Rashid, M., Clarke, P. M., & O'Connor, R. V. (2017). Exploring Knowledge Loss in Open Source Software (OSS) Projects. In A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, & A. Dorling (Eds.), Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4–5, 2017, Proceedings (pp. 481–495). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-67383-7_35

Exploring Knowledge Loss in Open Source Software (OSS) Projects

Mehvish Rashid [1,2], Paul M. Clarke[1, 2], Rory V. O'Connor[1, 2]
[1]Dublin City University, Dublin, Ireland
[2]Lero, the Irish Software Research Centre, Ireland
Mehvish.Rashid2@mail.dcu.ie, Paul.M.Clarke@dcu.ie, Rory.OConnor@dcu.ie

Abstract. Open Source Software is a term used to identify software developed and released under an "open source" license, meaning that under certain conditions; it is openly available for use, inspection, modification, and for redistribution free of cost (or with cost based on the license agreement). Incorporation of OSS while developing software can reduce time and cost of development. The nature of the work force (volunteers and paid) in OSS projects is transient and results in high turnover leading to knowledge loss. In this work, we explore the phenomenon of knowledge loss in OSS projects. Maintenance of OSS projects requires knowledge, typically shared asynchronously using technology-mediated channels. Knowledge sought in this manner is reactive in the sense that a developer will consult these channels looking for possible solutions or supporting information. We follow the backward snowballing to study the relevant literature on knowledge loss in OSS. Our work suggests that proactive knowledge exchange mechanisms may bring some benefits to OSS projects. Further integration of knowledge management practices with the established OSS practices can minimise knowledge loss.

Keywords: Software Development Process; Open Source Software; Knowledge Loss; Knowledge Retention Process; Open Source Software Contributor.

1 Introduction

Open Source Software (OSS) is a term used to identify software developed and released under an "open source" license that complies with Open Source Definition (OSD). The OSD uses either a short definition based on four criteria as in the Free Software Foundation (FSF) or a longer version based on ten criteria as in the Open Source Initiative (OSI). The difference between these two definitions is only of language while underlying meaning and outcome is the same. "The freedom to use, change, sell, or give away the software, the availability of source code, and the protection of authors' intellectual property rights are the central tenets of the OSD"

[1]. Users with technical inclination can use, freely access the code, inspect, modify and redistribute the software [2]. However, the freedom to use source code from an OSS project and its distribution varies based on which category of OSS license agreement is applied. There are three main categories of OSS licenses based on their degree of restrictiveness: Strong-copyleft, weak-copyleft and non-copyleft [3]. A strong-copyleft or restrictive license requires that any derivatives of the original software are also licensed in a similar manner. Weak-copyleft licenses allow the derivatives of the software to be released under different license. Non-copyleft licenses allow the software including derivatives to be redistributed under a different license than the original one. While free software mostly identifies with GNU Public License (GPL), OSS license agreements may vary based on the incorporation of the software that can be either propriety or free. Another term to represent free software is Free Open Source Software (FOSS). The term "free" in FOSS was not considered by some to adequately express the notion of freedom and consequently, in 2001, the European Commission (EC) introduced the term Free/Libre Open Source Software (FLOSS), to avoid taking sides in the debate and to stay neutral on the distinction between free software and open-source software. OSS projects are of varying sizes and at times involve commercial firms who heavily depend on OSS system [4]. A survey conducted in 2015 reported that almost 78% of companies run operations on open source software and 66% of companies have incorporated open source software to create products for customers [5].

In OSS projects, maintenance and development are not considered as two separate phases of software development lifecycle [6]. Software maintenance is the field which is concerned with the evolution of a software system after its initial release [6]. In Closed Source Software (CSS) or traditional software development, the maintenance phase starts after the software is complete, authorised and running [7]. Whereas in OSS projects, the source code may be released before it is complete or in workable form; and the maintenance activities in OSS start when system is still in the initial stage of development [6]. The OSS system, already in phase-out stage, experience a rebirth when other contributors start contributing with new enhancements [7]. Maintenance in an OSS system is the source of continuous evolution and requires knowledge in various forms. In order to solve a problem, a software developer has to understand existing source codes, design a solution, program the solution, and test it. The nature of the OSS work force (volunteers and paid) is transient and results in high turnover on projects. This turnover leads to knowledge loss on OSS projects [8]. As a potential solution to the knowledge loss situation in OSS projects we introduce the concept of Knowledge Management (KM).

1.1 Knowledge Management

It is important to clarify the distinction between the terms *data*, *information*, and *knowledge*. *Data* represents observations and facts without any contextual meaning, whereas *information* is the result of associating data with a meaningful context [9]. In order to convert data into information, it must be contextualized, categorised, calculated and condensed [9]. *Knowledge* is driven from information [9], it is the product of an individual's experience and accumulates as a result of communication or inference [10]. In a general sense, knowledge may be categorised as either *explicit*

or *tacit* (or implicit) [11]. Tacit knowledge comprises of skills learned due to the personal capabilities of contributors and if not documented, remains confined to an individual, whereas explicit knowledge is available in documented form [11]. At an organisational level, knowledge is created as a result of the interaction between the tacit and explicit knowledge [11]. Accumulated tacit knowledge is lost when contributors leave projects. Knowledge loss is a problem constantly faced due to employees leaving an organisation [12-14] and it is reported in OSS projects [15-17], where the majority of contributors are typically volunteers. The duration of volunteer participation in OSS projects is considered to be unpredictable [18], with the phenomenon of volunteers joining and leaving at their own discretion being more common in OSS projects when compared with employee-based arrangements that are typical in CSS [18].

In order to reduce the impact of knowledge loss on the organisation's productivity and on product's quality, organisations invest in KM processes. KM is defined as the approach adopted by an organisation to engage workers in relevant activities of creating, managing, sharing and reusing knowledge [19].

The purpose of this work is to explore the problem of knowledge loss in OSS due to the transient profile of contributors and to examine the affect this may have on productivity and quality of the project. In section 2, we will explore the literature related to OSS knowledge loss and further inspect KM activities in OSS projects. In section 3, we interpret the findings from section. We conclude this work in section 4 by proposing directions for future research.

2 Related Literature

In this section, we explore the existing literature relevant to the problem of knowledge loss. In order to identify the literature, the initial step was to find the key set of papers related to the topic. Different search strings were used on Google Scholar such as "knowledge loss in open source software", "knowledge loss in free libre open source software", and "knowledge loss in free open source software".

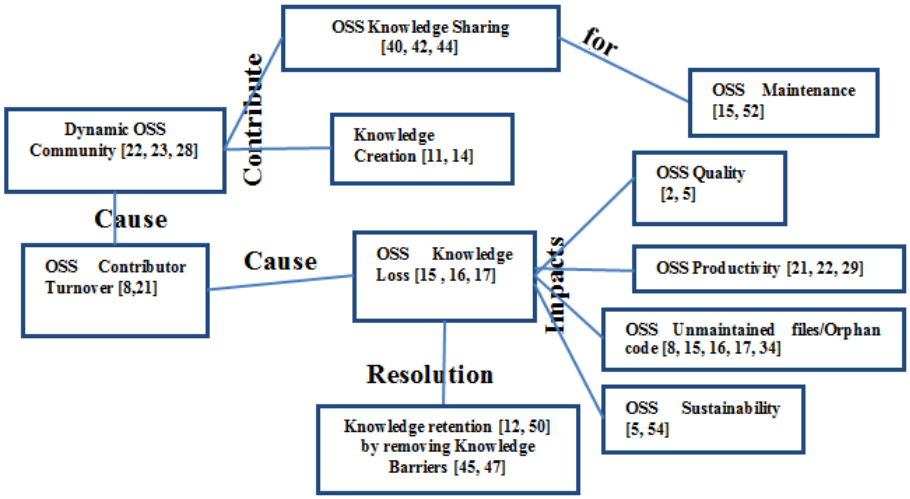

Figure 1: Mind map of related literature on OSS knowledge loss

Key papers of interest were identified using this approach and these formed the initial review corpus. While evaluating the core area of interest, backward snowballing was employed on the initial set of papers. Backward snowballing refers to the process of extending the literature review on the basis of following the trail offered up by the initial set of paper identified [20]. Accordingly, further works of interest were selected from the reference list of initially selected papers. In combination with the term "open source software", other terms relating to knowledge management were also searched on Google Scholar consisting of "knowledge sharing", "knowledge creation", "knowledge reuse", "knowledge retention", "knowledge acquisition ", and "knowledge capture". The themes consolidated from the collected papers are depicted in Figure 1. Each rectangle represents the theme found in the papers and line represents a connection between two themes.

Section 2.1 explains the problem of knowledge loss in OSS, followed by section 2.2 that describes the contributor profile as observed in OSS projects and its implication for knowledge loss. Finally, section 2.3 elaborates on KM related activities in OSS project.

2.1 Knowledge Loss in Open Source Software Projects

Evolution of OSS projects result in evolving teams of contributors who are constantly joining, leaving, or changing their role in the project. The phenomenon of resources joining and leaving in this fashion is referred as 'turnover' [21]. The contributor turnover leads to knowledge loss in OSS projects. In many large OSS projects, a high turnover has been observed leading to the formation of the new development teams [8]. Knowledge loss impacts the productivity of the OSS projects in two ways: 1) The effort required to acquire knowledge to perform the maintenance tasks; and 2) The loss of effort when code is orphaned and removed from the project.

In order to write quality software code knowledgeable contributors are required. Searching knowledge is argued to be time consuming and costly [22]. The search efforts can vary depending on the source and the level of details. A post or a query on the project mailing list require less efforts while searching through the results of search engine or examining the clues into source code documentation is time consuming [22]. A study on the GNOME[1] project reported that 30 months' time is needed for the contributor to understand the software code and to make a contribution [23]. Developers gradually become productive taking more than a year's time on a project to reach productivity plateau [24]. The time to complete distributed tasks is estimated to be three times longer than for co-located tasks [25].

The time required by a new person to learn the inner workings of the project when experienced contributors leave, causes considerable productivity loss [17]. In-depth understanding of software code and interconnecting file structure is not required to complete simple tasks. On the other hand, contributors may have difficulty performing non-trivial tasks due to 'information blocking', unavailability of the relevant information to complete a task [17]. The productivity of the contributor and overall project suffers due to the information blocking and a lack of understanding of the code base. According to estimates, information blocking consumes 60% of developers efforts [26].

During the preparation of a release, contributors make changes to align their work with the goals of the release [6]. As abandoned code increases on the project, the numbers of reported defects increase as well [27]. The maintenance of abandoned code is difficult because the team lacks knowledge of its creation and structure [15, 25]. The source code that remains unmaintained (unless a legacy system) has an element of uncertainty for the development team since the contributors who wrote it have left the project [17]. Removal of unmaintained code results in loss of existing functionality and may impact users of the system [6].

2.2 Contributors in Open Source Software Projects

Contributors are the knowledge workers in OSS projects. The development style in OSS projects is distinguished from CSS by the term: 'cathedral and bazaar' [28]. 'Cathedral' refers to closed approach of software development with a smaller group of developers having an access to the source code. On the contrary, 'Bazaar' refers to an open approach of software development with a large number of volunteers having an access to the source code to contribute on new requirements, bug fixes, and defect reporting. It is argued that a typical OSS project starts with a cathedral development and then transitions to bazaar development style [28]. In a large community of contributors, "the bazaar", the code is under review by many, which has an effect similar to self-corrective mechanism as in peer-review process [28]. Even though the OSS code is openly accessible, the code review is conducted by limited number of contributors [29], who have earned their status by meritocracy and have proven their skills, experience and expertise while working on the project.

[1] GNOME is a well-known large libre project sponsored by several companies. https://www.gnome.org/foundation/

In OSS, each project is an equivalent of an organization in traditional software industry or CSS. The development in OSS is completed in independent, self-assigned, and parallel streams without much coordination due to geographical dispersion [6]. There are two main roles of contributors in OSS projects, developers, and users. Developers contribute in the open source community in a distributed virtual environment and users in parallel provide their feedback. In OSS, developer and the user can be the same person who may contribute the code and test the software in user's role.

The layered structure called an onion model represents contributors in the OSS community [30]. The teams in OSS community consist of core, co-developer, active users, and passive users. The core is a small group of highly skilled and experienced members, responsible for most of the code development and ensures the design and evolution of the project. Co-developers contribute by reviewing or modifying the code or by bug fixes. Active users contribute bug reports or feature request but do not contribute any code. Passive users are the users of the code and do not make any contibution and their number is difficult to predict. However, in Linux developer organise themselves into two groups core and periphery [31]. The core in Linux project consists of project leader and hundreds of maintainers. Periphery is a large group of developers further divided into two teams: development and bug reporting. Based on the demonstration of skills and abilities on the project, the users transcend in onion model towards becoming a core member. A contributor can simultaneously perform more than one role in the OSS project. For example, a contributor can be a core member responsible for code commits and at the same time tester of the code.

The onion model is used in the literature to assess the difference in the progress of volunteers and commercially involved developers [23]. Volunteers joining the OSS project follow the onion model and contribute based on the meritocracy, while hired developers get integrated into the project faster [23]. The reason for the difference in the progression level of volunteers and hired members on GNOME project is due to variance in knowledge accessible to both kinds of contributors.

OSS project collaborations can be of three types: community-based, non-profit organisation and commercially based. Community-based open source projects take their organisational form from an Internet-based community, and the developers are mostly the volunteers [31]. Volunteers collaborate in OSS projects in their free time and do not directly profit economically in any way from their efforts [18]. The intention of the volunteer to participate in OSS projects is to learn new skills, contribute code and develop a reputation within the OSS community that may in the future result in career opportunities [32]. Another motivation for the volunteers is related to the feeling of satisfaction, competence, and fulfilment from code writing called intrinsic motivation [33]. Managing volunteer contributors can cause certain problems not evident in traditional software development [34]. Apache project is managed by volunteers, who are otherwise full time developers. Debian is another project with 100% volunteers who are responsible for tasks including maintaining software packages, supporting the server infrastructure, developing Debian-specific software.

In non-profit organisation OSS projects, developers are either paid workers or volunteers. The project is mature enough and is funded similar to a formal organisation. There is still some element of community projects maintained in such

projects, for example, Apache Software Foundation [35]. In commercially involved OSS projects, a software company sponsors projects and employs majority of contributors. A commercial company, Netscape, managed Mozilla project in the past. Companies like IBM, HP, SUN (now acquired by Oracle), sponsor OSS projects in which major contributors are paid developers [36].

Such a vast community of OSS project contributors and diverse collaborations raise concerns on acquiring distributed knowledge on software development. Software development knowledge is said to be distributed among developers [37]. In OSS projects, a small subset of contributors called core members make major code contributions (80%) [38]. Knowledge when distributed among a small group of contributors in OSS projects, one person leaving can cause the loss of 80% files in the system [15]. On the contrary, when knowledge distributes across a larger group of contributors, one person leaving causes minimum loss of files, as seen in the case of Linux project [15]. OSS projects require uniform distribution of knowledge with a mechanism that resonates with its dynamic work structure.

2.3 Knowledge Management in Open Source Software

Knowledge Management is one of the social processes and a major area of research in Open Source Software (OSS). OSS development is a knowledge-intensive activity and managing knowledge is a challenging task [4]. In this section we identify the knowledge related activities in OSS projects namely knowledge creation and knowledge sharing. Further, we discuss the knowledge barriers faced in OSS projects and details on knowledge retention process used in organisations.

2.3.1 Knowledge Creation in OSS Projects

Knowledge creation in OSS differs from the CSS [31]. A comparison of knowledge creation in OSS and CSS is given based on the five organising principles: Intellectual property ownership, membership restrictions, authority and incentives, knowledge distribution across organisational and geographical boundaries, and dominant mode of communication [31]. In case of CSS, the knowledge is owned by the organisation with an access given to employees hired. The employees are paid for their work, and the knowledge distribution is within the boundaries of the firm, mostly with face-to-face communication. While in the case of community based model the knowledge and membership is open to public and contribution is from members (mostly volunteers). Distribution of knowledge in community based models extends outside the community, and the dominant mode of communication is technology based (similar to CSS distributed development).

In OSS, the knowledge creation follows community based model and involves interaction of contributors on a larger scale than in CSS. Knowledge creation takes place when individuals are collectively working and interacting on a task and are constantly acquiring relevant information. Knowledge creation is through social interaction among individuals and organisations, and it is dynamic in nature [11].

Nonaka et al. proposed knowledge creation process, they explain conversion of tacit knowledge to explicit knowledge, which is then “crystalized.” Explicit knowledge is retained by the relevant organisation and becomes the basis of new knowledge [11]. The process of knowledge creation is based on four modes of

knowledge conversion: *Socialisation, Externalisation, Internalisation,* and *Combination* are coined as *SECI*. *Socialisation* is the sharing of experience and results in the creation of new tacit knowledge from the existing tacit knowledge. *Externalisation* is the conversion of tacit knowledge to explicit knowledge. Externalisation results into the articulated knowledge. *Combination* is the addition of the new explicit knowledge to the existing explicit knowledge in the knowledge system. *Internalisation* is the conversion of explicit knowledge to tacit knowledge. In internalisation, knowledge is acquired from artifacts in explicit form, and new mental models are created again resulting in tacit knowledge.

Nonaka's knowledge assets are produced as the results of inputs and outputs of the knowledge creation process SECI [11]. Knowledge created from socialisation among project members, results in intangible knowledge based on skills and expertise. Knowledge created from socialisation is made explicit through externalisation and results in conceptual knowledge assets such as product's concepts and design. Knowledge integrated with the existing explicit knowledge through combination results in systemic knowledge assets. Examples of systemic knowledge assets are documentations, specifications, and manuals. The explicit knowledge, when acquired by an individual converts to tacit knowledge by internalisation results in routine knowledge assets. The examples of routine knowledge assets are know-how in daily operations, organisation routines and operations.

The process of knowledge creation as detailed through SECI, can be used to understand knowledge creation in OSS projects. In OSS projects, contributors acquire knowledge from communication channels like Internet Relay Chat (IRC), mailing lists, posting on blogs and online resources. As a result, the new tacit knowledge is created similar to socialisation mode in SECI. The resulting communication is in explicit form but not very well structured. A Conversion to externalisation mode will apply to OSS projects, if the unstructured information is formally documented and made available to OSS community. Even though tacit knowledge to some extent is converted to explicit but it remains in unstructured form and is not readily available for reuse. Further, it is also time consuming for the contributors to search for the required information through unstructured communication archives. The combination and internalisation mode of SECI are not traceable in OSS projects. Knowledge loss occurs when during the process of knowledge creation tacit knowledge is not made explicit and is not retained for future reuse [39].

2.3.2 Knowledge Sharing in OSS Projects

In OSS projects, knowledge sharing is an ongoing activity in an intensely people-oriented and self-organised community [40]. As we shall demonstrate, this activity might also be considered to be characterised as both reactive and somewhat disorganised. In such a setup, knowledge is dispersed in the community of contributors interacting on a project and is not limited within a small group [41]. Knowledge sharing is through asynchronous means of communication and with a collection of artifacts, which are publicly available for reuse.

Knowledge is stored in repositories namely Concurrent Versions Systems (CVS), Subversion (SVN), Frequently Asked Questions (FAQs), project websites, blogs, bug reporting and bug tracking databases (e.g. Bug Tracking System BTS) . Knowledge is

also believed to be archived in the artefacts available to public such as mailing lists and knowledge sharing can be quantified by analysing the mailing lists exchange among the listed members in OSS projects [40]. The contributors in CSS share software coding knowledge as a part of their job, while contributors in OSS share knowledge voluntarily [42]. It is argued that contributors in OSS communities involve in free advice and tacit knowledge sharing to a larger extent than formal CSS organisations [42]. In OSS communities knowledge sharing can be associated to social motivation [43]. Social motivation such as supportive behaviour influences the behaviour of contributors and their performance. There is also intrinsic motivation for the knowledge provider such as altruism or learning, by helping others solve problems.

Connecting contributors in a social network also enhances mutual knowledge and skills among them [32]. A strong social network and without any extrinsic reward system may result in effective knowledge sharing [42]. A formal coordination mechanism can provide better visibility of contributions from other team members. Contributors can be more informed about the contributions made by others members working in OSS projects.

Gamification is another emerging form of knowledge sharing in OSS communities [44]. The community members vote upon the questions and answers posted on a site, the numbers of votes reflect on the poster's reputation and seen as a measure of their expertise by the potential employer. Gamification element on sites is found to have increased the engagement of the participants and popularity of the site. In OSS community gamification element is argued to provide a better visibility of contributors activities [44].

The social media sites also serve for contributors to learn, collaborate, share knowledge and interact with users of software [44]. Contributors contribute on software development social media sites such as GitHub for coding, Jira to track issues, StackExchange network for question answer website, StackOverflow for professional programmers, and CrossValidated for statisticians and data miners. Although knowledge sharing activates are taking place in OSS projects, the mechanisms to articulate tacit knowledge are non-existing.

2.3.3 Knowledge Barriers and Knowledge Retention in OSS

While *knowledge barriers* cause inhibition in the innovation and learning process of organisations, knowledge retention (KR) is the ultimate goal of an organisation striving to innovate and improve performance. The inaccessibility to a certain kind of knowledge can delay the contributions on development activities by a contributor [45]. We focus on two kinds of barriers namely *contribution barriers* and *knowledge sharing barriers*. The limited knowledge of programming language, difficulty level of algorithms, complexity of technologies and source code used in OSS, can cause contribution barriers [46]. Computer languages are complex and difficult to learn with intertwined modules, so an understanding of existing architecture is required to contribute to an inter-dependent module [46]. The barriers for the newcomers to contribute in a OSS project are the lack of knowledge on project practices, lack of documentation, understanding information flow, unclear comments, and outdated documentation [45].

Knowledge sharing barriers are categorised into three levels: individual, organisational and technological [47]. Individual level barriers that limit knowledge sharing are a lack of time, lack of trust, a person who is unconsciously not aware of the possessed knowledge and lack of social network. While discussing distributed global communities to facilitate knowledge, language barriers, lack of common terminology, and lack of trust all inhibit knowledge sharing [12]. On the organisational level, barriers including non-supporting environment and culture lead to unsuccessful knowledge sharing. On the technological level, barriers to knowledge sharing are a lack of training, lack of communication on the benefits of technology, unsuitable technology, and reluctance to use technology.

The removal of obstacles due to knowledge flow in projects has the potential for a decrease in labour cost, improved schedule observance, and better final product quality [48]. The top five problematic knowledge flows were divided into two categories: difficulties with the online storage and retrieval of documents, and intra-team communication. The first category relates to explicit knowledge flow problems, while the second relates to tacit knowledge [48]. In addition to the identification of knowledge barriers, KR processes are also required within an organisation for knowledge to be accessible for the future reuse. In OSS projects, KR processes do not exist as in CSS organisations. Knowledge retention relates to capturing knowledge in an organisation and is an important aspect of KM. Knowledge retention mainly comes into focus when an employee is leaving. Three things indicate the need of a KR mechanism in an organisation: Lack of knowledge and an overly long time to acquire it is due to steep learning curves; People repeating mistakes and performing rework because they forget what they learned from previous projects; Chances of individuals owning key knowledge becoming unavailable [49].

Knowledge retention can be seen as a way of embedding and enabling knowledge within an organisation and a critical factor for sustainable performance [50]. It is an effort-demanding task to identify potential knowledge for the organisation. The structure of the organisation in the context of how well it supports knowledge retention is of importance. Once the person who has the potential knowledge leaves the organisation, it is hard to retain this knowledge.

Codification and personalisation are considered useful strategies for knowledge bases to be further used in knowledge intensive activities like software development [51]. In knowledge bases, codification captures electronic information and personalisation deals with the ways humans' use and process knowledge. Organizations implement codification strategy to encourage the reuse of explicit knowledge. The core techniques designed to retain knowledge in an organisation are mainly dependent on its knowledge-sharing practices. The techniques that facilitate knowledge capture, sharing, and reapplication are after-action reviews, communities of practice, face-to-face meetings, mentoring programs, expert referral services, video conferencing, interviews, written reports, use of training and technology-based systems to transfer the knowledge [12].

3 Discussion

In a large, geographically dispersed and dynamic OSS community, contributors vary in their skills and experiences. The quality of contributions (mostly the source code)

on the projects reflects a contributor's expertise and skills [15]. Knowledge sharing in OSS communities is mainly by asynchronous communication and typically involves mailing lists, blogs, forums, and Internet Relay Chat (IRC). Researchers have utilised OSS project mailing list data in various studies and it is thought to be one of the primary communication mechanisms in OSS projects [52]. However, the knowledge shared suffers from only partial coverage [17] and it can lack effective levels of organisation. OSS project knowledge may be abruptly lost when volunteers cease to contribute, and with knowledge not shared (existing in tacit form), the impact on the overall health of the project can be very damaging [17]. In effect, the stability of OSS projects and their success are dependent on contributor retention [53], or perhaps more precisely on the retention of knowledge contributor [54] either through directly sustaining contributors on the project or by co-opting individual knowledge into the collective knowledge sphere.

The removal of knowledge flow obstacles in projects has the potential for a decrease in labour cost, improved schedule observance, and better final product quality [48]. We propose that certain proactive knowledge acquisition practices will reduce the total cost of knowledge exchange in OSS projects, thereby improving the project productivity. Techniques to identify the critical knowledge will be a necessary first step to improving the current position, though we expect that there will be a challenge in striking the appropriate balance between proactive and reactive knowledge management, and this must somehow take account of the preference of contributors for these two different styles of knowledge exchange.

As we have demonstrated, OSS communities are mainly composed of volunteers who cannot be constrained to work permanently on the project [18] or to share their knowledge. The challenging task is how to orchestrate knowledge management in such a dynamic and dispersed community as OSS, especially as open source projects become larger and more widely adopted. This we suggest is not just a concern for the custodians of OSS projects but also for the consumers of the OSS itself. A private company may be motivated not just by the immediate cost saving in adopting an OSS project, but they may also be concerned with the maintainability of the OSS into the future as a strategic product development decision. In this respect, we envisage that a set of OSS knowledge management principles may be a product of our research and we have already undertaken some limited work in this direction.

Having established the absence of research on knowledge loss in the OSS project space, we propose to undertake a sustained investigation of this problem and to aid this exercise; we have established the following two research questions. We expect that further research questions will be identified as our research evolves.

- **RQ1.** Which knowledge management practices enable an effective knowledge management strategy for OSS projects?
- **RQ2.** How can knowledge management practices be integrated with established work practices in OSS projects?

4 Conclusion

From our review of the related literature, we conclude that knowledge management in OSS projects has received only indirect or superficial treatment, and we have found

no single substantial examination of the reactive and proactive knowledge strategy for OSS projects. In OSS projects, contributors are not obliged to notify the project community when they leave. The general mechanism of knowledge retention in software firms may sometimes be reactive in nature, triggered when an employee is leaving but even then, the opportunity for knowledge repatriation into the organisation will endure at least to the extent that the employee is cooperative and within the notice period that is typical in contemporary employment contracts. Conversely, in OSS projects a contributor may simply fall off the project radar – without notice and perhaps also unnoticed by the project – thereby eliminating any opportunity for reactive knowledge repatriation. Therefore, a proactive approach to retain knowledge is instinctively appealing for OSS projects.

In summary, we have investigated the published literature into knowledge loss in OSS projects, finding that there has been insufficient treatment of this concern to date. We also find that given the nature of OSS projects, proactive knowledge management mechanisms may be especially important, for example, because of the highly fragmented and transient nature of OSS project contributors. Given the popularity of OSS and its widespread and growing adoption, we believe that there is benefit in examining mechanisms to promote proactive knowledge management in OSS projects, and that these benefits can be shared by both contributors to and consumers of OSS.

Acknowledgments: This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero, the Irish Software Research Centre (www.lero.ie).

References

1. Feller, J., Fitzgerald, B.: Understanding open source software development. Addison-Wesley London (2002)
2. Crowston, K., Howison, J., Annabi, H.: Information systems success in free and open source software development: theory and measures. Software Process: Improvement and Practice 11, 123-148 (2006)
3. Subramaniam, C., Sen, R., Nelson, M.L.: Determinants of open source software project success: A longitudinal study. Decision Support Systems 46, 576-585 (2009)
4. Crowston, K., Wei, K., Howison, J., Wiggins, A.: Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. Acm Computing Surveys 44, (2012)
5. 78% Of Companies Run On Open Source Yet Lack Formal Policies | Black Duck Software. Black Duck Software. N.p., 2017. Web. 8 June 2017.
6. Michlmayr, M.: Quality Improvement in Volunteer Free and Open Source Software Projects: Exploring the Impact of Release Management. . University of Cambridge (2007)
7. Capiluppi, A., Gonzalez-Barahona, J.M., Herraiz, I., Robles, G.: Adapting the "staged model for software evolution" to free/libre/open source software. Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting, pp. 79-82. ACM, Dubrovnik, Croatia (2007)
8. Robles, G., Gonzalez-Barahona, J.M.: Contributor turnover in libre software projects. IFIP International Federation for Information Processing, vol. 203, pp. 273-286 (2006)
9. Davenport, T.H., Prusak, L.: Working knowledge: How organizations manage what they know. Harvard Business Press (1998)
10. Zack, M.H.: Managing Codified Knowledge. Sloan Management Review 40, 45-58 (1999)

11. Nonaka, I., Toyama, R., Konno, N.: SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planning 33, 5-34 (2000)
12. De Long, D.W., Davenport, T.: Better practices for retaining organizational knowledge: Lessons from the leading edge. Employment Relations Today 30, 51-63 (2003)
13. Jennex, M.E., Durcikova, A.: Assessing knowledge loss risk. In: System Sciences (HICSS), 2013 46th Hawaii International Conference on, pp. 3478-3487. IEEE, (2013)
14. Viana, D., Conte, T., Marczak, S., Ferreira, R., Souza, C.d.: Knowledge Creation and Loss within a Software Organization: An Exploratory Case Study. In: System Sciences (HICSS), 2015 48th Hawaii International Conference on, pp. 3980-3989. (2015)
15. Donadelli, S.M.: The impact of knowledge loss on software projects: turnover, customer found defects, and dormant files. Software Engineering, pp. 85. Concordia University (2015)
16. Rigby, P.C., Zhu, Y.C., Donadelli, S.M., Mockus, A.: Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a project at Avaya. In: Proceedings of the 2016 International Conference on Software Engineering. (2016)
17. Izquierdo-Cortazar, D., Robles, G., Ortega, F., Gonzalez-Barahona, J.M.: Using software archaeology to measure knowledge loss in software projects due to developer turnover. In: System Sciences, 42nd Hawaii International Conference on, pp. 1-10. IEEE, (2009)
18. Robles, G., Gonzalez-Barahona, J.M., Michlmayr, M.: Evolution of volunteer participation in libre software projects: evidence from Debian. In: Proceedings of the 1st International Conference on Open Source Systems, pp. 100-107. (2005)
19. Dingsoyr, T., Bjornson, F.O., Shull, F.: What Do We Know about Knowledge Management? Practical Implications for Software Engineering. Software, IEEE 26, 100-103 (2009)
20. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, pp. 38. ACM, (2014)
21. Foucault, M., Palyart, M., Blanc, X., Murphy, G.C., Falleri, J.-R.: Impact of developer turnover on quality in open-source software. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 829-841. ACM, Bergamo, Italy (2015)
22. von Krogh, G., Spaeth, S., Haefliger, S.: Knowledge reuse in open source software: An exploratory study of 15 open source projects. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pp. 198b-198b. IEEE, (2005)
23. Herraiz, I., Robles, G., Amor, J.J., Romera, T., González Barahona, J.M.: The processes of joining in global distributed software projects. Proceedings of the 2006 international workshop on Global software development for the practitioner, pp. 27-33. ACM, Shanghai, China (2006)
24. Zhou, M., Mockus, A.: Developer fluency: achieving true mastery in software projects. Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, pp. 137-146. ACM, Santa Fe, New Mexico, USA (2010)
25. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions onSoft. Eng. 29, 481-494 (2003)
26. Liu, W., Chen, C.L., Lakshminarayanan, V., Perry, D.E.: A design for evidence - based soft research. SIGSOFT Softw. Eng. Notes 30, 1-7 (2005)
27. Otte, T., Moreton, R., Knoell, H.D.: Applied quality assurance methods under the open source development model. In: Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International, pp. 1247-1252. (2008)
28. Capiluppi, A., Michlmayr, M.: From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds.) Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software, June 11–14, 2007, Limerick, Ireland, pp. 31-44. Springer US, Boston, MA (2007)

29. Rigby, P.C., German, D.M., Cowen, L., Storey, M.-A.: Peer Review on Open-Source Software Projects: Parameters, Statistical Models, and Theory. ACM Trans. Softw. Eng. Methodol. 23, 1-33 (2014)
30. Crowston, K., Howison, J.: The social structure of free and open source software development. First Monday 2, (2005)
31. Lee, G.K., Cole, R.E.: From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. Organization Science 14, 633-649 (2003)
32. Crowston, K.: Lessons from volunteering and free/libre open source software development for the future of work. Researching the Future in Information Systems, pp. 215-229. Springer (2011)
33. Schilling, A., Laumer, S., Weitzel, T.: Is the source strong with you? A fit perspective to predict sustained participation of FLOSS developers. In: International Conference on Information Systems 2011, ICIS 2011, pp. 1620-1630. (2011)
34. Michlmayr, M., Robles, G., Gonzalez-Barahona, J.M.: Volunteers in large libre software projects: A quantitative analysis over time. Emerging Free and Open Source Software Practices 1-24 (2007)
35. Xu, B.: Volunteers' Participative Behaviors in Open Source Software Development: The Role of Extrinsic Incentive, Intrinsic Motivation and Relational Social Capital. Texas Tech University (2006)
36. Fitzgerald, B.: The transformation of open source software. MIS Quarterly 587-598 (2006)
37. Yunwen, Y., Yamamoto, Y., Kishida, K.: Dynamic community: a new conceptual framework for supporting knowledge collaboration in software development. In: Software Engineering Conference, 2004. 11th Asia-Pacific, pp. 472-481. (2004)
38. Mockus, A., Fielding, R.T., Herbsleb, J.: A case study of open source software development: the Apache server. Proceedings of the 22nd international conference on Software engineering, pp. 263-272. ACM, Limerick, Ireland (2000)
39. Assimakopoulos, D., Yan, J.: Sources of knowledge acquisition for Chinese software engineers. R&D Management 36, 97-106 (2006)
40. Sowe, S.K., Stamelos, I., Angelis, L.: Understanding knowledge sharing activities in free/open source software projects: An empirical study. Journal of Systems and Software 81, 431-446 (2008)
41. Ye, Y., Kishida, K.: Toward an understanding of the motivation Open Source Software developers. Proceedings of the 25th International Conference on Software Engineering, pp. 419-429. IEEE Computer Society, Portland, Oregon (2003)
42. Endres, M.L., Endres, S.P., Chowdhury, S.K., Alam, I.: Tacit knowledge sharing, self-efficacy theory, and application to the Open Source community. Journal of Knowledge Management 11, 92-103 (2007)
43. Licorish, S.A., MacDonell, S.G.: Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. Information and Software Technology 56, 1578-1596 (2014)
44. Vasilescu, B., Serebrenik, A., Devanbu, P., Filkov, V.: How social Q&A sites are changing knowledge sharing in open source software communities. Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, pp. 342-354. ACM, Baltimore, Maryland, USA (2014)
45. Steinmacher, I., Silva, M.A.G., Gerosa, M.A., Redmiles, D.F.: A systematic literature review on the barriers faced by newcomers to open source software projects. Information and Software Technology 59, 67-85 (2015)
46. von Krogh, G., Spaeth, S., Lakhani, K.R.: Community, joining, and specialization in open source software innovation: a case study. Research Policy 32, 1217-1241 (2003)
47. Kukko, M., Helander, N.: Knowledge Sharing Barriers in Growing Software Companies. In: System Science, 45th Hawaii International Conference on, pp. 3756-3765. (2012)

48. Mitchell, S.M., Seaman, C.B.: Software process improvement through the identification and removal of project-level knowledge flow obstacles. Proceedings of the 34th International Conference on Software Engineering, pp. 1265-1268. IEEE (2012)
49. Lindvall, M., Rus, I.: Knowledge Management for Software Organizations. In: Aurum, A., Jeffery, R., Wohlin, C., Handzic, M. (eds.) Managing Software Engineering Knowledge, pp. 73-94. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
50. Doan, Q.M., Rosenthal-Sabroux, C., Grundstein, M.: A Reference Model for Knowledge Retention within Small and Medium-sized Enterprises. In: KMIS, pp. 306-311. (2011)
51. Donnellan, B., Fitzgerald, B., Lake, B., Sturdy, J.: Implementing an open source knowledge base. IEEE Software 22, 92-95 (2005)
52. Sharif, K.Y., English, M., Ali, N., Exton, C., Collins, J.J., Buckley, J.: An empirically-based characterization and quantification of information seeking through mailing lists during Open Source developers' software evolution. Information and Software Technology 57, 77-94 (2015)
53. Schilling, A., Laumer, S., Weitzel, T.: Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects. In: System Science, 45th Hawaii International Conference on, pp. 3446-3455. (2012)
54. Ayushi, R., Ashish, S.: What Community Contribution Pattern Says about Stability of Software Project? In: Software Engineering Conference, 21st Asia-Pacific, pp. 31-34. (2014)