leshan and Papyrus are projects in the Eclipse ecosystem and are governed by the Eclipse Foundation. Leshan is an implementation of the Lightweight Machine to Machine (LWM2M) protocol [78] and is overseen by the Eclipse Internet of Things Working Group [79]. Papyrus is an industry-led project to create a UML and SysML modelling tool and has been an Eclipse Foundation project since before its initial release in 2008 [35], [80]. Papyrus is managed by the Papyrus Industry Consortium [81], a member of the PolarSys [82] working group, which oversees a number of projects focused on model driven engineering and embedded systems. OpenStack provides a platform that supports the provision of private and public clouds through virtual servers and software defined storage on heterogeneous hardware [83]. Development of OpenStack is supported by the Open Stack Foundation [84].

The governance of an OSS project outlines the management of the project itself and its assets, and defines the mechanisms through which the project publishes information, manages activities and receives contributions. Markus [37] identified six categories of formal and informal governance structures and rules found in OSS projects.

- *Ownership of Assets*: rules for ownership of intellectual property, foundation structure.
- *Chartering the Project*: the goals of the project.
- *Community Management*: rules pertaining to membership and the roles members may have.
- *Software Development Processes*: rules for requirements gathering, coordination, software changes and release management.
- *Conflict Resolution and Rule Changing*: rules concerning conflict resolution and changing rules.
- *Use of Information and Tools*: rules concerning communication, and the use of tools and repositories.

Table 3 provides an overview of the way each of the eight selected projects implements governance mechanisms for each category identified by Markus, with the exception of 'Chartering the Project' which all eight projects do through their websites and in other documentation, and is omitted from the table to avoid duplication. The MariaDB Server project, for example, states that the software is "... an enhanced, drop-in replacement for MySQL" and will remain open source [88].

The practical differences for individual contributors that arise from the governance mechanisms are in two main areas. First the communication channels and software development coordination tools used by the projects vary in number and complexity from Contiki-NG's use of a GitHub repository, to the multiple mailing lists, code review tools, planning system and internet relay chat channels used by OpenStack. Second, the mechanisms used by the projects to acquire the right to use and distribute contributed source code introduces a layer of bureaucracy in six projects. The ASF, Eclipse Foundation projects, and OpenStack ask individual contributors, and companies, for ASF projects and OpenStack, to complete a Contributor Licence Agreement (CLA) [89], [90], [91], [92][3,4] which gives the project or foundation a perpetual *copyright licence* for the contribution. The MariaDB CLA [93] is used for contributions made using version two of the GNU Public License (GPL v2) and relies on *copyright assignment*. The MariaDB CLA grants joint copyright at the point of contribution and, thus, appears to function similarly to a copyright licence. Contributions to MariaDB can also be made using the permissive MIT Licence. From a contributor's perspective, a CLA requires a business to approve the terms under which technical contributions are made on its behalf, and for practitioners acting for a company to seek that approval.

2. A *fork* occurs where an OSS project divides into two separate projects. There are many reasons why projects may fork including inactivity by core developers and disagreements over the direction of software development [41].

3. We refer to v2.0.0 of the ECA dated 2016-08-22, which was in force during the period of data collection for this research. v3.0.0 of the ECA was published in October 2018.

4. Access to the OpenStack CLA requires a Launchpad account.

The licences under which the source code for each project is distributed place varying obligations on users that modify the source code, and few restrictions on users deploying the software. The more permissive licences—BSD 3-Clause, the Eclipse Distribution Licence (EDL) and MIT—place yet fewer obligations on users of the software, allowing industrial users to integrate the code with their own solutions and products.

Companies and individuals may also contribute to the foundations financially, or in kind, and the nature of the relationships formed between the foundations and the donors is related to the legal status of each foundation (See Table 4). The Legion of the Bouncy Castle Inc. supports the development of the Bouncy Castle library through donations from companies and individuals [94]. The ASF uses sponsorship to pay for the infrastructure used by projects, accounting and legal costs, as well as marketing [95]. The Eclipse, MariaDB and OpenStack Foundations offer a range of membership types through which companies and organisations are able to influence the direction of software development to varying degrees [96], [97], [98], [99].

5 Results

In this section we report on interactions between practitioners representing companies and OSS projects. First we report observations of *how* businesses interact with OSS projects when making contributions. We then report on *why* software developers, both core and non-core, use specific work practices, and their employers' motivations to contribute to the OSS projects.

5.1 Work Practices Used to Contribute to OSS Projects

Governance frames the activities of contributors to OSS projects and the public communication systems through which they contribute to projects. Through examination of interactions found in archives of the collaboration platforms used by the projects[5] we identified work practices used to contribute to OSS projects.

5.1.1 Bug Reporting and Fixing

Practitioners adopt two fundamental approaches when reporting bugs. One approach is to ask an exploratory question on a mailing list or in a forum. The question typically enquires about some functionality to ensure that the submitter understands how the software is intended to work and whether the observed behaviour is to be expected, the result of error on their part, or a genuine issue (See Table 5, Example 1). Often the contributor will include precise details of the software, hardware and operating system they are using to provide context. Where the issue is identified as a bug it may then be reported via the issue tracker, either by the original contributor, or by a core developer. Where the contributor is more certain they have identified a problem with the software a bug report is submitted to the mailing list or issue tracker with supporting evidence (Example 2).

5. Appendix A, available in the online supplemental material lists the data sources and archives examined.

Bug fixes are contributed to projects in response to an identified fault and, sometimes, with a bug report. The mechanism for contributing bug fixes varies according to the tools projects use and the project workflow. For example, a bug report accompanied by source code may be submitted as a pull request on GitHub. Many projects prefer that bug fixes are submitted with unit tests that establish both the problem and that the bug fix solves the problem and to support regression testing (Example 3). Where unit tests have been omitted from a proposed bug fix the core developers will often request unit tests to support fixes as part of the code review process.

Interactions with an OSS project require the input of company resources, including staff time, and, thus, are a financial cost for the business. Many reports of issues consist of a few steps and messages exchanged between the contributor and the OSS project (as in Example 2). However, sometimes, despite details being provided the core and non-core developers manage to misunderstand each other, which may require further contribution of time by both parties (Example 4). Contributions in larger projects can also be forgotten or take a long time to be integrated into the code base, especially when they are not a priority for core developers, and, thus, have the potential to become wasted effort for the contributor (Example 5).

5.1.2 Feature Requests

Non-core contributors also make requests to add new features to OSS projects. As with reporting bugs, the opportunities available for the initial approach include an exploratory question on a mailing list, in a forum or as a GitHub issue, so that the contributor can understand whether the project would be receptive to the proposed feature (Example 6).

Sometimes, as with bug fixes, non-core developers will submit a feature request with the source code that implements the feature. The implementation represents an investment of resources by the contributing company. Typically, as with code submitted to fix bugs, the developer will take part in a review process and revise the code to ensure that it meets the core developers' requirements before it is integrated into the code base. There are also occasions where contributors do not take part in the review process and the code, depending on project policy, often does not get merged into the project. It is uncommon, but significant amounts of software development work (sometimes of the order of thousands of lines of code) can be submitted and abandoned by their contributors. In some cases, the core developers have reviewed the contribution, the requested revisions are not made, and the contributor does not respond to further messages (Example 7).

Review comments on source code submitted by non-core contributors are, in many OSS communities, mostly made by the core developers who have to accept the code. Sometimes, however, commercially unrelated non-core developers will contribute to code reviews and we infer that there are likely to be strong motivations for what appear to be unconventional actions. Observed interactions include the addition of relatively minor technical points that may improve the maintainability of the code (Example 7). We

TABLE 5
Sources of Illustrative Examples of Work Practices Given in Section 5.1

	Summary	Source
Example 1	A user reports that a Papyrus plugin is not loading through a tentative query in the Papyrus forum. The response confirms that there is indeed an issue and the bug is fixed.	https://www.eclipse.org/forums/index.php?t=msg&th=1087822&goto=1769255&#msg_1769255
Example 2	A non-core developer reports a key parser bug in Bouncy Castle. The bug report has plenty of supporting evidence. A core developer responds saying the bug has been fixed in the development branch and will be part of the next beta release.	Bouncy Castle (JIRA): BJA-685 http://www.bouncycastle.org/jira/browse/BJA-685
Example 3	A bug report was submitted to Solr by a core developer, with a failing unit test to demonstrate the issue. The core developer then implements a solution, with unit tests.	Apache Solr (JIRA): SOLR-10908 https://issues.apache.org/jira/browse/SOLR-10908
Example 4	A developer asks for a feature, which is implemented by the core developers and released. The implementation is not the feature requested and time is spent by both core and non-core developer to identify and implement the desired functionality.	Bouncy Castle (GitHub): pull request #234 https://github.com/bcgit/bc-java/issues/234
Example 5	A pull request implementing a minor change is made. Over a year later the pull request has not been merged into the code base.	MariaDB (GitHub): pull request #387 https://github.com/MariaDB/server/pull/387
Example 6	A non-core developer asks, in a GitHub issue, about his understanding of functionality he needs to add to Leshan. He checks with the core developers whether he is correct to think that the functionality has not been developed, and asks if anyone is working on the feature?	Leshan (GitHub): pull request #369 https://github.com/eclipse/leshan/issues/369
Example 7	A pull request implementing a plugin to support a hardware device is submitted by a non-core developer. After code review, and some requests from core developers, the submitter becomes silent.	Apache CloudStack (JIRA): pull request #2105 https://github.com/apache/cloudstack/pull/2105
Example 8	A non-core developer submits a new feature. During the code review, a committer, who is not part of the review process, but has a related use case in his business asks for a more generic solution. The code is revised.	Apache Solr (JIRA): SOLR-10783 https://issues.apache.org/jira/browse/SOLR-10783
Example 9	A blueprint for OpenStack Nova defines the task of removing a testing framework and links to activities that complete the task.	https://blueprints.launchpad.net/nova/+spec/mox-removal
Example 10	A core developer proposes an improvement to the system management API. A discussion between a second core developer and a potential implementer starts.	Apache CloudStack (JIRA): CLOUDSTACK-10262 https://issues.apache.org/jira/browse/CLOUDSTACK-10262
Example 11	A non-core developer asks a question about database cluster replication. Another non-core developer responds giving an answer.	https://lists.launchpad.net/maria-discuss/msg04843.html
Example 12	A user asks for help on a specific issue. Before receiving a response, he continues to work on the problem and adds further evidence to the mailing list thread.	Solr users mailing list http://mail-archives.apache.org/mod_mbox/lucene-solr-user/201711.mbox/<SY3PR01MB1546D5F756F046F96ECCE2B3C85F0@SY3PR01MB1546.ausprd01.prod.outlook.com>
Example 13	A developer encounters a problem when running Solr and creates a JIRA issue to ask for help. A core developer indicates that help requests should be made on the user mailing list.	Apache Solr (JIRA): SOLR-10820, SOLR-11002 https://issues.apache.org/jira/browse/SOLR-10820https://issues.apache.org/jira/browse/SOLR-11002
Example 14	A core developer uses a JIRA issue to coordinate a revision of the documentation for Solr.	Apache Solr (JIRA): SOLR-10842 https://issues.apache.org/jira/browse/SOLR-10842
Example 15	Leshan developers discuss the consequences of a change to the Eclipse Foundation's intellectual property rules.	https://www.eclipse.org/mhonarc/lists/leshan-dev/msg00851.html
Example 16	A developer asks about revising an uncompleted patch to implement proposed functionality, and coordinates activity with another developer.	http://eavesdrop.openstack.org/irclogs/%23openstack-nova/%23openstack-nova.2017-10-11.log.html#t2017-10-11T15:35:54

also observed a core developer not directly involved in the code review process intervene when they recognise a more generic use of a new feature that helps their employer's use of the software (Example 8).

OpenStack Nova uses a different process for making a feature request. The most common workflow is that an idea is discussed initially in the IRC channel and a proposal subsequently developed and added as a *blueprint* on the Launchpad platform (Example 9). The blueprint can be reviewed by the community to ensure that the proposal is relevant for the project, that the proposed implementation is sound and does not duplicate or restrict existing functionality, or overlap with other proposals. The blueprints also support traceability of the feature implementation and associated code reviews.

Core developers also make feature requests. Mostly, feature requests made by core developers are openly documented alongside those made by non-core developers in the projects investigated, including Leshan, OpenStack, and Solr. Feature requests are made openly so that other members of the project community can understand what functionality and implementation is being proposed, that the idea is sound, and welcome, and to identify whether the idea has been considered previously (Example 10). Furthermore, documenting intended areas of development may attract potential contributors.

5.1.3 Support

Project documentation may be incomplete, misunderstood, not read closely, or possibly out of date. Consequently, users of the software often seek help on mailing lists and in forums, and both core and non-core contributors provide support (Example 11). As software users, those giving help

have detailed collective knowledge of the deployment and use of the software.

Mailing lists and issue trackers are asynchronous communication channels. Sometimes practitioners seeking help may have solved a problem, or have continued working on it, while waiting for a response. In such cases, practitioners can provide a commentary on their ongoing problem solving activity and the outcomes (Example 12).

Occasionally, contributors make mistakes and take actions in communications channels that the core developers may not anticipate, despite extensive documentation of how the core developers expect contributors to behave. Typically, users would be expected to ask for help on a mailing list and only to use the issue tracker when a bug has been identified. However, on occasion, help questions can be posted directly to the issue tracker (Example 13). The response of the core developers to such errors varies from project to project.

5.1.4 Other Activities

While the literature reports three groupings of contribution made in the communication channels of OSS projects, those active in OSS projects contribute in other ways. Three main additional types of activity are observed. The first is the creation and maintenance of documentation. A second type of activities relates to project governance and administration, and community building. Third there are opportunities for additional activity in some larger projects as a consequence of the additional communication channels and tools available to developers.

Documentation activity in OSS projects can be seen as two processes. There is a deliberate effort to create and maintain documentation of the software. In some larger projects, such as Solr, there is often at least one person who focuses on managing the documentation effort (Example 14). The other form of documentation is a knowledge maintenance task undertaken by contributors—mostly core developers—that occurs during other activities such as providing support, fixing bugs, and feature implementation. While working on the primary task, links or connections are made to related items in the issue tracker, and sometimes to sources of information outside the project. Recording connections between mailing list threads, issues and other items annotates and connects knowledge within the communication and software development systems to create a detailed, emergent documentation of the project.

Project governance activities and opportunities differ between foundations and projects. ASF project core developers, for example, vote to accept a release candidate as the next release. ASF project management committees also vote privately to appoint new committers, who are subsequently introduced to the community on the developer mailing list. A similar process happens when new *core reviewers* are appointed for OpenStack Nova. Occasionally, projects need to discuss issues such as the impact of the foundation's intellectual property rules on the project (Example 15). The core developers in most of the projects studied are responsible for ensuring contributors have completed the appropriate CLAs when making their initial technical contribution.

In addition, OpenStack Nova, as noted, uses IRC extensively and makes logs available so others can, as with email threads, follow discussions and understand the discussion leading to a specific decision. IRC is also used to help coordinate other aspects of OpenStack development in close to real time (Example 16), as well as a channel for automated messages from the Gerrit code review tool, and to organise or invite review for particular code revisions. Also within OpenStack are a number of special interest groups (SIG)[6] that developers with common interests use to coordinate their activities across the project.

5.2 Motivations to Adopt Contribution Strategies

In this and the following subsection we report on analysis of the data collected from interviews conducted with contributors to the eight OSS projects studied. This subsection focuses on the strategic choices made about company contributions to OSS projects, and the following subsection (Section 5.3) focuses on why practitioners use individual work practices. In both subsections we report additional work practices, or aspects of those reported above, which were uncovered during the interviews.

A variety of factors influencing the type and extent of company engagement with the eight community OSS projects emerged from analysis of the collected interview data (See Table 6). Two key factors influencing the extent of engagement are the relationship between the company's business model and the deployment of the project software, and the maturity of the domain and the software. Another factor identified is the influence of the location of knowledge and expertise within the project and the contributing business on the manner in which companies contribute to OSS projects.

Some contributors to OpenStack and Apache CloudStack worked 90 percent or more of their time on the OSS project. In each case the company involved deploys the software as a key component of one or more of the company's revenue streams. In one business model, for example, the software is deployed to customers as a platform for them to deliver their services and products, in another the business deploys the software as a platform to support service delivery to their customers, as well as being a platform their customers could also add value to. The cost to the businesses of switching to different software would be considerable, or even existential, as stressed by one interviewee:

> *"... if the project dies then basically our company dies because the core business of the company is based on [the OSS project]."*

Where the business's product is part of the OSS project, the business adds value to the OSS by developing functionality for itself or a client, and contributes enhancements to the OSS project, perhaps identifying possible bugs as well. Some other businesses contributing to OpenStack and Apache CloudStack use a more *product-focused* strategy. Product-focused companies are similarly reliant on deployment of software from the OSS project to deliver services to their customers. While the contributors, the company developers, still work on the upstream project their main focus is

6. The OpenStack SIGs are listed at https://wiki.openstack.org/wiki/OpenStack_SIGsand include activities related to supporting newcomers, and high performance computing

TABLE 6
Motivating Factors Influencing the Type and Extent of OSS Project Engagement

Factor	Description	Illustrative Observations
Deployment	The business context in which software from OSS projects is deployed to generate revenue.	OSS is deployed to generate revenue by delivering functionality or service to customers. Revenue generated through adding value with software that depends on, adapts, or enhances OSS. OSS project as an open innovation platform and revenue is generated by other company products.
Software and Domain Maturity	The technical context of the OSS project software, including the factors contributing to the evolution of the domain and the pressures for continuing software development.	Domain: evolves through external pressures e.g., technology change. Software: additional functionality required by user. Software: incomplete implementation of specification.
Knowledge and Expertise	The knowledge and expertise required to deploy and develop software is not evenly distributed in OSS project communities.	Core developers have implementation expertise. Other contributing businesses may have the required implementation or domain expertise. Users of an OSS project have extensive experience of deploying the software.

on the product they develop within the business. The product is integrated with or dependent on the software created by the OSS project, and company developers contribute features in the project software and fix bugs to meet specific requirements to support their product.

The high level of confidence that businesses place in the capabilities and initiative of some individuals at the centre of their engagement with OSS projects was also reflected in the qualitative analysis. Some developers were given a great deal of license by their employer to work on the OSS project while delivering value to the business. For example, one interviewee commented that around 90 percent of their work on the OSS project was not specifically requested or directed by the company. The employer was said to have decided to invest in the OSS project to improve the quality and extend the functionality of the software, which is integral to the business. As the interviewee observed:

> *"If we work towards code quality and community building, [the project] will become more attractive for other developers."*

By encouraging wider deployment and participation the company anticipates maintenance costs will be reduced in the long term.

In addition, interviewees suggested other reasons for the intensity of the company's engagement with CloudStack or OpenStack including the relative size of the project and the relatively fast pace of software development, as well as the evolution of cloud services and the cloud domain. Two developers commented on the reciprocated development effort between company and OSS project as mutually beneficial, with one saying:

> *"... we provide contributions to [the OSS project] at the same time that we get benefits from the community."*

Another described the process as follows:

> *"... we execute the change in our branch and then we push to upstream. We work with a custom fork of [the project], which enables us to customize it to our needs and to create new features faster. We do have a roadmap to migrate back to upstream versions. Then, we fork again, and so on."*

The IoT sector, for example, follows a similar open innovation model to that seen in CloudStack and OpenStack by developing standards compliant infrastructure, such as communication protocol stacks, in OSS projects. The business model identified from analysis of interview data is product-focused. Companies collaborate in OSS projects, such as Leshan, to implement infrastructure that complies with established technical standards. Accordingly, development and maintenance costs for the communication systems are shared between companies contributing to the OSS projects. Individual businesses generate revenue through the development of connected products. Importantly, those products are interoperable because they have been developed to use standards-based implementations of supporting infrastructure. Furthermore, the development of new products is supported through the provision of reference implementations of communication and server infrastructure. In addition to businesses developing products, some companies employ developers to work on open innovation projects, though not exclusively on a single project.[7]

A third group of businesses are identified from analysis of the interview data. The companies can be less directly engaged with OSS projects as a consequence of their business model. Interviewees working for consultancies, and as individual consultants, for example, tended to interact intermittently with the OSS project, mostly when deploying software for a client or maintaining an existing installation. Although their business model is dependent on software from the OSS project, the need to interact with the project is reduced because of the manner in which the software is deployed. Typically the interaction consists of reporting issues or asking questions on the user mailing list. However, as some interviewees explained, there are occasions where

7. For example, the approach used by Bosch Software Innovations GmbH is outlined as part of a talk at FOSDEM 2018 (https://fosdem.org/2018/schedule/event/eclipse_iot/)

TABLE 7
Interviewees' Motivations to Adopt Specific Work Practices

Activity	Work Practice	Illustrative Observations of Motivation
Bug Reporting	Preparation of question	To ensure observed behaviour does not have a known explanation and solution. To develop understanding of the problem and explain it clearly.
	Tentative bug report	To acknowledge potential knowledge gap between reporter and core. To protect reputation in project community by allowing for error.
Feature Requests	Feature proposal	To ensure relevance of proposed contribution. To prevent unnecessary work by contributor.
	Code review	To ensure suitability of contributed code. To support and encourage new contributors. To document project source code quality expectations.
Support	Help-seeking	To identify a solution to a problem. To indicate continuing use of feature under threat of deprecation. To document a problem for client.
	Help-giving	To encourage software use. To identify bugs that might lie behind support requests. To gain knowledge and develop skills to support customers.
Other Activities	Documentation	To record project processes for fellow core developers.
	Source code maintenance	To ensure tasks that do not attract non-core developers are completed.
	Governance	To provide a layer of project oversight.

they develop software for the OSS project to support their particular use case and contribute that code back to the project, if appropriate.

The mechanisms available for businesses to contribute to an OSS project obviously influence the types of contributions that can be made. Analysis of the data collected from interviews also reveals constraints and opportunities within both businesses and OSS projects that shape contribution strategies. The value of participation in project governance to their employers, and in particular the enhancement of company reputation through active involvement in project governance, was a strong influence for some interviewees. One interviewee, however, provided note of caution through a critique of company involvement in foundations and project steering committees, arguing that tensions within larger companies around budgets and company aims often mean that company commitment to projects does not always fully reflect the business's strategic interests.

Companies may also adopt strategies that support a non-technical effort made by the project which brings business benefits to the contributor. The Bouncy Castle library, for example, is deployed in some operational contexts where software must be certified as meeting the Federal Information Processing Standards (FIPS). An interviewee described how the FIPS certification process is expensive and companies donate money to the Legion of the Bouncy Castle Inc. to contribute towards the payment of FIPS certification fees. Furthermore, other interviewees commented that financial contributions, or contributions in kind can be more relevant for the project or foundation in addition to technical contributions, in some circumstances, and can form part of a strategy to support the OSS projects the business uses. One example highlighted by interviewees is the contribution of money through foundation memberships to finance the computing infrastructure required for the development of OpenStack.

The location of expertise and knowledge within the business and the project also emerged, during analysis of interview data, as a factor influencing decisions about contributing to an OSS project. Constraints on a business in terms of expertise, knowledge or personnel required to contribute can mean that sometimes it can be more cost effective to commission work through consultancy and software development companies already engaged with a project to fix bugs or develop software for the OSS project. Committers for the two ASF projects studied, for example, are mostly company developers paid to work on the project. Some businesses employing core developers sell services and support for software from the OSS project. One interviewee spoke of their employer having a support contract for much of the work with the upstream project, but added that they also undertake some smaller tasks in-house and submit bug fixes and feature requests as well. A slightly different model is found with Bouncy Castle and MariaDB Server where considerable domain expertise is often required to work with the source code. Both projects receive technical contributions from their respective communities, including larger companies. In addition, Crypto Workshop, a company run by the founders of Bouncy Castle, sells support subscriptions and undertakes commissioned work on the project source code. The MariaDB Corporation also undertakes commissioned work to develop additional functionality. Interviewees commented that acquiring the necessary expertise to create technical contributions can be prohibitively expensive, or cannot produce the timely solution required. Accordingly, paying for established expertise can be a cost effective means of contributing code to the project and gaining the required functionality.

5.3 Motivations to Use Specific Work Practices

5.3.1 Bug Reporting

The care taken and attention to detail by developers contributing bug reports was a theme identified during analysis of the interviews (See Table 7). An interviewee explained that bug reporting was often a slow and painstaking process.

They highlighted the challenge of gathering information from different parts of the project documentation and systems such as issue trackers, as well as external sources including question and answer sites like Stack Overflow, to determine whether the observed problem had been seen previously, and potentially resolved. The same interviewee asserted that only then would they ask a question on the appropriate mailing list. Even at that stage, they explained, the question would employ the use of negative politeness in a formula such as "...or have I missed something?", so that the possibility of an error or oversight is allowed for to protect the reporter's community reputation.

Another interviewee emphasised the value of bug reports for improving project documentation. In their experience—both as a core developer and contributor—some bug reports arise because software behaviour found by users differs from the documentation. They also explained that where standards are implemented, bug reports can help identify and document cogent misinterpretations and misunderstandings of the standard.

As well as the desire expressed by bug reporters to report their observations clearly, interview analysis found that core developers also needed clear evidence in bug reports. Without clear evidence to help identify the underlying cause they found the process of determining the nature of the problem, and possible solutions to be considerably more challenging, and often time-consuming. Some request specific forms of evidence are included in bug reports, but that it is not always supplied. One core developer interviewed, however, remained relatively optimistic:

> *"... we have a template ... please tell us which branch, please give us the logs. And it's frequently ignored. But, OK, if it's ignored and we can handle the issue in another way it's OK."*

5.3.2 Feature Requests

Qualitative evaluation of the collected interview data found some developers are often motivated to submit feature requests to migrate already implemented features into the OSS project. Often the feature has been implemented and tested within the contributor's private version of the OSS project source code, and the feature would be easier to manage if it were incorporated in the upstream source code and released as part of the software from OSS project. One interviewee explained the challenging and expensive process of needing to integrate local code revisions into each release of the upstream project software to introduce functionality required before the company could deploy the software from the OSS project to customers. The effort required to maintain their own version of the upstream project's source code and make revisions to each release to integrate their own code introduced an unacceptable level of effort and cost for the business. By having their features incorporated into the upstream project, future revisions to the project would not break the code, and the company would not have the overhead of reintegrating code to each release. The interviewee observed that the process of having the feature accepted and integrated in the OSS project had taken a long time and had been achieved in a series of steps, rather than as a single feature request. They also emphasised the importance of the project to the business, saying:

> *"We had to do some fairly awkward things ... to continue being able to produce what we needed to produce and to make a saleable product."*

Depending on the project infrastructure, there are opportunities to negotiate with core developers and other users about proposed features to understand whether the proposed feature is likely to be accepted. Some interviewees commented that discussion of proposals of new or extended functionality was an effective process for scrutinising proposals and revising them to ensure the quality of the proposed contribution, and its acceptability to the rest of the community. Another interviewee added that working in relevant areas of the project source code that other contributors appeared not to be interested in increased the likelihood of features being accepted.

Part of the process of submitting a feature request is the code review work undertaken by the core developers prior to accepting and integrating the feature. In a large project such as OpenStack there is an extensive process of code review undertaken by developers from different contributing companies. A theme that emerged from the analysis of the collected interview data was the value of the review process as a means of preventing unwanted or undesirable changes, and supporting the longevity of the project. Interviewees also noted the need for vigilance during reviews to ensure that implementations were sufficiently generic so that the project remained useful to the wider user community and that new features were implemented for all supported platforms. Core developers also highlighted the value of a supportive and educational review process. In particular that the contributing developer should to be encouraged to continue contributing to support the longevity of the community and the software created by the project. Two core developers also commented that supportive code reviews can require additional effort and might seem an inefficient use of time. However, they both emphasised the value to the OSS project of investing time, with one saying:

> *"Is it worth it? Personally, I feel that code contribution integration is often less efficient than if I coded it myself but this is normal as contributors need to gain skills on the project, this is a kind of investment. For the longevity of the project, it's important to have more people involved."*

An additional benefit of code reviews, explained by one interviewee, is that they document the core developers' expectations for source code quality, and, in their opinion, potentially, influence the quality of future contributions.

One interviewee also drew attention to the challenges of processing large feature requests, explaining that the larger a submitted feature was the more difficult it was to review. In practice they preferred submissions to consist of smaller features so that each could be better understood, tested in isolation, and integrated more easily. Non-core contributors with limited experience of OSS reported finding the practice a challenge at first.

Interviewees working as core developers also identified that some software has additional requirements that are not always apparent to contributors of code. Additional

considerations can consequently make integration of contributions time-consuming and challenging. Examples given by interviewees include the security aspects of Bouncy Castle, Contiki-NG and MariaDB, and differences between the virtualisation models implemented on hardware platforms used for CloudStack and OpenStack installations.

5.3.3 Support

The definition of support activities used earlier includes both asking and answering questions as well as the provision of documentation, both for fellow contributors and end users.

Some subtleties of the process of asking for and providing help in forums and on mailing lists were illuminated by analysis of the interviews. Rather than simply asking for help, help requests can be considerably richer and have multiple intended audiences. One interviewee explained having used a mailing list question about a potential bug:

> *"... to document to the project that there are people still using it [the functionality] and there are likely to be people using it for quite some time."*

Furthermore, the same interviewee reported using questions about potential bugs and possible solutions to document for their clients that the company was making progress towards resolving the issue.

Additional uses of mailing lists were identified through the qualitative analysis. Mailing lists are not just help forums, or places to make announcements, but can also be a practical means of disseminating information. One interviewee saw the provision of an email summarising decisions made in different communication channels as helpful for those involved in one particular area of development. Furthermore, they argued that while a variety of communication channels enhanced interaction in large projects, it also created problems for information management, particularly in the sense of curating knowledge of the project's evolution.

Analysis also found both help-givers and help-seekers value the learning process required to formulate and respond to mailing list questions. One help-seeker explained the value of preparing questions to their working life saying:

> *"... part of the motivation is that I found it to be a useful way of fixing problems. So I probably write twice as many questions with the intention of posting them to a mailing list as I actually post."*

They then elaborated:

> *"... by the time you have formulated a good question and collected all the information to say what the problem is then the process of asking the question will often make the answer become clear."*

A consultant explained the value of reading mailing lists and providing help as a way of acquiring knowledge and skills. As well as learning the soft skills required to help others, the interviewee identified an additional benefit to their professional practice as:

> *"... learning about the problems that other people face so that when I run into similar problems with the consultancy work I can remember the problem."*

5.3.4 Other Activities

We also found that core developers, in particular, but not exclusively, engaged in a wide range of activities, both technical and non-technical, to support the longevity of the project. An interviewee explained the value of documenting project processes, "... because then anyone else can take over parts of the process when someone leaves ..." Three more interviewees highlighted, from their experience as core developers, the importance of undertaking basic software maintenance tasks and code quality improvement activities, such as fixing some bugs, refactoring code and identifying unused portions of source code for deletion. One commented about the motivation of contributors to maintain code:

> *"... if there is a very well-known bug that somebody needs to sit and fix: for some people it's not part of the day-to-day job so they will not do it. ... sometimes it means that we end up getting lots of nice cool new things, but there's that old thing [bug] back there that nobody is looking at."*

From the interview analysis we also identified motivations for other forms of contribution. Some, such as contributions to project governance, for example, depend on the opportunities provided by the project and foundation structure. Projects including Eclipse Foundation projects and OpenStack have steering committees that help determine the direction of software development. One interviewee emphasised the value of strong steering committees to projects in providing a layer of oversight. OpenStack is a very large project and has forms of internal organisation and structures that many other projects do not. For example there are a number of SIGs for cross-cutting, project-wide concerns, such as security. One OpenStack developer interviewed stressed participation in the SIGs as a valuable aspect of their work because they provide input into technical contributions in the form of trying to standardise development approaches across the project.

6 Analysis

The combination of observations and analysis of project archives and rich insights and experiences of how experienced contributors work with community OSS projects provides rich accounts of work practices used, as well as explanations of why the observed approaches are used. In this section we elaborate on the variation in type, extent and intensity of interaction between company practitioners and OSS projects. We also identify how the nature of some contributions represent an investment in the project by the business, and analyse how costs and availability of resources can influence the way that businesses and practitioners contribute to community OSS projects.

6.1 Type, Extent and Intensity of Interaction

A wide variation in the extent and intensity of engagement between individual practitioners and companies, and the OSS projects was observed. Some practitioners spend a large proportion of their working time directly on a project while others interact with projects less often. The form and the intensity of the engagement with the OSS project appear to be largely related to how the business adds value to the project software.

The software might be deployed as a component of a product or service that directly generates revenue, for example, or the business may add value by applying their expertise to deploy, and perhaps manage, solutions for customers that include the OSS project software.

An additional factor identified by interviewees was the critical nature of the OSS project software to the business model of the company they worked for. For some practitioners there was no viable alternative software solution. Their interactions with the project focused on the two objectives of delivering a viable product or technical solution now, despite difficult challenges, and working towards a more cost effective solution for the business in the future.

Where companies provide consultancy services to support deployment of the project software then their interactions with the OSS project may be infrequent and limited largely to help-seeking and bug reporting. The company's main requirement in such situation is reliable software for their customers to use and, perhaps, some additional knowledge of how to deploy the software. Companies that add value by incorporating the OSS in a product or service will have similar needs to those adding value through consultancy services, but also may need to add or improve functionality to support their work. It is notable, however, that many companies deploying OSS components in products and services do not engage with or contribute to some, if not most, of the OSS projects whose software they use. The first reason is that the component is perceived to be a commodity, or is used as if it were one. For example a specific version of a component may be deployed and only reliable, fully tested functionality is used. The second reason is that the component is replaceable.

Amongst the interviewees there was also variation in the intensity of engagement with the OSS project. Some contributors, especially in the cloud domain, spent a great deal of their time working on the project software either to improve the software or to add functionality required by products they were developing that built on the OSS project software. The reason for the intense or extensive engagement with the project seemed to be related to the domain, as the greatest intensity of activity was seen with contributors to CloudStack and OpenStack. A contributing factor reported by interviewees was the speed of product development within the domain, where, typically, development work was undertaken on a private fork of the project software, and then reintegrated with the upstream project.

6.2 Investment in Projects

Some interviewees, particularly core developers, identified strategic dimensions to their activities. They spoke of an additional investment of time and effort when reviewing and integrating contributed source code to encourage further contributions. Others identified software maintenance tasks such as refactoring that would not be done by non-core contributors. In some cases, where the project software is a major component of the employer's revenue stream, core developers also reported a larger part of the work they did was of direct benefit to the project, but of less immediate benefit to their employer. Work of this kind contributes to the quality of the software created by the project as well as making technical contributions easier by reducing technical debt. Similarly, activities, that may be non-technical, such as supporting the governance of the project contribute to the longevity of the project, and represent a longer term perspective of the project. That companies invest in the project rather than just contributing to the technical effort is indicative of the long-term importance of the project to the business.

6.3 Operational Costs

Engagement with a project is often seen as a long-term investment in staff time and expertise, which also consumes company resources that are typically considered a short-term operational cost. Consequently, for both businesses and individual contributors it is desirable that interactions with OSS projects should be effective and efficient. We found interactions where developers proceed cautiously by, for example, trying to explore whether a specific feature might be accepted (Example 6) so as to avoid duplication of effort, or unnecessary work. However, we also found that contributors can be drawn into time-consuming interactions (e.g., Example 4). Example 4 and similar cases can have obvious causes, such as miscommunication or not following instructions, but in some instances the cause is less clear and further research is needed to understand the causes of inefficient interactions between contributors and project and how they might be avoided. If, as Milinkovich argues [43], OSS will be used increasingly, then without understanding the causes of inefficient interaction and how to avoid them, a lot of working time, and thus resources, may be used unnecessarily.

Several interviewees indicated the value of preparation of questions and bug reports using the project documentation, and external sources, as a way of ensuring that interactions within the project were more effective and efficient. One interviewee drew attention to the value to their employer of help-giving within a project as preparation for working with customers. Interviewees also identified the richness of some contributions that was not immediately apparent from observation as a means of trying to achieve additional goals. For example, an outwardly simple help-request was used to try to influence software development plans within the project.

A further aspect of the costs encountered, and identified by interviewees, is that of technical debt, in the sense that maintaining local source code improvements, or bug fixes and reintegrating them at each release, is not an efficient or cost effective way of working. It is, however, as was identified in one case, a necessity where the upstream project is critical to the business model. In the long term a business reduces costs through enhancements to OSS project software being integrated in the project software; so long as they do not generate revenue through the addition of commercially differentiating functionality.

As well as the influence of deployment and the business model on the way that companies contribute to OSS projects, two additional factors appear to motivate the approaches to contribution by companies and those working with them, or on their behalf. The first is the maturity of the software implementation and the domain, and the

second is the location of the knowledge and expertise required to complete a given task. In both cases care is needed to identify cost effective opportunities to make contributions.

6.3.1 Software and Domain Maturity

Projects where the software implementation is perceived as relatively immature and the core functionality under development, such as Leshan (where some parts of the OMA LWM2M specification remain to be implemented), can require greater investment of company resources in the project (in collaboration with competitors or not) to ensure the software meets the company's requirements. Opportunities for a company to work with the community include company developers joining the community, and the company employing core developers within the community. The latter is possible where the implementation aims of the company do not conflict substantially with those of the community. An example might be the development of software to implement an existing standard where the domain is thought to be relatively mature and clearly defined.

Software maturity is not easily evaluated and has many aspects. Consequently, determining the maturity of an OSS project requires recognition of aspects of the project, and whether they might be subject to change. The core functionality of Solr, for example, is perceived as relatively mature. However, machine learning techniques are relevant in the search domain and are being introduced to Solr. In addition, relative increase in the amount of data searched and consequent changes to the hardware platforms on which Solr is deployed in industry also influence the direction of software development. Consequently there can be implementation or reimplementation of features and functionality as software technology develops, and software and configuration changes in response to developments in external technology. Accordingly, though aspects of an OSS project may be considered mature and the direction of software development largely self-regulates, there are aspects of the project that may evolve as a consequence of external changes. Although the involvement of some companies using the software is mainly limited to contributing bug reports, vigilance is also required to ensure the project software continues to meet existing requirements.

6.3.2 Knowledge and Expertise

Interviewees also identified the location of knowledge and expertise as a significant factor when deciding to make a contribution to an OSS project, because it can indicate who may be best placed to contribute as well as the type of contribution that can be made. Three broad categories of knowledge emerged from analysis of OSS projects and the interview data: knowledge of the *application domain*, knowledge the *software implementation*, and knowledge of software *deployment and use*.

First, the *application domain* knowledge in the software is an asset that companies exploit to add value when delivering a service or a product. In some sectors, for example security, or during product innovation there can also be a significant level of domain knowledge within the company using the software. In the case of Bouncy Castle, for example, the application domain expertise and awareness of the community helps to identify new areas for development, as well as to report bugs clearly to the developers. Also, in projects associated with open innovation, like Leshan, expertise within companies that arises from product development supports the implementation of features, or missing functionality, in the upstream project.

Second, detailed knowledge of the *software implementation* is usually limited to the core developers within a project. Generally, it is possible to delegate responsibility for implementing bug fixes and feature requests to the upstream project. However, some deployment contexts can require timely implementation of fixes and features. Where the project software is deployed as part of a product, or delivers a critical service, it can be necessary for contributors to implement bug fixes to maintain revenue generation. A key challenge in such circumstances, therefore, is to acquire sufficient knowledge of the software implementation to be able to implement meaningful changes. Furthermore, there is a trade-off between implementing a bug fix that resolves the problem until it is fixed in the next release, and a solution that meets the requirements and development plans of the core developers sufficiently that the changes will be incorporated into the upstream project. Striking the right balance is a key challenge for a company, and, as identified through the analysis of interview data, there are many ways to acquire software implementation knowledge and expertise. Employees can acquire knowledge sufficient to implement fixes, though there may be limitations to the level of expertise that can be acquired alongside their day-to-day work. However, some core developers are willing to invest time to nurture contributors so that they are able to make more effective technical contributions. There is also the issue highlighted by some interviewees where the level of knowledge and expertise required to develop the software, in particular cases, can be such that it may be an unrealistic proposition for a contributor (individual or company) to acquire that expertise, especially where business resources are constrained. A further option, identified by some interviewees, is to hire expertise already within the upstream project in the form of core developers — either as consultants, or to invest in the project, if appropriate, and employ core developers as staff. Alternatively, companies may develop working relationships with businesses that employ core developers.

Third, *deployment and use* knowledge and expertise lies both with the user community and the core developers. Contributing knowledge to the project community makes the project more attractive to new users and contributors, and can contribute to the development of professional expertise of the help-giver. Documenting knowledge of deployment also helps improve the quality of the software by recording use cases and requirements that the core developers may not have considered.

To summarise: the challenge for any business contributing to an OSS project is to understand the wide range of factors that might motivate the contribution, as well as the factors that constrain avenues of action, and the need to identify appropriate means of interacting with the project that are an effective use of resources.

6.4 Discussion

In this article we have reported on how and why companies and practitioners acting on their behalf contribute to eight community OSS projects. Companies contribute to OSS projects for business reasons, and decisions about the nature of the contribution are influenced primarily by the need for the business to generate long-term value, i.e., for the business to benefit from the contribution. We found a wide variety of ways businesses contribute to projects and provided some explanations of the choices made from analysis of interviews with practitioners. Observations of practitioners' work practices and the interviews provide rich descriptions of the factors considered, including where expertise and knowledge lie, and how effectively the business might be able to contribute in order to achieve its goals. The goals may be short-term, such as fixing a bug, or more long-term, such as supporting a particular development effort, or perhaps even influencing the direction of software development.

A major influence on the type and level of engagement with an OSS project is the way in which the business deploys the software created by the project. In some deployment contexts the software requires little or no modification to create revenue for the company; the revenue comes from selling the expertise required to deploy the software. At the other end of the spectrum the software is deployed in a context where ongoing development of features is required to support the generation of revenue. In such usage contexts the software may be deployed to provide services to customers who have evolving requirements, or it may be that the software is in a rapidly evolving domain. However, it is important to remember that businesses participate in OSS projects for multiple reasons, and while there may be common factors for many companies contributing to a given project, they do not lead to the same form of contributions to the project. Contributions, and consequently the pace and direction of development, are the combination of the needs and capabilities of many companies.

The capability of a company to make a technical contribution to an OSS project is contingent on having the necessary technical and implementation knowledge and expertise, as well as other resources, including staff time to make the contribution. Companies need to be aware of their strengths and weaknesses when making decisions and to adopt a cautious approach when proposing additional features, for example. Furthermore, that if, for any reason, the company lacks the capacity to make a technical contribution, then there is the opportunity to outsource the work to others. An orthogonal factor that must be considered is the timeliness of any implementation; can the company afford to wait for the project to implement the change? The company may then need to acquire the necessary expertise.

The contributions made by interviewees were motivated, mostly, by the need to complete software development tasks for their employer. The majority of activity was intended to meet short-term goals, and driven by the need to deliver a product or service. Some activity was more strategic and intended by the commissioning business to support the community OSS project. We also identified instances where individuals spent part of their paid work making contributions to develop skills beneficial to the business. While many interviewees were motivated for career reasons to work in jobs where they could contribute to OSS projects, contributions were made in the context of paid employment, and for the benefit of the business.

Rather than simply contributing in ways that support the technical effort where there are direct benefits to the company's revenue, some companies contribute to the longer term future of the project. There appears to be a point at which the project becomes of sufficient importance for the business to support aspects of software development, through employing core developers, that contribute to the long-term future of the project in ways that do not have an apparent financial return for the company. Exactly what factors lead to the decision to invest in a community OSS project is a subject for future research.

We have identified a range of non-code contributions including donations, sponsorship, and participation in governance processes, and some motivations to make such contributions. Documentation, for example, is an activity that developers contribute to and that some individuals specialise in. Individuals contributing primarily to the documentation process did not respond to invitations for interview. The broad topic of non-technical contributions and particularly those who make only non-code contributions to community OSS projects is therefore an area for future research.

The study presented in this article presents systematic analyses of collected data from public sources and contributors to eight widely used community OSS projects implementing software in a variety of domains. We acknowledge the inherent characteristics of utilising a purposeful sampling for transferability of findings from the study. While we cannot reflect every possible form of business pressure on, and work practice used by, contributors to OSS projects in this study, we have provided results which draw from a systematic analysis of rich insights and experiences of activities related to the investigated community OSS projects, including drawing on the long experience of the authors. Consequently, we conjecture that the findings may be particularly representative for transferability of results related to company involvement with other community OSS projects.

7 Conclusions

We have reported on the interactions of companies with eight community open source software projects governed independently and by foundations. The work practices used by companies to contribute to OSS projects are identified and characterised, and the motivations for the use of particular contribution strategies and work practices explored through analysis of data gathered during interviews with contributors.

Our investigation provides a picture of the inherent complexity for businesses working with OSS and illuminates the manner in which companies and practitioners contribute to OSS projects, despite the outward similarity of the project structures, available communication channels, and apparent business models and priorities of participants. We found key factors that help determine how a company interacts with a community OSS project include the maturity of the software created by the project, the business context within which the company deploys the software, and the

balance of areas of knowledge and expertise between the company and the project. In addition, companies have a strong interest in the longevity and sustainability of projects they use and contribute to, that also motivates their activities and both technical and non-technical contributions, and can motivate a much more strategic investment of resources in the project.

This study makes the following contributions to the existing body of knowledge:

- Identification of work practices used by companies and practitioners to contribute to eight widely used community OSS projects.
- Documentation of factors for both the community OSS project and contributors that influence company and practitioner decision-making.
- Documentation of insights from industrial praxis into the opportunities and constraints of the relationships between projects, companies and practitioners.
- Identification of opportunities for companies and practitioners to improve their strategies for working with community OSS projects.

In summary, this study contributes novel findings about the nature of, and the decision-making behind, strategic and everyday contributions by companies and practitioners to community OSS projects. The rich descriptions and analysis of the interviewed practitioners' insights and experiences provide an understanding of the nature of the complex interplay between influences from technical and business considerations that inform decisions made by businesses and individual practitioners about the work practices used to make contributions to independently governed OSS projects. The findings draw from investigations of company involvement with eight community OSS projects which to large extent may be transferable to other similar contexts involving other OSS projects. However, findings from the study should not be perceived as supporting a context-independent prescribed method for contributing successfully to all other community OSS projects. Hence, findings from the study indicate the need for awareness and understanding by businesses and practitioners of the many characteristics of both their own situation and goals, and those of the OSS project, in order to be able to contribute effectively.

Acknowledgments

The authors are very grateful for the input of contributors to the OSS projects investigated who were interviewed for this research. 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.

References

[1] B. Fitzgerald, "The transformation of open source software," *Manage. Inf. Syst. Quarterly*, vol. 30, no. 3, pp. 587–598, Sep. 2006.

[2] K. Fogel "Dissecting the myth that open source software is not commercial," *IEEE Software Blog*, Apr. 2016, Accessed on: Aug. 29, 2018. [Online]. Available: http://blog.ieeesoftware.org/2016/04/dissecting-myth-that-open-source.html

[3] K. Fogel, *Producing Open Source Software: How to Run a Successful Free Software Project*, 2nd ed. O'Reilly Media, Jan. 2017, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.producingoss.com/

[4] F. van der Linden, B. Lundell, and P. Martiin, "Commodification of industrial software: A case for open source," *IEEE Softw.*, vol. 26, no. 4, pp. 77–83, Jul.-Aug. 2009.

[5] K.-J. Stol, P. Avgeriou, M. A. Babar, Y. Lucas, and B. Fitzgerald, "Key factors for adopting inner source," *ACM Trans. Softw. Eng. Methodology*, vol. 23, no. 2, pp. 18:1–18:35, Apr. 2014.

[6] D. Riehle, M. Capraro, D. Kips, and L. Horn, "Inner source in platform-based product engineering," *IEEE Trans. Softw. Eng.*, vol. 42, no. 12, pp. 1162–1177, Dec. 2016.

[7] J. Linåker, H. Munir, K. Wnuk, and C.-E. Mols, "Motivating the contributions: An open innovation perspective on what to share as open source software," *J. Syst. Softw.*, vol. 135, no. Supplement C, pp. 17–36, 2018.

[8] H. Munir, J. Linåker, K. Wnuk, P. Runeson, and B. Regnell, "Open innovation using open source tools: A case study at Sony Mobile," *Empirical Softw. Eng.*, vol. 23, no. 1, pp. 186–223, Feb. 2018.

[9] B. Lundell, B. Lings, and E. Lindqvist, "Open source in Swedish companies: Where are we?" *Inf. Syst. J.*, vol. 20, no. 6, pp. 519–535, 2010.

[10] A. Bonaccorsi and C. Rossi, "Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business," *Knowl. Technol. Policy*, vol. 18, no. 4, pp. 40–64, Dec. 2006.

[11] A. Bonaccorsi, D. Lorenzi, M. Merito, and C. Rossi, "Business firms' engagement in community projects. Empirical evidence and further developments of the research," in *Proc. 1st Int. Workshop Emerging Trends FLOSS Res. Develop.*, 2007, pp. 13–17.

[12] J. Lerner and J. Tirole, "Some simple economics of open source," *J. Ind. Econ.*, vol. 50, no. 2, pp. 197–234, 2002.

[13] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *Proc. Int. Workshop Principles Softw. Evolution*, 2002, pp. 76–85.

[14] K. Crowston and B. Scozzi, "Bug fixing practices within free/libre open source software development teams," *J. Database Manage.*, vol. 19, no. 2, pp. 1–30, 2008.

[15] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open source software development: What we know and what we do not know," *ACM Comput. Surveys*, vol. 44, no. 2, pp. 7:1–7:35, Mar. 2012.

[16] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Comput. Supported Cooperative Work*, vol. 14, no. 4, pp. 323–368, Aug. 2005.

[17] K. Wei, K. Crowston, U. Y. Eseryel, and R. Heckman, "Roles and politeness behavior in community-based free/libre open source software development," *Inf. Manage.*, vol. 54, no. 5, pp. 573–582, 2017.

[18] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *Proc. 12th Int. Conf. Global Softw. Eng.*, 2017, pp. 66–75.

[19] G. Pinto, I. Steinmacher, and M. A. Gerosam, "More common than you think: An in-depth study of casual contributors," in *Proc. 23rd Int. Conf. Softw. Anal. Evolution Reengineering*, Mar. 2016, pp. 112–123.

[20] D. Riehle, P. Riemer, C. Kolassa, and M. Schmidt, "Paid vs. volunteer work in open source," in *Proc. 47th Hawaii Int. Conf. Syst. Sci.*, Jan. 2014, pp. 3286–3295.

[21] D. Riehle, "How open source is changing the software developer's career," *IEEE Comput.*, vol. 48, no. 5, pp. 51–57, May 2015.

[22] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: The integrator's perspective," in *Proc. 37th IEEE/ACM Int. Conf. Softw. Eng.*, 2015, pp. 358–368.

[23] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 285–296.

[24] M.-A. Storey, A. Zagalsky, F. Figuera Filho, L. Singer, and D. M. German, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Trans. Softw. Eng.*, vol. 43, no. 2, pp. 185–204, Feb. 2017.

[25] A. Bosu, J. C. Carver, C. Bird, J. D. Orbeck, and C. Chockley, "Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at Microsoft," *IEEE Trans. Softw. Eng.*, vol. 43, no. 1, pp. 56–75, Jan. 2017.

[26] D. M. German, G. Robles, G. Poo-Caamaño, X. Yang, H. Iida, and K. Inoue, ""Was my contribution fairly reviewed?" a framework to study the perception of fairness in modern code reviews," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 523–534.
[27] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, and M. Burnett, "Open source barriers to entry, revisited: A sociotechnical perspective," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 1004–1015.
[28] Ø. Hauge and S. Ziemer, "Providing commercial open source software: Lessons learned," in *Proc. 5th IFIP WG 2.13 Int. Conf. Open Source Syst.*, 2009, pp. 70–82.
[29] L. Dahlander and M. W. Wallin, "A man on the inside: Unlocking communities as complementary assets," *Res. Policy*, vol. 35, no. 8, pp. 1243–1259, 2006.
[30] L. Dahlander and M. Magnusson, "How do firms make use of open source communities?" *Long Range Planning*, vol. 41, no. 6, pp. 629–649, 2008.
[31] M. Schaarschmidt, G. Walsh, and H. F. von Kortzfleisch, "How do firms influence open source software communities? a framework and empirical analysis of different governance modes," *Inf. Organ.*, vol. 25, no. 2, pp. 99–114, Apr. 2015.
[32] M. Levy and M. Germonprez, "Is it egalitarianism or enterprise strategy? Exploring a new method of innovation in open source," in *Proc. Amer. Conf. Inf. Syst.*, 2015, pp. 4778–4789.
[33] M. Zhou, A. Mockus, X. Ma, L. Zhang, and H. Mei, "Inflow and retention in OSS communities with commercial involvement: A case study of three hybrid projects," *ACM Trans. Softw. Eng. Methodology*, vol. 25, no. 2, pp. 13:1–13:29, Apr. 2016.
[34] D. Riehle, "Controlling and steering open source projects," *IEEE Comput.*, vol. 44, no. 7, pp. 93–96, Jul. 2011.
[35] J. Gamalielsson, B. Lundell, and A. Mattsson, "Open source software for model driven development: A case study," in *Proc. 7th IFIP Int. Conf. Open Source Syst.*, 2011, pp. 348–367.
[36] M. Germonprez, J. P. Allen, B. Warner, J. Hill, and G. McClements, "Open source communities of competitors," *Interaction*, vol. 20, no. 6, pp. 54–59, Nov. 2013.
[37] M. L. Markus, "The governance of free/open source software projects: Monolithic, multidimensional, or configurational?" *J. Manage. Governance*, vol. 11, no. 2, pp. 151–163, May 2007.
[38] M. Shaikh and O. Henfridsson, "Governing open source software through coordination processes," *Inf. Org.*, vol. 27, no. 2, pp. 116–135, 2017.
[39] J. Gamalielsson and B. Lundell, "On licensing and other conditions for contributing to widely used open source projects: An exploratory analysis," in *Proc. 13th Int. Symp. Open Collaboration*, 2017, pp. 9:1–9:14.
[40] G. Poo-Caamaño and D. German, "The right to a contribution: An exploratory survey on how organizations address it," in *Proc. 11th IFIP Int. Conf. Open Source Syst.*, 2015, pp. 157–167.
[41] J. Gamalielsson and B. Lundell, "Sustainability of open source software communities beyond a fork: How and why has the LibreOffice project evolved?" *J. Syst. Softw.*, vol. 89, pp. 128–145, 2014.
[42] S. Butler, J. Gamalielsson, B. Lundell, P. Jonsson, J. Sjöberg, A. Mattsson, N. Rickö, T. Gustavsson, J. Feist, S. Landemoo, and E. Lönroth, "An investigation of work practices used by companies making contributions to established OSS projects," in *Proc. 40th Int. Conf. Softw. Eng.: Softw. Eng. Practice Track*, 2018, pp. 201–210.
[43] M. Milinkovich, "Open collaboration, the Eclipse way," The Eclipse Foundation, 2017, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.slideshare.net/mmilinkov/icse-2017-keynote-open-collaboration-at-eclipse
[44] B. Lundell, J. Gamalielsson, S. Tengblad, B. H. Yousefi, T. Fischer, G. Johansson, B. Rodung, A. Mattsson, J. Oppmark, T. Gustavsson, J. Feist, S. Landemoo, and E. Lönroth, "Addressing lock-in, interoperability, and long-term maintenance challenges through open source: How can companies strategically use open source?" in *Proc. 13th IFIP Int. Conf. Open Source Syst.*, 2017, pp. 80–88.
[45] B. Fitzgerald, K.-J. Stol, S. Minör, and H. Cosmo, *Scaling a Software Business: The Digitalization Journey*. Cham, Switzerland: Springer International Publishing, 2017.
[46] D. Riehle, "The economic case for open source foundations," *IEEE Comput.*, vol. 43, no. 1, pp. 86–90, Jan. 2010.
[47] M. Andersen-Gott, G. Ghinea, and B. Bygstad, "Why do commercial companies contribute to open source software?" *Int. J. Inf. Manage.*, vol. 32, no. 2, pp. 106–117, 2012.
[48] K. R. Lakhani and E. von Hippel, "How open source software works: "free" user-to-user assistance," *Res. Policy*, vol. 32, no. 6, pp. 923–943, 2003.
[49] P. van Wesel, B. Lin, G. Robles, and A. Serebrenik, "Reviewing career paths of the OpenStack developers," in *Proc. IEEE Int. Conf. Softw. Maintenance Evolution*, Sep. 2017, pp. 544–548.
[50] P. J. Ågerfalk and B. Fitzgerald, "Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy," *Manage. Inf. Syst. Quart.*, vol. 32, no. 2, pp. 385–409, 2008.
[51] C. Alves, J. Oliveira, and S. Jansen, "Software ecosystems governance - a systematic literature review and research agenda," in *Proc. 19th Int. Conf. Enterprise Inf. Syst. - Vol. 3*, 2017, pp. 215–226.
[52] M. Germonprez, J. E. Kendall, K. E. Kendall, and B. Young, "Collectivism, creativity, competition, and control in open source software development: Reflections on the emergent governance of the SPDX® working group," *Int. J. Inf. Syst. Manage.*, vol. 1, no. 1/2, pp. 125–145, Jun. 2014.
[53] 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 Serv. Appl.*, vol. 9, no. 1, Aug. 2018, Art. no. 17.
[54] B. Lundell and F. van der Linden, "Open source software as open innovation: Experiences from the medical domain," in *Managing Open Innovation Technologies*, J. S. Z. Eriksson Lundström, M. Wiberg, S. Hrastinski, M. Edenius, and P. J. Ågerfalk, Eds. Berlin, Germany: Springer, 2013, pp. 3–16.
[55] L. Morgan and P. Finnegan, "Beyond free software: An exploration of the business value of strategic open source," *J. Strategic Inf. Syst.*, vol. 23, no. 3, pp. 226–238, 2014.
[56] Y. Zhang, M. Zhou, W. Zhang, H. Zhao, and Z. Jin, "How commercial organizations participate in OpenStack open source projects," *J. Softw.*, vol. 28, no. 6, pp. 1343–1356, 2017.
[57] Y. Zhang, X. Tan, M. Zhou, and Z. Jin, "Poster: Companies' domination in FLOSS development an empirical study of OpenStack," in *Proc. 40th Int. Conf. Softw. Eng.: Companion Proc.*, 2018, pp. 440–441.
[58] M. Zhou and A. Mockus, "Who will stay in the FLOSS community? modeling participant's initial behavior," *IEEE Trans. Softw. Eng.*, vol. 41, no. 1, pp. 82–99, Jan. 2015.
[59] J. Gerring, *Case Study Research: Principles and Practices*, 2nd ed. Cambridge, U.K.: Cambrige Univ. Press, 2017.
[60] M. Q. Patton, *Qualitative Research and Evaluation Methods*, 4th ed. Thousand Oaks, CA, USA: Sage Publications Inc., 2015.
[61] R. Pearce-Moses, *A Glossary of Archival and Records Terminology*. Chicago, Il, USA: The Society of American Archivists, 2005, Accessed on: Oct. 30, 2018. [Online]. Available: http://files.archivists.org/pubs/free/SAA-Glossary-2005.pdf
[62] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Trans. Softw. Eng. Methodology*, vol. 11, no. 3, pp. 309–346, Jul. 2002.
[63] V. Singh, "Newcomer integration and learning in technical support communities for open source software," in *Proc. 17th ACM Int. Conf. Supporting Group Work*, 2012, pp. 65–74.
[64] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. London, U.K.: Weidenfeld and Nicolson, 1967.
[65] B. Lings and B. Lundell, "On the adaptation of grounded theory procedures: Insights from the evolution of the 2G method," *Inf. Technol. People*, vol. 18, no. 3, pp. 196–211, 2005.
[66] P. Ågerfalk, A. Deverell, B. Fitzgerald, and L. Morgan, "Assessing the role of OSS in the European secondary software sector: A voice from industry," in *Proc. 1st Int. Conf. Open Source Softw.*, 2005, pp. 82–87.
[67] Apache Software Foundation, Apache Software Foundation, 2019, Accessed on: Apr. 2, 2019. [Online]. Available: http://www.apache.org/theapacheway/
[68] Apache Software Foundation, Bylaws of the Apache Software Foundation, Apache Software Foundation, 2002, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.apache.org/foundation/bylaws.html
[69] Apache Software Foundation, "Code of conduct", The Apache Software Foundation, 2018, Accessed on: Aug. 2018. [Online]. Available: http://www.apache.org/foundation/policies/conduct.html
[70] Eclipse Foundation, "Community code of conduct," Eclipse Foundation, Inc., 2015, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.eclipse.org/org/documents/Community_Code_of_Conduct.php

[71] Eclipse Foundation, Bylaws of Eclipse Foundation, Inc. Eclipse Foundation, Aug. 2011, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.eclipse.org/org/documents/Eclipse%20BYLAWS%202011_08_15%20Final.pdf
[72] Canonical Ltd, "Ubuntu code of conduct v2.0," Canonical Ltd, 2017 Accessed: 2018–08-29. [Online]. Available: https://www.ubuntu.com/about/about-ubuntu/conduct
[73] OpenStack Foundation, "The OpenStack Foundation community code of conduct," OpenStack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/legal/community-code-of-conduct/
[74] OpenStack Foundation, "Bylaws of the OpenStack Foundation," OpenStack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/legal/bylaws-of-the-openstack-foundation/
[75] OpenStack Foundation, "The OpenStack Foundation code of conduct," OpenStack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/legal/code-of-conduct/
[76] Apache CloudStack, "CloudStack's history," Apache Software Foundation, 2017, Accessed on: Aug. 29, 2018. [Online]. Available: https://cloudstack.apache.org/history.html
[77] Contiki OS, "Contiki: The open source OS for the internet of things," Thingsquare, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.contiki-os.org/
[78] Eclipse Leshan, "Leshan," Eclipse Leshan, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://github.com/eclipse/leshan/blob/master/README.md
[79] Eclipse IoT Working Group, "Open source for IoT," Eclipse IoT Working Group, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://iot.eclipse.org/
[80] R. Faudou, D. Sciamma, P. Hofer, S. Gérard, E. Juliot, L. Bigeardel, K. Hussey, F. Allilaire, N. Boldt, P. Gaufillet, and J. Lescot, "MDT/Papyrus-proposal," Eclipse Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://wiki.eclipse.org/MDT/Papyrus-Proposal
[81] Papyrus Industry Consortium, "Papyrus IC," Papyrus Industry Consortium, 2017, Accessed on: Aug. 29, 2018. [Online]. Available: https://wiki.polarsys.org/Papyrus_IC
[82] PolarSys, "Open source solutions for systems engineering and embedded systems," PolarSys, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.polarsys.org/
[83] OpenStack Foundation, "Open source software for creating private and public clouds," OpenStack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/
[84] Open Stack Foundation, The Open Stack Foundation. Open Stack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/foundation/
[85] United States Federal Government, "Title 26 - internal revenue code: Section 501," Cornell Law School, Accessed: 2018–10-31. [Online]. Available: https://www.law.cornell.edu/uscode/text/26/501
[86] Australian Charities and Not For Profits Commission, "The legion of the Bouncy Castle: Registration details," Australian Charities and Not For Profits Commission, 2017, Accessed on: Apr. 2, 2019. [Online]. Available: https://www.acnc.gov.au/charity/390a94c0f97cc6e8942ac4ba0628169f
[87] MariaDB Foundation, "Governance," The MariaDB Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://mariadb.org/about/governance/
[88] MariaDB Foundation, "About MariaDB," MariaDB Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://mariadb.org/about/
[89] Apache Software Foundation, "Contributor licence agreements," The Apache Software Foundation, 2004, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.apache.org/licenses/#clas
[90] ASF, "Individual contributor license agreement ("agreement") v2.0," Apache Software Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.apache.org/licenses/icla.pdf
[91] ASF, "Software grant and corporate contributor license agreement ("agreement")," Apache Software Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.apache.org/licenses/cla-corporate.pdf
[92] Eclipse Foundation, "Eclipse contributor agreement," Eclipse Foundation, Aug. 2016, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.eclipse.org/legal/ECA.php
[93] MariaDB Foundation, "MariaDB contributor agreement," The MariaDB Foundation, 2017, Accessed on: Aug. 29, 2018. [Online]. Available: https://mariadb.org/get-involved/getting-started-for-developers/mca/
[94] The Legion of the Bouncy Castle Inc, "Donate to support the Bouncy Castle APIs," The Legion of the Bouncy Castle Inc, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.bouncycastle.org/donate/index.cgi
[95] ASF, "The Apache Software Foundation sponsorship program," Apache Software Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.apache.org/foundation/sponsorship.html
[96] Eclipse Foundation, "Eclipse membership," Eclipse Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.eclipse.org/membership/
[97] Eclipse Foundation, "Eclipse corporate sponsors," Eclipse Foundation, 2017, Accessed on: Aug. 29, 2018. [Online]. Available: http://www.eclipse.org/corporate_sponsors/
[98] MariaDB Foundation, "Sponsor," MariaDB Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://mariadb.org/donate/
[99] OpenStack Foundation, "Join the OpenStack Foundation," OpenStack Foundation, 2018, Accessed on: Aug. 29, 2018. [Online]. Available: https://www.openstack.org/join/

Simon Butler received the BSc degree in information technology and computing from the Open University, in 2007 and the PhD degree in computing from the Open University, in 2016. He is a researcher with the Software Systems Research Group, University of Skövde in Sweden. His research interests include software engineering, open source software, program comprehension, software development tools and practices, and software maintenance. He is a member of the IEEE, the IEEE Computer Society, the ACM, and the BCS.

Jonas Gamalielsson received the PhD degree from Heriot Watt University, in 2009. He is a senior lecturer with 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.

Björn Lundell received the PhD degree from the University of Exeter, in 2001. He is a professor with the University of Skövde where he leads the Software Systems Research Group, and has been a staff member and researcher since 1984. His research contributes to theory and practice in the software systems domain and centres on different aspects of openness (in particular open source and open standards) related to development, use, and procurement of software systems. His research addresses fundamental socio-technical challenges concerning software systems, and focuses on different aspects of lock-in, interoperability, and longevity of systems. His research is reported in more than 100 papers in a variety of international journals and conferences. He has been active in a number of international and national research projects which have lead to significant scientific and societal impact, and has been an expert advisor and contributed to national guidelines and policies in different countries and the EU. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

Christoffer Brax received the MSc degree from the University of Skövde, in 2000, and the PhD degree 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. He has 18 years experience as a systems engineer.

Johan Sjöberg received the BSc degree in computer science from Chalmers University of Technology, in 2002 and the MSc degree in distributed systems from the Chalmers University of Technology, in 2009. He is CTO of Findwise AB where he develops enterprise search solutions to support businesses manage and analyse their information.

Anders Mattsson received the MSc degree from Chalmers University of Technology, Sweden, in 1989 and the PhD degree 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. He is particularly interested in strengthening software engineering practices in organizations. Special interests includes software architecture and model-driven development in the context of embedded real-time systems. He is a member of the IEEE and the IEEE Computer Society.

Tomas Gustavsson received the MSc 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. He 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 MSc degree in computer science from the Institute of Technology, Linköping University, in 1988. He is senior executive and co-founder of RedBridge AB, a computer consultancy business in Stockholm.

Erik Lönroth received the MSc degree in computer science and is the technical responsible for the high performance computing area at Scania IT AB. He has been leading the technical development of four generations of super computing initiatives with Scania and their supporting subsystems. He frequently lectures on development of super computer environments for industry, open source software governance and HPC related topics.