A Systematic Examination of Knowledge Loss in Open Source Software Projects

1. Introduction

Software development is a knowledge-intensive activity, which involves intense complexity (Clarke *et al.* 2016). Open Source Software (OSS) has had a profound impact on the way in which software is developed and consequently on the perception of software development (Hagan *et al.* 2007). Open source software is one of the representative examples of open collaboration (Lee *et al.* 2017). Many researchers have pointed out that the open source movement is an interesting phenomenon that is difficult to explain with conventional economic theories (Andersen-Gott *et al.* 2012). In Open Source Software (OSS) projects, contributors can be volunteers or paid workers who participate in software development activities. While working on OSS projects, contributors acquire project related knowledge and gain experience and skills. Examples of knowledge that is required to accomplish software development tasks on projects include application domain, system's architecture, use of particular algorithms to code, insights into requirements, programming language and development environment (Anquetil *et al.* 2007). Valuable individualistic knowledge, which remains unshared with others, is lost once contributors leave the project. Organisations constantly face the problem of knowledge loss as employees leave (De Long and Davenport 2003; Jennex and Durcikova 2013; Viana *et al.* 2015), a situation which is perhaps exasperated in OSS projects (Izquierdo-Cortazar *et al.* 2009; Donadelli 2015; Rigby *et al.* 2016) where most (if not all) contributors are volunteers with largely unpredictable engagement durations (Robles *et al.* 2005). The phenomenon of volunteers joining and leaving at their discretion is more common in OSS projects than with hired employees in Closed Source Software (CSS) (Robles *et al.* 2005). Such contributor attrition leads to knowledge loss on OSS projects.

The importance of OSS in our daily lives can be realised from the fact that there are thousands of OSS projects operating worldwide such as the Linux operating system, Apache Web Server, Mozilla Firefox, OpenOffice and many more. There has been an exponential rise in OSS products and their use, as indicated by 430,000 projects hosted in 2014 on the SourceForge portal (Silic and Back 2017). The projects are of varying sizes and also involve commercial firms who are heavily depend on OSS systems (Crowston *et al.* 2012). During 2014, Google's mobile Android operating system had about one billion users across all devices (Conn 2014). 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 in creating software for customers (BlackDuck 2015). The use of Apache and NGINX is calculated to be 54% of all webservers used worldwide (Silic and Back 2017).

Many technology firms traditionally known for closed organizational structures and propriety software development, such as IBM, Microsoft, and Facebook, have embraced the strategic opportunities that open source development models offer (Daniel *et al.* 2018). Currently it is reported that 40% of Fortune 50 companies are developing software systems using open source software, with firms such as Microsoft and Facebook being the most active open source development communities on GitHub. For example Microsoft boasting the highest number of contributors and more than 258 software projects under development (Octoverse 2016).

The phenomenon where contributors working in project teams join, leave or change their role is referred to as ‘turnover’ (Foucault *et al.* 2015). The turnover can be catastrophic for the project if a contributor who is knowledgeable on major parts of the system leaves and this reduces the spread of the knowledge (Donadelli 2015). Contributor turnover is rated as being very high both in the software industry (Zhou 2009) and in OSS projects (Robles and Gonzalez-Barahona 2006; Otte *et al.* 2008; Foucault *et al.* 2015; Rigby *et al.* 2016) and mitigating its effects is considered a significant problem (Fronza *et al.* 2013). Turnover is inevitable due to the transient nature of OSS project workforces (Michlmayr 2007; Yu *et al.* 2012), causing knowledge loss (Izquierdo-Cortazar *et al.* 2009; Rigby *et al.* 2016). Knowledge loss in this work refers to the loss of experience and expertise in OSS projects that can result in the decline of evolution in OSS systems (Joblin *et al.* 2017). Knowledge loss not only impacts software quality (Mockus 2010; Foucault *et al.* 2015) and contributor productivity (Izquierdo-Cortazar *et al.* 2009; Schilling *et al.* 2011), but might also threaten the overall sustainability of OSS projects.

OSS projects are constantly evolving as indicated in the adaptated staged model for OSS systems (Capiluppi *et al.* 2012), and maintenance plays a significant role in project evolution (Rigby *et al.* 2016; Lin *et al.* 2017). As asserted “it is harder to separate out maintenance and development since they tend to occur together (Michlmayr 2007)”. A system that stops evolving is an indication that it may become a legacy in the near future (Capiluppi *et al.* 2012). Taking the view of software development and maintenance being part of the broader phenomenon of software evolution, it is argued that the adoption of knowledge management practices in software engineering would improve both software construction and, more particularly, software maintenance (de Vasconcelos *et al.* 2017).

Knowledge Management (KM) processes play a significant role in the implementation of various Information Systems (IS). Researchers have introduced different KM processes, each of which contributes to the efficient use of ISs (Al-Emran *et al.* 2018). In terms of IS type, most of the analysed studies focused on investigating the impact of KM processes on E-business systems, knowledge management systems and IS outsourcing respectively (Al-Emran *et al.* 2018). Effective knowledge management practices in organizations are focused on knowledge creation and knowledge transfer activities (Barão *et al.* 2017). KM processes are the essential elements for improving the capabilities of a particular technology, and the successful implementation of such technology increasingly depends on the efficient use of these processes (Lee *et al.* 2007). KM processes are considered the fundamental processes for the successful adoption and implementation of a new IS (Chong *et al.* 2013). For instance, information and knowledge can be seen as key resources for improving the internationalisation processes of small and medium-sized enterprises (SMEs) including collaboration, which is considered an important facilitator of these processes, particularly by nurturing information and knowledge sharing (Costa *et al.* 2016).

The objective of this work is to deeply and systematically investigate the phenomenon of knowledge loss due to contributor turnover in OSS projects as presented in the state-of-the-art literature and to synthesise the information presented on the topic. Furthermore, based on the learning arising from our investigation it is our intention to identify mechanisms to reduce the overall effects of knowledge loss in OSS projects.

To gain an insight into the phenomenon of knowledge loss in OSS projects due to turnover, we conducted a literature review. The research question is structured using the PICOC (Population, Intervention, Comparison, Outcome and Context) criteria to structure research questions for a Systematic Literature Review (SLR) (Petticrew and Roberts 2006), with only population, intervention, and outcome being relevant for our research question:

- Population: Open Source Software and associated synonyms
- Intervention: Knowledge loss and turnover
- Outcome: Existing themes and patterns related to knowledge loss and turnover in OSS projects

Our central research question is phrased as "*What is the existing state-of-the-art literature on knowledge loss due to turnover in OSS projects?*"

The remainder of the paper is structured as follows: Section 2 provides an insight into the setting up of OSS projects, work structure and knowledge relevant concepts. Section 3 presents the literature review methodology and Section 4 presents the application of the literature review methodology. Section 5 elaborates on problem identification and details the impact of knowledge loss in OSS projects. Section 6 comprises practices to reduce knowledge loss in OSS projects. Section 7 discusses limitations to this work and the final Section 8 marks the conclusion of this work with future directions.

2. Background

2.1 Open Source Software

Open Source Software (OSS) is enlisted as one of the four core components of Open Science (OS), which relates to the movement that provides free accessibility to scientific research data and its dissemination to all levels of inquiring society (Pontika *et al.* 2015). Open Source Software (OSS) is a term used to embrace software developed and released under an "open source" license that complies with Open Source Definition (OSD). The OSD uses either the shorter version based on a four point criteria as in Free Software Foundation (FSF) or the longer version based on a ten point criteria as in Open Source Initiative (OSI). The difference between the two definitions is one of language while the 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" (Feller and Fitzgerald 2002). Users with a technical inclination can use, freely access the code, inspect, modify and redistribute the software (Crowston *et al.* 2004; Crowston *et al.* 2006). However, the freedom to use and distribute source code from OSS varies based on which category of OSS license agreement applies (Scacchi 2007). There are two main categories of OSS licenses: copyleft and non-copyleft (DeBrie and Goeschel 2016). Copyleft or a restrictive license requires that any work based on copyleft license, when incorporated with other work, stay under the copyleft license.

While free software mostly identifies with GPL, OSS license agreements may vary based on the incorporation of the software that can be propriety or free. Another term used to represent free software is Free Open Source Software (FOSS). The term “free” in FOSS fails to express the notion of freedom and becomes invisible, with the main stress on open source software. In 2001, the European Commission (EC) used the term Free/Libre Open Source Software (FLOSS), for the first time to avoid taking sides in debate and stay neutral on the distinction between free software and open-source software.

Three differences exist between OSS and CSS or commercial software (Capiluppi *et al.* 2007). The first concerns the availability of commercial software releases to third parties only when it is complete, running, fully tested and authorised (Capiluppi *et al.* 2007). Conversely, OSS systems are available before the first official release while still in versioning system repositories, and at any instant might be downloaded (Michlmayr 2007). The second difference of OSS to CSS, relates to the presence of an evolution stage (on going software development) in OSS after the start of the servicing stage (no new functionalities are added but fixes are performed to existing system). The third difference noticed was the transition of the OSS from a phase out (no fixes of existing functionalities or addition of new ones) to evolution.

Software maintenance is an ongoing activity in software development, comprising several types of activities relating to fixing faults in the system, adapting the system to changes in the operating environment and adapting the system to changes in the original requirements Basili (1990). Software maintenance is the field which is concerned with the evolution of a software system after its initial release Michlmayr (2007). In OSS, maintenance and development (or evolution) are not considered as two separate phases of the software development cycle (Michlmayr 2007). Maintenance in an OSS project is compared to reinvention (Crowston *et al.* 2012) , which is “a continuous source of adaptation, learning and improvement in OSS functionality and quality” (Scacchi 2004). Overall maintenance activities in OSS projects are ongoing along with the evolution of the system.

2.2 Organisational Structure in OSS Projects

The development in OSS projects is distinguished from CSS or traditional software by the usage of the terms: ‘cathedral’ and ‘bazaar’ (Raymond 1999). In the cathedral, control is with one main person who controls progress. On the contrary, in the bazaar, the contributor decides when to contribute to a project (Crowston *et al.* 2004). In the cathedral phase, software development starts with a smaller group of developers and the source code is not shared or accessible to users. While in the bazaar phase, software development has a large number of volunteers who can access the source code and contribute such as adding new requirements, bug fixes, and reporting defects. It is argued that a typical OSS project starts with a cathedral development style and then transitions to a bazaar development style (Capiluppi and Michlmayr 2007). That said, cathedral and bazaar phases are not considered mutually exclusive in OSS development (Capiluppi and Michlmayr 2007).

In OSS, each project is considered similar to an organization as in a traditional software industry or CSS. The layered structure called an onion model represents the organisational structure in the OSS community (Dinh-Trong and Bieman 2004; Crowston and Howison 2005). The teams in OSS have a hierarchical onion-like structure, consisting of core, co-developer, active users, and passive users (Mockus *et al.* 2002; Crowston *et al.* 2004). The core is a small group of contributors who are responsible for most of the code and ensure the design and evolution of the project. Co-developers contribute by reviewing or modifying the code or providing bug fixes. Active users use the recent release and contribute bug reports or feature requests but do not contribute code. The farthest group of members from core is passive users and their number is difficult to predict (Crowston *et al.* 2004). The representation of contributors in OSS projects can deviate from the above presented onion model. For instance, Linux developers organise themselves into two groups, ‘core’ and ‘periphery’ (Lee and Cole 2003). The core consists of the project leader and hundreds of maintainers. Periphery is a large group of developers further divided into two teams: development and bug reporting. Core contributors go through all roles starting as a user and progressing to be among the core group of contributors (Ye *et al.* 2005). There can be different paths that may be followed to become a core contributor. It is reported that volunteers joining the OSS project follow the onion model and climb the ladder to become a core contributor based on the meritocracy, while hired developers are integrated into the project faster (Herraiz 2006).

OSS project collaborations can be of three types: community-based, non-profit organisation and commercially based. Community projects are online projects where contributors participate voluntarily and contribute their time as is suitable for them (Xu 2006). Volunteers collaborate on OSS projects during their free time without profiting directly by any economic incentives for their efforts (Robles *et al.* 2005). The motivation for volunteers to participate in OSS projects is to learn new skills, make code contributions and become known within the OSS community, which might pave the way to future career opportunities (Crowston 2011). Volunteers thrive on the intrinsic motivation, which relates to the feeling of satisfaction, competence, and fulfilment from code writing (Xu 2006; Schilling *et al.* 2011). There is also an intrinsic motivation for the knowledge provider, such as altruism or learning by helping others solve problem (Vasilescu *et al.* 2014). In OSS, it is hard to predict when a volunteer will leave a project. This unpredictable nature of commitment from volunteers creates an element of risk within OSS projects (Robles and Gonzalez-Barahona 2006). Managing volunteer contributors can cause certain problems not evident in traditional software development (Robles and Gonzalez-Barahona 2006). The development of free software slows or stalls in the absence of the lead developer who initiated the project or when a contributor decides to stop working on the project.

Community open source projects take their organisational form from an Internet-based community, and the developers are mostly the volunteers (Lee and Cole 2003). For instance, volunteers who manage the Apache project are in fact developers with a full-time job who work part-time on the Apache project. Debian project develops an open source operating system where all contributors are volunteers (Robles *et al.* 2005). The tasks performed by volunteers in Debian include maintaining software packages, supporting the server infrastructure, developing Debian-specific software, for instance, the installation routine and package management tool, translating documentation and Web pages. In non-profit organisation projects, developers are either paid workers or volunteers. The project is mature enough and is funded as a formal organisation. There is still some element of a community project maintained in such projects, for example, the Apache Software Foundation (Xu 2006). In commercially involved projects, a software company sponsors projects and employs the majority of contributors. A commercial company, Netscape, managed the Mozilla project in the past. The developers hired for the project worked full-time and for pay (Dinh-Trong and Bieman 2004). Companies like IBM, HP, SUN (now acquired by Oracle), sponsor OSS projects in which major contributors are paid developers (Fitzgerald 2006).

The organizational structure of OSS projects is considered to be highly dynamic compared to CSS or traditional software development organizations (Jensen and Scacchi 2005). Furthermore, the contributors in OSS projects are transient in nature. While Internet-based knowledge communities are great avenues for contributors to learn and expand their skill sets, it is argued that they influence job hopping behaviour due to the availability of more opportunities, since it sends a strong signal to a potential employer of the contributors expertise and skills (Huang and Zhang 2016). OSS projects can be a hub of innovation and evolving ecosystems with the involvement of commercial organisations (Capiluppi *et al.* 2012; Crowston *et al.* 2012), which align their strategic goals with the product development in OSS projects. OSS communities operate on globally distributed and virtual environments using the Open Source Software Development (OSSD) model (Jensen and Scacchi 2005). The development in OSS projects is independent, self-assigned and in parallel streams without much coordination due to geographical dispersion (Michlmayr 2007). In OSS, tasks are not assigned and contributors make contributions based on their interest and discretion (Mockus *et al.* 2002).

The collaboration and communication is through asynchronous means, facilitated by technology-mediated channels (Sowe *et al.* 2008; Vasilescu *et al.* 2014; Sharif *et al.* 2015). In an informal, loosely defined community, reciprocity is highly relevant to OSS development (Kuk 2006). It is argued that interaction is of a strategic nature between individual developers and highly resourceful developers. This leads to the formation of smaller but better organized structures in OSS development (Kuk 2006). Some measures suggested to gauge the impact of knowledge sharing communities include participation inequality, conversational interactivity, and cross-thread connectivity (Kuk 2006). Participation inequality, if it exists to a certain level, has a positive impact on knowledge sharing. Moreover, strategic interaction expands knowledge sharing through reciprocal and overlapping activities (Kuk 2006).

Epistemic interactions in OSS development involves a balancing act between exploration and exploitation activities (Lee and Cole 2003). As an example, the successful Linux kernel development project has adopted a simple two-tier structure for interaction through mailing lists. The first tier consists of the peripheral mailing lists useful for innovation and for criticisms to encourage learning and quality control. The second tier consists of the core mailing lists, helpful for selecting and retaining the features, applications and codes for the future incremental additions to the next product release (Kuk 2006).

Altogether, OSS projects have an organizational outlook of an extreme example of large-scale Global Software Development (GSD) (Mockus *et al.* 2002). In such a dense environment where interaction and networking among community members is complex and spreads across various online communities, contributor departure can cause serious damage to the OSS project. We depict the organisational structure of an OSS project as a mind map in Figure 1. In the parenthesis reference to respective study is given.

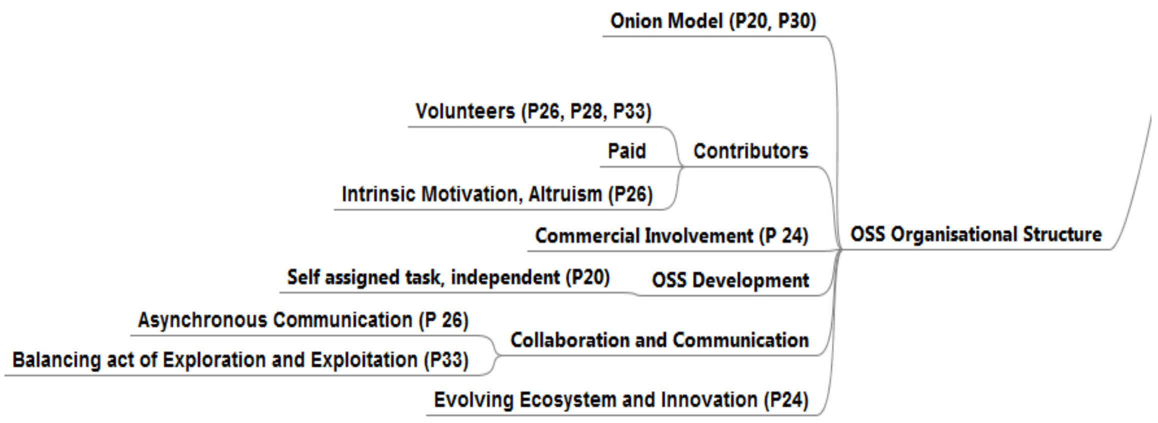
Onion Model (P20, P30)
Volunteers (P26, P28, P33)
Paid
Contributors
Intrinsic Motivation, Altruism (P26)
Commercial Involvement (P 24)
OSS Organisational Structure
Self assigned task, independent (P20)
OSS Development
Asynchronous Communication (P 26)
Collaboration and Communication
Balancing act of Exploration and Exploitation (P33)
Evolving Ecosystem and Innovation (P24)

Figure 1: Mind map of the Organisation Structure in OSS projects

2.3 Data, Information, and Knowledge

It is important to clarify the relationship between "data", "information", and "knowledge". Data can be considered the raw material for information, which, in turn, can be considered the raw material for knowledge (Zins 2007). Data represent observations and facts without any context or meaning and information is the result of associating data in a meaningful context (Michael H. Zack 1999). In order to convert data into information, it must be contextualized, categorised, calculated and condensed (Davenport and Prusak 1998). There are different views on the definition of knowledge. Nonaka defines knowledge as "a justified belief that that increases an entity's capacity for effective action" (Nonaka 1994)., Using state of mind perspective, Schubert et al define "knowledge by emphasising, knowing and understanding through experience and study" (Schubert *et al.* 1998). Knowledge is indicated to be generated when information is combined with context and experience and considered a fluid mix of framed experience, values, contextual information, expert insight and grounded intuition that provides an environment and framework for evaluating and incorporating new experience and information (Vinogradova and Galandere-Zile 2005). This research refers to knowledge as experience and expertise built by a contributor that evolves from the day-to-day interaction in OSS projects.
Knowledge is driven from information (Davenport and Prusak 1998). Knowledge is the product of an individual's experience and accumulates because of communication or inference (Michael H. Zack 1999). Knowledge storage is argued to be impossible and it is asserted that information leads to knowledge transformation (Aggestam *et al.* 2010). When information combines with context and experience, the resultant knowledge enables effective decision making as depicted in Figure 2.

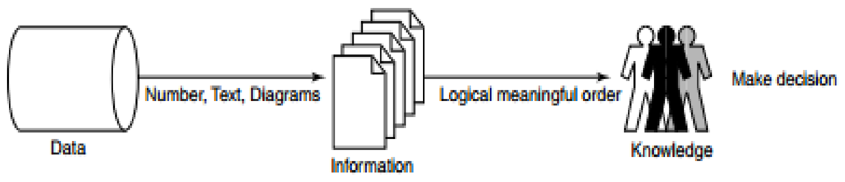

Figure 2: Data- information- knowledge (Rhem 2005)

Software development is a knowledge-intensive activity and involves knowledge integration from several domains, such as software architecture, software design methodologies and business application (Tiwana 2004). Knowledge is also created and shared during the software development process, when different facets such as products, tools, processes and people interact with each other (Qumer and Henderson-Sellers 2008).

In order to solve a problem, a software developer has to understand it, design a solution, program the solution, and test it (Pennington 1987). During problem solving, knowledge is required in various forms comprising of real world domain, programming language syntax, program and programming conventions, features that have an influence on program implementation and the expected users of the program (Pennington 1987).

Knowledge is differentiated into two types: 'explicit' and 'tacit' (or implicit) (Nonaka and Takeuchi 1995; Sowe *et al.* 2005). Explicit knowledge is formalised, codified and documented while tacit knowledge emerges from social interactions among peers (Hutchison 2001). Tacit knowledge comprises skills learned due to the personal capabilities of contributors and if not documented remain confined to an individual (Michael H Zack 1999). Therefore, it is difficult to code, articulate and transfer tacit knowledge because it is retained with the person (Sowe *et al.* 2008). Domain knowledge required by software developers to perform their job roles is mainly tacit (Ryan and O'Connor 2013). Explicit knowledge, on the other hand, is available in written form (Nonaka *et al.* 2000).

In an increasingly globalized and hyper connected business environment, using knowledge strategically is often critical for competitive performance (Venkitachalam and Willmott 2017). Continuous innovation is a key ingredient in maintaining a competitive advantage in the current dynamic and demanding marketplace. It requires an organization to regularly update and create knowledge for the current generation, and reuse it later for the next generation of a product (Tyagi *et al.* 2015).

In this section, we discussed OSS and different terms associated with it, the organisational structure of OSS projects, and their difference from CSS organisations. Furthermore, we explain the role of knowledge in software development and the relationship between data, information, and knowledge. In the next section, we elaborate on the literature review methodology, which was followed to investigate the phenomenon of knowledge loss due to contributor turnover in OSS projects.

3. Literature Review Methodology

In order to find the relevant literature on the topic of knowledge loss in OSS, we conducted our literature review using the snowballing approach (SB) (Jalali and Wohlin 2012; Wohlin 2014; Badampudi *et al.* 2015). Systematic literature studies have become common in software engineering, and hence it is important to conduct them efficiently and reliably (Wohlin 2014). In software engineering, the main recommended first step is using search strings in a number of databases, while in information systems, snowballing has been recommended as the first step (Jalali and Wohlin 2012). Snowballing refers to using the reference list of a paper or the citations to the paper to identify additional papers and benefits systematic literature review from not only looking at the reference lists and citations, but to complement it with a systematic way of looking at where papers are actually referenced and where papers are cited (Wohlin 2014). The SB process is executed in iterations starting from a baseline set of papers using backward and forward snowballing. Backward snowballing involves looking through the reference list of the baseline papers and forward snowballing searches for studies that cite papers from the baseline set. The snowballing process comes to an end once no new papers are found on the relevant topic (Felizardo *et al.* 2016). The efficiency of SB might be higher when the keywords for searching include general terms (Jalali and Wohlin 2012). Snowballing is an efficient and reliable way to conduct a systematic literature review, providing a robust alternative to mechanically searching individual databases for given topics (Wohlin 2014). We followed the step-by-step process illustrated by Wohlin (Wohlin 2014), in which he affirms that the snowballing approach finds papers almost similar to the ones in the conventional style of systematic literature review, which involves database search.

The snowballing literature review methodology specified the research question, search strategy, inclusion, exclusion, quality criteria, and data synthesis. The search strategy, inclusion, exclusion and quality criteria, are applied as a part of the snowballing procedure, as explained in the next section. Execution of the snowballing procedure was followed by data synthesis. A visualisation of our approach to the literature review methodology is depicted in Figure 3.

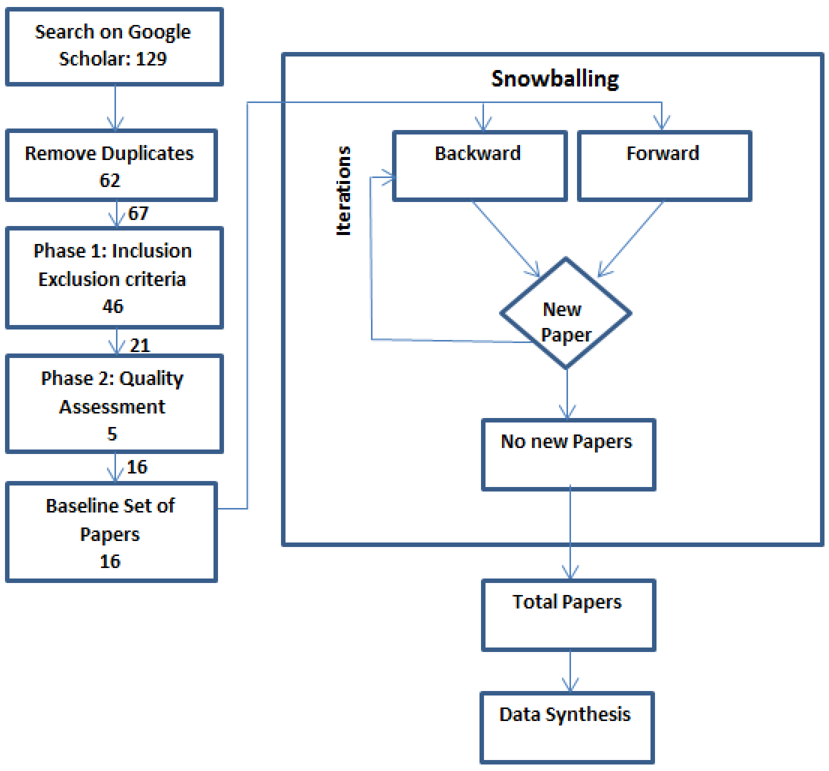
Search on Google Scholar: 129
Remove Duplicates 62
67
Phase 1: Inclusion Exclusion criteria 46
21
Phase 2: Quality Assessment 5
16
Baseline Set of Papers 16
Snowballing
Iterations
Backward
Forward
New Paper
No new Papers
Total Papers
Data Synthesis

Figure 3: Depiction of Literature Review Methodology (modified from (Wohlin 2014))

3.1 Details on Snowballing Procedure

In this section, we explain the details of the SB procedure to search relevant papers. The selection of a baseline set of papers and iterations in SB will rigorously follow the given criteria for inclusion, exclusion, and quality assessment. A detailed overview of the SB procedure that we have utilised to find relevant papers is illustrated in Figure 4. The initial baseline papers are pooled from search engines using search string(s) designed to reflect the topic. The final baseline papers are selected from a pool of initial baseline papers, which are then subjected to subsequent iterations using backward and forward snowballing. Papers found in iteration are assessed based on relevancy to the topic using inclusion, exclusion, and quality criteria. Consequently, new papers are added to the set of final baseline papers. The iterations continue until no new papers are found.

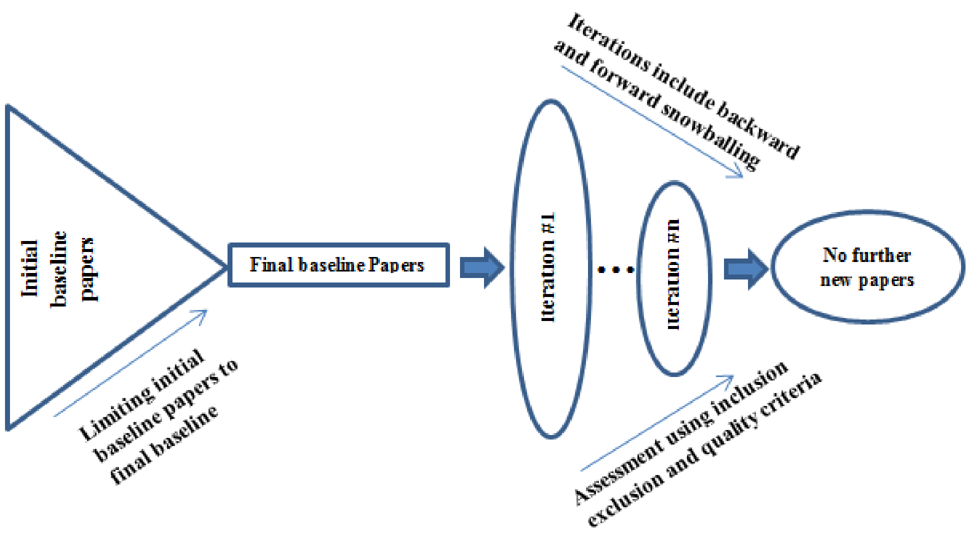
Initial baseline papers
Final baseline Papers
Limiting initial baseline papers to final baseline
Iteration #1
Iteration #n
No further new papers
Iterations include backward and forward snowballing
Assessment using inclusion exclusion and quality criteria

Figure 4: The detailed view of snowballing procedure to search papers

3.1.1 Initial Baseline set of papers

For the purpose of this search, we make no distinction in the following terms: Free/Libre Open Source Software (FLOSS), Open Source Software (OSS), Free Open Source Software (FOSS), and Free Software. The search strings are designed using a combination of all three terms to represent open source software. Google Scholar is used to search each string to avoid bias in the selection of publishers (Wohlin 2014). Table 1 shows the five strings designed to be executed on Google Scholar, to extract an initial baseline set in SB.

Table 1: Five search strings designed to extract initial baseline set in SB

Open source software and turnover and "knowledge loss"
Free open source software and turnover and "knowledge loss"
Free software and turnover and "knowledge loss"
Free/ Libre software and turnover and "knowledge loss"
Libre software and turnover and "knowledge loss"

The snowballing approach outlined later in this section represents a comprehensive pursuit of related works that are identified in the initial baseline of relevant research papers. Accordingly, our methodology will consider the top 30 search results provided by Google Scholar for each of our search strings. Most likely Google Scholar search returns the best results in the first few pages. To establish the baseline set of papers we expect to collect 150 papers from Google Scholar using five search strings and then subject them to systematic inclusion / exclusion analysis, followed by extensive snowballing iterations.

3.1.2 Inclusion/Exclusion Criteria for Baseline Set

The initial baseline set of papers will be further assessed, to be included in the final baseline set, by applying inclusion or exclusion criteria in two phases. In the first phase, papers will be included based on the following criteria:

- The papers that are available in full-text [1]
- Papers written in English
- Papers are peer-reviewed
- Papers relevant to the topic are to be included after reading the title
- When it is unclear from the title abstract and introduction it will be read

In the second phase, we will evaluate the full texts of papers utilising quality assessment criteria to make a final decision on inclusion or exclusion of papers into the final baseline set.

3.2 Quality Assessment

The three main concerns relevant to quality assessment are rigour, credibility and relevance as defined below (Dybå and Dingsøyr 2008):

- Rigour: Has a thorough and appropriate approach been applied to baseline research methods in the study?
- Credibility: Are the findings well presented and meaningful?
- Relevance: How useful are the findings to the software industry and the research community?

The above three concerns are assessed through twelve recommended criteria (Dybå and Dingsøyr 2008). The quality assessment criteria given in (Dybå and Dingsøyr 2008) has been effectively used by Kitchenham and Brereton with some adaptions (Kitchenham and Brereton 2013). In this literature review, the main intention is to find literature relevant to knowledge loss due to contributor turnover in open source software. After reviewing the quality criteria presented by (Dybå and Dingsøyr 2008; Kitchenham and Brereton 2013), we formulate the following check list to assess papers for their quality assessment given in Table 2.

Table 2: Quality assessment criteria to evaluate papers

No.	Criteria Details
1	Is there a clear statement of the aims of the study?
2	What is the theory or context of research?
3	What research questions are asked?
4	What kind of paper is it? (Problem identification and/or problem solution or Experience Paper, Opinion Survey or Discussion paper)
5	What research method was used: Experiment, Quasi-Experiment, Lessons learnt, Case study, Opinion Survey, Tertiary Study, Other (specify)?
6	Was the research method appropriate to address the aims of the research? Yes/Partly/No/ Not applicable (i.e. Expert Opinion)
7	Was experimental material or context (for lessons learnt) appropriate to the aims of the research? Yes/Partly/No/Not applicable (i.e. Expert Opinion)
8	For empirical studies (apart from Lessons Learnt), was the data collected in a way that addressed the research issue? Yes/Partly/No/Not applicable (i.e. Lessons learnt or Expert opinion).
9	For empirical studies (apart from Lessons Learnt), was the data analysis sufficiently rigorous? Yes/Partly/No/Not applicable

[1] The reason for the unavailability of the paper may be because it is simply not published through any of the leading and most widely read outlets (including posting by the researcher themselves online). It should be noted that researchers did have access to all leading research databases for this study.

10	Is there a clear statement of findings? Yes/Partly/No
11	Are there any validity threats and limitations presented? Yes/Partly/No
12	Is the study of value for research or practice? Yes/Partly/No

The given quality criteria will be applied thoroughly during the snowballing procedure including the selection of the baseline set of papers and subsequent iterations of SB.

3.3 Iterations in Snowballing

Iterations in snowballing include backward snowballing and forward snowballing described in this section.

3.3.1 Backward Snowballing

In backward snowballing (BSB), the references from each paper in the baseline set will be reviewed based upon the relevance to the topic and the inclusion and exclusion criteria will be applied in two different phases similar to the one applied to the baseline set of papers. The only difference will be in inclusion and exclusion criteria for the first phase. The following criteria will be applied in the first phase of BSB (Badampudi *et al.* 2015):

- Read the title of the referenced paper
- Read the context in which the reference appears in the paper
- Review the abstract of the referenced paper

It is important to understand the context in which the reference is cited within the text of the paper. The referenced paper in full will be assessed based on the criteria of inclusion and exclusion, and quality assessment given in section 3.2.

3.3.2 Forward Snowballing

Identification of a new paper in forward snowballing (FSB) is based on the papers citing the paper under examination (Wohlin 2014). This information is available from the Google Scholar search engine. The inclusion and exclusion criteria is based on the following steps (Badampudi *et al.* 2015):

- Read the title of the citing paper
- Read the abstract of the paper citing the paper under examination
- Understand the context in which the paper is being cited

The final two steps for FSB are in reverse order compared to backward snowballing (Badampudi *et al.* 2015). Reading the abstract before the context of use in FSB will give the researcher clarity on the topic of the paper and enable him or her to understand the context in a better way. The newly identified papers at the end of an iteration in the snowballing procedure will be added to the final baseline set. The iterations will cease when no new papers are found.

4. Application of Literature Review Methodology

4.1 Identifying the Baseline Set

The queries in Table 1 were executed on Google Scholar using Zotero[2] (a free tool for managing bibliographies) to retain Google Scholar results generated by executing the queries and removing duplicates. Zotero further facilitated to download the citation details of the papers including electronic versions. Quotes around the term knowledge loss help to search for it on Google Scholar as one word.

During the execution of the search strings, three search strings returned results less than 30. For those search strings that returned more than 30 results on Google Scholar, only the top 30 results were selected. In all, 129 publications were extracted and after identifying 62 duplicates, 67 individual papers remained. Inclusion / exclusion criteria given in section 3.1 were applied on these 67 publications.

4.2 Final Baseline Set

The exclusion criteria were applied in two phases, the outcome of phase one resulted in 21 papers out of 67 papers. One replication study was found by Nassif and Robillard[3] based on earlier work in the paper (Rigby *et al.* 2016) which was not peer reviewed and used the same experimental method. According to the guidelines given when a study uses the same experimental method but different material, it is treated as muti-case case study, and it is believed to increase the scope and size of the study and not the quality (Kitchenham and Brereton 2013). The replication study found among 67 papers was excluded. Peer reviewed doctoral symposiums were not included in the initial baseline set, as these papers were short papers (1-4 pages) and tended to propose future work. In case it was not clear from the title of the paper to include or exclude it, the abstract, introduction and conclusion was read. A number of papers were excluded because they did not discuss knowledge loss or turnover in the context of OSS projects. Furthermore, papers with a focus on knowledge relevant issues in organisations other than in OSS projects were excluded, giving in all 21 papers at the end of phase one.

In phase two, the full text of 21 papers was evaluated based on the criteria to assess quality (section 3.2) and details were retained in an excel spreadsheet for future reference. One paper that is on knowledge transfer challenges and mitigation in Global Software Development (GSD) was included, since OSS is considered to be an example of GSD (Herraiz 2006; Lin *et al.* 2017). Five papers excluded focused on topics including knowledge gaps in post-merger integrations, managing knowledge and organisational costs, strategies to manage knowledge loss in organisations, a peer reviewed short paper on impact of mentoring with a research model, and on open innovation. The final baseline set consisted of 16 papers that discuss OSS in relation to knowledge loss or turnover. The selection of a final baseline set of papers is shown in Figure 3. In the next step, 16 papers were iterated as detailed in the following section.

4.3 Iterations

[2] Open-source reference management software http://www.zotero.org/

[3] Nassif, M. and Robillard, M.P. Revisiting Turnover-Induced Knowledge Loss in Software Projects.

4.3.1 Iteration 1 - Backward Snowballing and Forward Snowballing

During the first iteration, the first step in backward snowballing was to read titles of referenced papers in the baseline set. In all 643 titles referenced in the baseline set of papers were read for relevance to this work. As a result, 41 papers were identified for further evaluation. We found that 20 of the papers were already included in the baseline set or the first iteration of backward snowballing. Examination of the remaining 21 papers led to the exclusion of four papers leaving 17 papers to add to the collection of relevant literature.

We applied forward snowballing to 16 papers in the baseline set. The Google Scholar search engine was useful to find a total number of papers citing the corresponding paper in the baseline set. In all, 203 papers cited 16 papers in the baseline set. Only 24 papers were relevant to this work and 23 were already included in either the baseline set of papers or through the first iteration of backward snowballing. Iteration 1 resulted in 18 new papers from the initial 16 papers in the start set.

4.3.2 Iteration 2 - Backward and Forward Snowballing

The 18 papers resulting from iteration 1 were iterated by following the steps of BSB and then FSB. During BSB, 845 references appeared in 18 papers and after reading all titles and applying inclusion exclusion criteria on selected ones, only two new papers were included. We then checked Google Scholar to find out papers that cited 18 papers from Iteration 1. The papers that were cited in more than 100 publications, titles only for the first 100 papers were read. During FSB, 1174 paper titles were read and two new papers were found. Iteration 2 ended with four new papers.

4.3.3 Iteration 3 - Backward and Forward Snowballing

During iteration 3, four papers resulting from iteration 2 were iterated. In BSB, 252 titles were read for their relevance to this research and no new papers were found. In FSB, 226 papers cited four papers on Google Scholar. However, no new papers were found in iteration 3. Consequently, iteration 3 marked an end to the snowballing procedure. The snowballing phase ended with 38 papers, which are listed under primary studies in Appendix I. We summarise our application of the snowballing procedure resulting in 38 papers in Figure 5.

4.4 Data Synthesis

The three iterations from the application of the snowballing procedure yielded 38 papers. During iterations, the general information collected was about paper title, venue of publication, and year of publication. The specific information on papers is constituted upon 12 quality assessment criteria given in section 3.2.

Data synthesis was performed by collating and summarising the contents of the included 38 papers, similar to the guidelines provided for systematic literature review (Keele 2007). At first, the text of each study is analysed to derive individual themes. Secondly, the derived set of themes are analysed on their relationship with each other based on the inferences made from the study. The derived themes were added to the details of quality assessment. In addition to data synthesis, other details were also collected about the 38 papers.

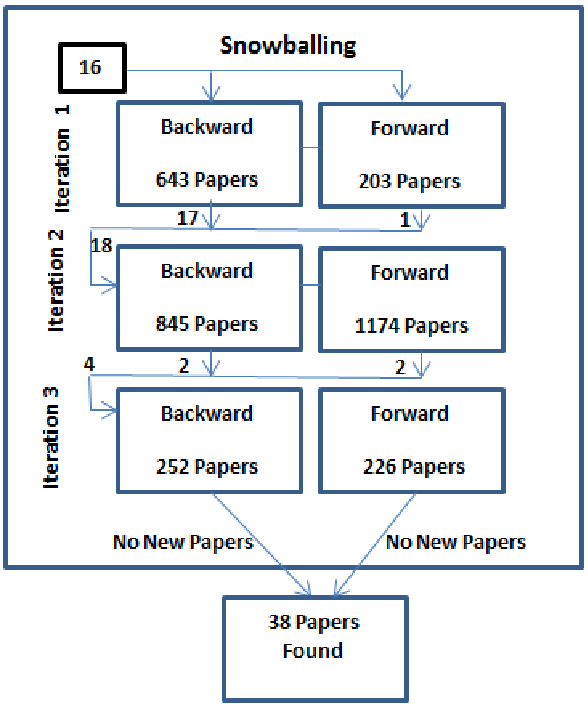

Figure 5: Application snowballing process to search relevant papers on knowledge loss in OSS

The papers selected are from year 2000 to year 2017, the year wise distribution is given in Figure 6. There are 18 conference papers, 17 journal papers, and 3 peer reviewed papers published in workshops. The distribution of papers based on their methodology is shown in Figure 7.

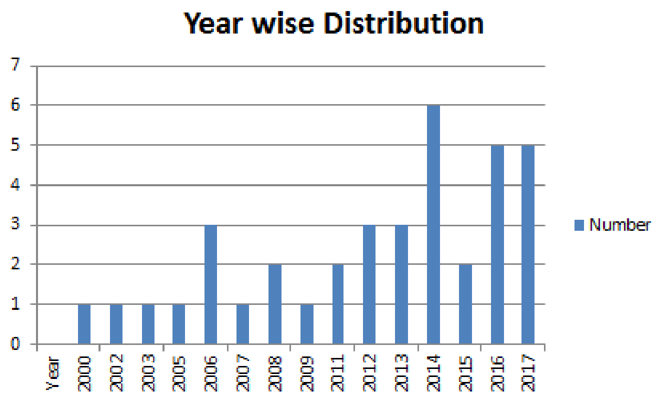

Figure 6: The yearly distribution of papers

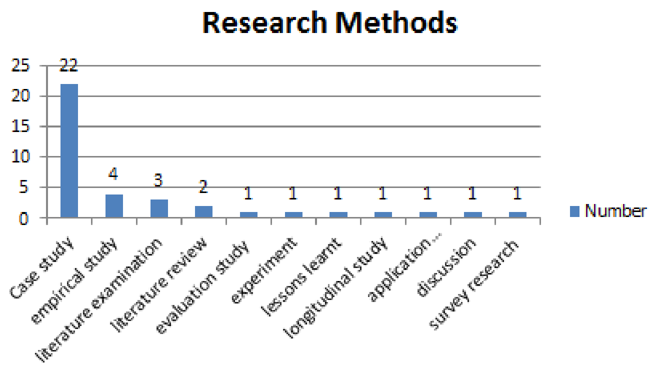

Figure 7: The distribution of papers according to research method

5. Discussion

The emerging themes identified from the papers in this work are subjective in nature and a rigorous process was followed to collect papers in a systematic literature review. In this section, we examine the problem of knowledge loss in OSS projects and the subsequent section lists the impact of knowledge loss in OSS projects.

5.1 Examining Knowledge Loss in Open Source Software

The evolution of an OSS project results in 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 to as turnover (Foucault *et al.* 2015). Some of the reasons for the contributors leaving OSS projects include loss of interest in the project, new career opportunities, no new learning or improvement of skills (Shah 2006). The author of the source code has a strong relationship with the authored code. When the author of the code leaves the project and their code is abandoned, software development can halt due to knowledge loss (Rigby *et al.* 2016). Knowledge loss is a problem that has been reported equally in CSS (Izquierdo-Cortazar *et al.* 2009; Jennex and Durcikova 2013; Viana *et al.* 2015; Zhou and Mockus 2015) and OSS (Izquierdo-Cortazar *et al.* 2009; Rigby *et al.* 2016) projects. The phenomenon of contributors joining and leaving at their discretion is more common in OSS projects than with hired employees in CSS (Robles *et al.* 2005).

A moderate amount of turnover is thought to bring innovation to the project (Ling *et al.* 2005). Similarly it is reported that in Wikipedia, an online community, moderate levels of membership turnover positively affect collaborative success (Ransbotham and Kane 2011). However, in OSS projects, contributor turnover is reported to have a negative impact on quality (Foucault *et al.* 2015) and productivity, as due to knowledge loss, extra time is spent to learn the workings of the project (Izquierdo-Cortazar *et al.* 2009).

In many large OSS projects, a high turnover has been observed leading to the formation of the succeeding development teams (Robles and Gonzalez-Barahona 2006). In order to continue with the software development tasks, succeeding development teams require knowledge about the developed source code. Constant evolution requires that new contributors be knowledgeable enough to perform maintenance tasks on the project during the absence of the earlier owner of the system (Rigby *et al.* 2016).

A study on the GNOME[4] project reported that 30 months' time is needed for the contributor to understand the software code and to make a contribution (Herraiz 2006). A lack of understanding of the design of abandoned code (Mockus 2010) slows down the contributors and impacts their productivity (Schilling *et al.* 2011). Searching knowledge is argued to be time consuming and costly (von Krogh *et al.* 2005). The search efforts can vary depending on the source and the level of details. For instance, a post or a query on the project mailing list requires less effort, while searching through the results of search engine use or examining the clues into source code documentation is time consuming (von Krogh *et al.* 2005).

Accumulated knowledge, which is not codified, is lost once the contributors leave the project. Contributors gain skills, experience while working on software projects, and take the knowledge with them once they leave the project. In this work knowledge, loss refers to the loss of accumulated knowledge that is not available or is too costly to reacquire once the contributor leaves the OSS project.

5.2 Impact of Knowledge Loss in OSS projects

In this section, we unfold the effects of the knowledge loss phenomenon on OSS projects as consolidated by synthesis of the data collected from the relevant literature.

A) Abandoned code: When a contributor who has owned or worked on code files leaves the OSS project, the files or Lines of Code (LOC) are orphaned or abandoned (Izquierdo-Cortazar *et al.* 2009). Abandoned code characterises the situation of knowledge loss in terms of efforts lost that affect the productivity of the OSS project. It is emphasised that with the increase of orphaned lines of code, the project may become invisible to the current contributors, since the major parts of the code is written by the contributors who are no longer available (Izquierdo-Cortazar *et al.* 2009). Researchers have quantified knowledge loss based on the number of lines of code and files that become orphaned or abandoned when a contributor leaves (Izquierdo-Cortazar *et al.* 2009; Rigby *et al.* 2016).

During the preparation of a release, contributors make changes to align their work with the goals of the release (Michlmayr 2007). A central person or body then selects a subset of the developed code for the "official" releases in Open source developments and makes it widely available for distribution (Mockus *et al.* 2002). At the time of release, the unmaintained source code is removed from the project since the original contributors who maintained it are not there anymore. The code has an element of uncertainty for the development team since the contributors who wrote it have left (Izquierdo-Cortazar *et al.* 2009). Removal of unmaintained code results in loss of existing functionality and may impact users of the system (Michlmayr 2007). It is reported that on average, 12% of Source Line of Code (SLOC) is abandoned (Otte

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

et al. 2008). As abandoned code increases on the project, the numbers of reported defects have been shown to increase as well (Otte *et al.* 2008). The maintenance of abandoned code is difficult because the team lacks knowledge of its creation and structure (Donadelli 2015).

B) Project Instability: The stability of the OSS project and its success is dependent on the ability to sustain contributors (Schilling *et al.* 2012). A strong combination of experience and knowledge is associated with contributor retention (Schilling *et al.* 2012). Contributors who work on modifying files by others have a better chance of survival (Lin *et al.* 2017), since it diversifies their experience and knowledge about the work of others. There is a focus on identifying newcomers at an earlier stage of joining an OSS project, with a higher chance of contributors staying for the long term by being trained by the current contributors (Schilling *et al.* 2012). In another study for the purpose of decision making and observation, the assessment of community stability is based on the estimations of the future participation (Ayushi and Ashish 2014). Lack of sustained contributors lead to the failure of 80% of OSS initiatives (Colazo and Fang 2009).

C) Discontinued Knowledge Building: Knowledge building results from collaboration and the use of a variety of tools and resources including Internet Relay Chat (IRC), Concurrent Version System (CVS), Bug Report, Feature request, Patches, Tasks and News (Ge *et al.* 2006). From the initiation of the problem to resolution, contributors with different expertise and knowledge contribute to knowledge building in the community. Knowledge in collaborative online communities is not concentrated with an individual but rather results from the collaboration of contributors. Community members who leave not only take their knowledge and expertise but also the established relationship with other members (Wang and Lantzy 2011). The knowledge building process may discontinue due to disruption in social ties.

D) Community Health Chaos: The driving force of OSS projects is their communities consisting of adhoc or informal, foundation or non-profit and commercial companies (Zheng *et al.* 2008). Community health is affected by turnover due to loss of human capital and social capital (Wang and Lantzy 2011). Consequently, with human capital the long-term knowledge, which is tacit in nature, is lost. Furthermore, in the case of social capital, the developed relationship among community members is disrupted. Social capital has resources embedded in social relationships with inherent norms and values (Droege and Hoobler 2003). Furthermore, social structures have a central role in knowledge creation (Nonaka 1994). For members who are central in the structure of the community, their departure will have a higher impact on the community health (Wang and Lantzy 2011). For instance, membership turnover in Wikipedia, has a significant negative effect on group outcomes, and there exists an association between social capital losses through member turnover and group (Wang and Lantzy 2011). Moreover, member turnover in Wikipedia was associated negatively with the group productivity (Qin *et al.* 2014).

E) Damage to the Ecosystem: In a software ecosystem, "collections of software projects are developed and evolve together in the same environment" (Lungu 2008). Disruption of an ecosystem not only has an impact on the technical aspects (such as its package dependencies), but also on social aspects (Mens 2016). Ecosystems evolve over time "through socio-technical changes that may greatly impact the ecosystem's sustainability. Social changes like developer turnover may lead to technical degradation" (Constantinou and Mens 2017). There is a high probability that contributors will abandon an ecosystem if contributors do not interact in discussion with others (Constantinou and Mens 2017). It is reported that when a contributor with a central position in the community unexpectedly left, it was a cause of major disruption in the community (Mens 2016). More specialised or central leavers cause risk to the functioning of an ecosystem (Lin *et al.* 2017). For example, a package containing only a few lines of source code but with thousands of dependent projects, when removed, had important consequences "almost breaking the internet" (Mens 2016).

F) Evolutionary Pressure: When contributors withdraw from a project and new ones join, the transition influences the evolution of the contributor's network and generates evolutionary pressure (Joblin *et al.* 2017). Evolutionary pressures are due to the difference in turnover rates between core and peripheral developers (Joblin *et al.* 2017). Evolution of the system will require the new contributor to have knowledge on the project to perform maintenance tasks, in the absence of the owning developers who owned the system earlier. In the absence of a contributor, original owner of the files or system and a successor to files with related knowledge on the project to perform maintenance tasks, the files are at a risk of abandonment (Rigby *et al.* 2016).

G) Non-uniform Knowledge Distribution: A small subset of contributors, typically 20%, called core members, make major code contributions of about 80% in OSS projects (Mockus *et al.* 2002). Knowledge distribution in OSS projects among contributors is found to be non-uniform, one developer leaving can cause a major file loss (removal of file ownership leads to its abandonment) in the system (Rigby *et al.* 2016). For example, on the Gimp project, where relevant project knowledge is limited to a small set of contributors, one contributor leaving results in the loss of up to 80% of files in the system (Donadelli 2015). On the contrary, when knowledge is distributed across a larger group of contributors, file loss is minimised when one contributor leaves, as seen in the case of Linux project (Donadelli 2015).

H) Knowledge Differentiation: Knowledge differentiation refers to the extent of specialisation of team members in different knowledge domains (Chen *et al.* 2013). Knowledge differentiation in CSS teams shows that knowledge differentiation is specifically important for software development and it involves integrating knowledge from various domains, such as software architecture, software design methodologies, and business application domain knowledge (Tiwana 2004). Although OSS teams are different from CSS teams in various ways, the baseline for the knowledge intensive nature of software development does not change for CSS or OSS contributors. In the case of Freenet, a file sharing OSS project, it was found that the majority of contributors specialize in coding few and not all modules, because of existing knowledge barriers between the modules (von Krogh *et al.* 2003). Coding specialization exhibits knowledge differentiation in OSS teams (Chen *et al.* 2013). The knowledge differentiation concept intertwines with knowledge distribution and can have consequences in OSS projects when the contributors leave.

The impact of contributor turnover has been realised in the software engineering community and it has been asserted that most existing studies are conducted in the open source communities as compared to non-open source. The data collection is mainly from software project management and obtained by performing qualitative observations and quantitative analysis of contributor activity (Bao *et al.* 2017).

In this section, we discussed the main impacts of knowledge loss in OSS projects. The impact of knowledge loss on quality and productivity was discussed in section 5.1. Similarly, the role of knowledge in relation to maintenance, reinvention, and evolution was discussed in section 2.1, which we illustrate through the mind map in Figure 8.

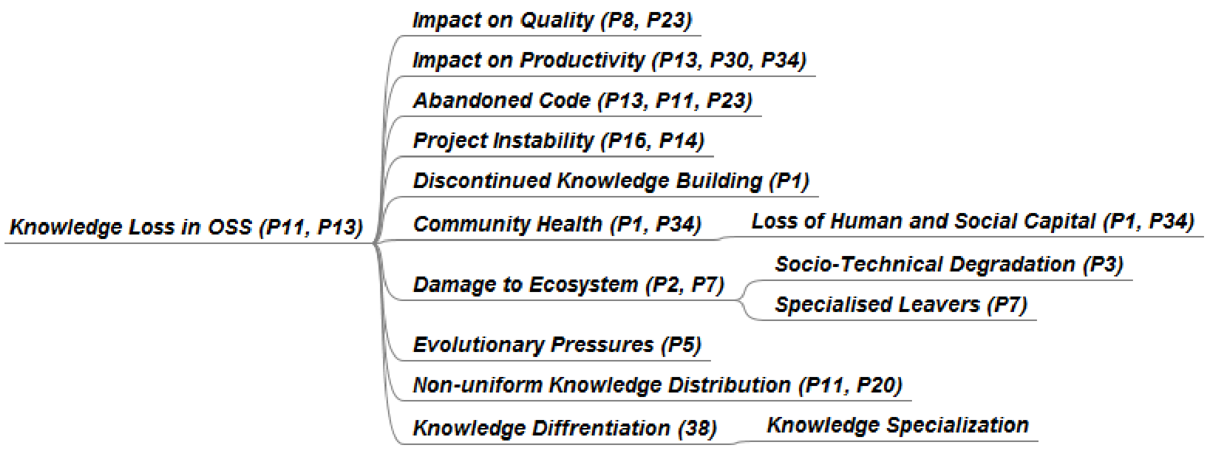
Knowledge Loss in OSS (P11, P13)
Impact on Quality (P8, P23)
Impact on Productivity (P13, P30, P34)
Abandoned Code (P13, P11, P23)
Project Instability (P16, P14)
Discontinued Knowledge Building (P1)
Community Health (P1, P34)
Loss of Human and Social Capital (P1, P34)
Damage to Ecosystem (P2, P7)
Socio-Technical Degradation (P3)
Specialised Leavers (P7)
Evolutionary Pressures (P5)
Non-uniform Knowledge Distribution (P11, P20)
Knowledge Diffrentiation (38)
Knowledge Specialization

Figure 8: Depicting impact of knowledge loss in OSS projects

6. Reducing Knowledge Loss in OSS Projects

The knowledge loss phenomenon and its impact in OSS projects were discussed in section 5. In this section, our focus is on the literature that discusses the reducing knowledge loss in OSS projects due to contributor turnover. We first draw our attention to Knowledge Retention (KR) in traditional organisations, which mainly comes into focus when an employee is leaving (Lindvall and Rus 2003). The need for a KR mechanism in an organisation is assessed based on the following (Lindvall and Rus 2003):

1. A lack of knowledge in the organisation and an overly long time to acquire knowledge due to a steep learning curve
2. Employees repeating mistakes and performing rework because they forget what they learned from previous projects.
3. Chances of employees owning key knowledge becoming unavailable

Knowledge Retention relates to capturing knowledge in an organisation and is an important aspect of Knowledge Management (KM). KM is defined as the approach adopted by an organisation to engage workers in relevant activities of creating, managing, sharing and reusing knowledge (Dingsoyr *et al.* 2009). KM also refers to coordination and exploitation of knowledge in an organisation and making the right knowledge available to the right people while benefiting from it for competitive advantages (Drucker 1999). KM enables an organisation to enhance its capabilities to deal with new and unusual situations (Choi *et al.*

2008). Knowledge retention can be seen as a way of embedding and enabling knowledge within an organisation and a critical factor for sustainable performance (Doan *et al.* 2011).

The KR strategies in an organisation advance innovation, organisational growth, efficiency, employee development, and competitive advantage. Sometimes techniques such as exit interviews are used to capture the knowledge of the employee. Some organisations also follow formal knowledge retention strategies. 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 as well. 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 managing knowledge intensive activities like software development (Donnellan *et al.* 2005). Codification is the documentation of the knowledge to be stored, disseminated, and reused. Personalisation is the development of networks to connect people where they can share tacit knowledge (Hansen *et al.* 1999). In knowledge bases, codification captures electronic information and personalisation deals with the ways humans use and process knowledge (Donnellan *et al.* 2005). Organizations implement codification strategy to encourage the reuse of explicit knowledge.

Table 3: Representation of KR techniques with knowledge conversions type and codification strategy

Retention Techniques	**Knowledge Conversion Type**	**Managing Strategy**
Community of Practice	Tacit to Tacit	Personalisation
Post-mortem	Tacit to Tacit	Personalisation
Mentoring	Tacit to Explicit	Codification
Storytelling	Tacit to Explicit	Codification
Experienced Based Memory	Tacit to Explicit	Codification
Interviews	Tacit to Explicit	Codification
Training	Explicit to Tacit	Personalisation

The core techniques designed to retain knowledge in an organisation are mainly dependent on its knowledge-sharing practices. There are many techniques that facilitate knowledge capture, sharing and reapplication, namely 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 (De Long and Davenport 2003).

In Table 3, KR techniques practiced in organisations with the type of knowledge conversion and managing strategy are given. Community of Practice[5] are used as a long term strategy by management to retain knowledge before it is lost (De Long and Davenport 2003), where experienced members deal with the problem of losing expertise by contributing valuable lessons to member companies. Post-mortem reviews are a practical method to capture and reuse experience from projects that are complete or have finished a major activity phase (Dingsøyr 2005). Mentoring is suggested to be a logical approach for transferring important tacit and

[5] A community of practice is a group of people who share a concern or a passion for something they do, and learn how to do it better as they interact regularly

implicit knowledge (De Long and Davenport 2003). Story telling can play a valuable role in transferring knowledge to convert tacit knowledge into explicit knowledge (De Long and Davenport 2003). Software engineers in their daily work environment, create projects and product memory, which are based on their experience (Rus and Lindvall 2002). Experienced based memory is built on software engineering practices and product memories, which are an indirect or direct effect of software development. Some examples include version control systems, change management, documentation design decisions, and requirements traceability. Interviews in an organisation is a knowledge transfer process used to integrate the knowledge captured into the organisation (De Long and Davenport 2003). Training is another knowledge transfer practice found to include some combination of formal classroom training, eLearning, video or computer-based training, on-the-job training, coaching and shadowing (De Long and Davenport 2003).

We briefly discussed KR practices that are used in traditional organisations to overcome knowledge loss due to leaving employees. In contrast to traditional organisations, KR is not formally practiced in OSS projects. The research community recognises Knowledge Management (KM) in OSS development as challenging because of its highly distributed, dynamic work structure, and knowledge-intensive characteristics (Ciborra and Andreu 2001; Crowston *et al.* 2012) . KM is one of the social processes and is of potential importance to manage the knowledge necessary for a successful development effort in OSS projects (Crowston *et al.* 2012). “Social processes capture cognitive, verbal, and behavioural activities performed by team members to manage interpersonal relationships among them” (Marks *et al.* 2001).

The research on knowledge management in OSS projects has focused on knowledge sharing, knowledge reuse, cross boundary learning in open communities, knowledge sharing involving strategic interaction, learning theory on knowledge creation and sharing in online communities, learning driven by criticism and error correction and a social view of learning that have overcome problem of tacit knowledge transformation. The details are summarised in Table 4 including the name of the KM related activity, description of study and references.

Further research is required to explore suitable KR practices applicable in OSS projects as indicated by one of the questions raised on mechanisms and team norms that are used to store knowledge contributed by team members. In CSS organisations KR mainly comes into focus when an employee is leaving an organisation (Lindvall and Rus 2003). On the contrary, in OSS projects the unpredictable nature of commitment from contributors creates an element of risk (Robles and Gonzalez-Barahona 2006). In OSS, contributors can leave since they are not under any contractual binding as in CSS organisations.

Table 4: Summary of Research focus on KM relevant activities in OSS (Crowston *et al.* 2012)

KM Activities	**Description**	**Ref.**
knowledge Reuse	Knowledge shared or reused in OSS development	(Lakhani and Von Hippel 2003; Lee and Cole 2003; Hemetsberger and Reinhardt 2004; Dafermos 2005;

		Huysman and Lin 2005; von Krogh *et al.* 2005; Singh *et al.* 2006)
Cross Boundary	online communities without strict membership requirements activate cross-boundary learning and knowledge sharing	(Huysman and Lin 2005)
Knowledge Sharing and Strategic Interaction	Knowledge sharing and strategic interaction	(Kuk 2006)
Learning Theory on Knowledge Creation and sharing	Learning theory provides an understanding of online knowledge creation and sharing	(Lee and Cole 2003)
A Social View of Learning	A social view on learning and knowledge creation and solution to the problem of tacit knowledge transformation through technological tools, task-related features, collective reflection, stories, and usage scenario	(Hemetsberger and Reinhardt 2004)

In this literature review, we found that KM relevant activities of knowledge creation and knowledge sharing are evident in OSS projects as discussed in section 6.1. Furthermore, literature examination directed us to 10 mitigations to reduce the impact of knowledge loss due to contributor turnover in OSS projects discussed in section 6.2.

6.1 Manifestation of Knowledge in OSS Projects

OSS projects are considered an example of GSD (Herraiz 2006). OSS contributors are scattered globally while working together on projects. They collaborate on tasks through asynchronous technology mediated channels and utilise them to acquire and share knowledge. Earlier we defined knowledge for this work "as experience and expertise built by contributors that evolves from the day-to-day interaction on the OSS project." The manifestation of knowledge in OSS projects is experienced in operations of OSS communities. The knowledge manifestation in OSS projects can be explained by three categories consisting of an object of knowledge development, a process of knowledge development, and a location of knowledge (Venzin *et al.* 1998). Accordingly, the object of knowledge refers to procedural knowledge and knowledge of events including trends within and outside the organisation. Further, the knowledge development process refers to either cognitive abilities or knowledge construction process. The knowledge construction process includes knowledge creation, sharing, transfer, and its application. Finally, location of knowledge refers to carriers of knowledge, who can be individuals, groups, organisations, inter-organisations, and customers. Principally the location of knowledge is an indication towards tacit knowledge, which requires physical presence. Availability of tacit knowledge is dependent on its transfer and encoding before the contributor leaves the organisation. Contributors who participate in OSS projects with various forms of contributions (Rahardjo Emanuel 2014) have project relevant tacit knowledge and are responsible for its transfer and encoding before they leave the project.

We did not find any specific details on Knowledge Transfer (KT) in OSS projects. Since OSS projects are considered extreme case of GSD, we consulted the paper found earlier through the snowballing process that investigated the mitigation strategy on KT in GSD settings (Nidhra *et al.* 2013). The paper enlists mitigation strategies based on personnel, project and technology factors. We found some mitigation strategies similar to the ones listed in section 6.2. In order to resolve personnel challenges mitigation strategies applied are mentoring and shadowing, proactive learning and peer-to-peer help. In addition, project relevant challenges are mitigated by acquiring knowledge from community of practice and by maintaining documents and process. Finally, technological challenges are overcome by using document management systems, e-mails, wikis, instant messaging, configuration management system web-based mentoring, and access to a knowledge repository.

In the following sections, we discuss knowledge creation and knowledge sharing as part of the knowledge construction process in OSS projects. We refer to the literature found in this work that elaborates on knowledge creation and knowledge sharing in OSS projects.

6.1.1 Knowledge Creation

Knowledge creation takes place when individuals are collectively working and interacting on a task and are constantly acquiring relevant knowledge. Nonaka et al. (Nonaka *et al.* 2000), explain the knowledge creation process involving conversion of tacit knowledge to explicit knowledge which is then “crystalized”. 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. New tacit knowledge can be in the form of shared mental models or technical skill (Juntunen and Raisanen 2015). **Externalisation** is the conversion of tacit knowledge to explicit knowledge. Externalisation results in 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 resulting in tacit knowledge. The process of knowledge creation as detailed by SECI, can be used to explore knowledge creation in OSS projects (Rashid *et al.* 2017).

Further, two dimensions of knowledge are said to be widely used namely tacit vs. explicit and individual vs. collective (Nidhra *et al.* 2013). Individuals create individual knowledge and it exists with them, while collective actions of group leads to the creation of collective knowledge (Nonaka 1994). Furthermore, founded on the combinations of the two dimensions of knowledge, four types of knowledge are proposed (Lam 2000):

- Embrained Knowledge: Individual - Explicit (e.g. theoretical knowledge)
- Embodied Knowledge: Individual - Tacit (e.g. practical experience)
- Encoded Knowledge: Collective - Explicit (e.g. written rules, procedures)
- Embedded Knowledge: Collective - Tacit (e.g. routines, norms)

Knowledge creation in OSS projects differs from the CSS organisation, as highlighted through five organising principles of community based model: intellectual property ownership, membership restrictions, authority and incentives, knowledge distribution across organisational and geographical boundaries and dominant mode of communication (Lee and Cole 2003). In CSS organisations, the knowledge is owned by the organisation with access given to employees, the knowledge distribution is within the boundaries of the firm and mostly with face-to-face communication. On the contrary, in OSS projects, the knowledge contributions and sharing is open without any membership constraints, distribution of knowledge extends outside the community, and interaction of contributors is on a larger scale than in CSS organisations. In OSS projects, knowledge creation is through social interaction among individuals and organisations, and it is dynamic in nature (Nonaka *et al.* 2000).

6.1.2 Knowledge Sharing

In OSS projects, knowledge sharing is an ongoing activity in an intensely people-oriented and self-organised community (Sowe *et al.* 2008). Knowledge sharing is declared as a cognitive task and an OSS team is considered an example of a complex cognitive system (Chen *et al.* 2013). Knowledge sharing is through asynchronous means of communication and with a collection of artifacts, which are publicly available for reuse (Rashid *et al.* 2017). OSS contributors demonstrate their technical competence by actively participating in mailing-list discussions, reporting bugs and submitting code (von Krogh *et al.* 2003). In a longitudinal study, it was established that core contributors' knowledge sharing behaviours brings a high level of skills and understanding to the project teams (Licorish and MacDonell 2014). Effective knowledge sharing improves the productivity of the project (Levy and Hazzan 2009).

Measures of knowledge sharing activities identified with knowledge posting and viewing are indicators of the good state of health in virtual communities (Koh and Kim 2004). Utilising a similar concept of knowledge sharing activities such as posting and viewing, the Knowledge Sharing Model (KSM) (Figure 9) is developed, by analysing e-mail exchanges in Debian projects among a list of participants (Sowe *et al.* 2008). The KSM model follows a constructivist approach, which stresses the active and autonomous role of the learner (Sowe *et al.* 2005). Knowledge is constructed based on the learner's interaction with the learning environment while prior knowledge impacts the learning process (Sowe *et al.* 2005).

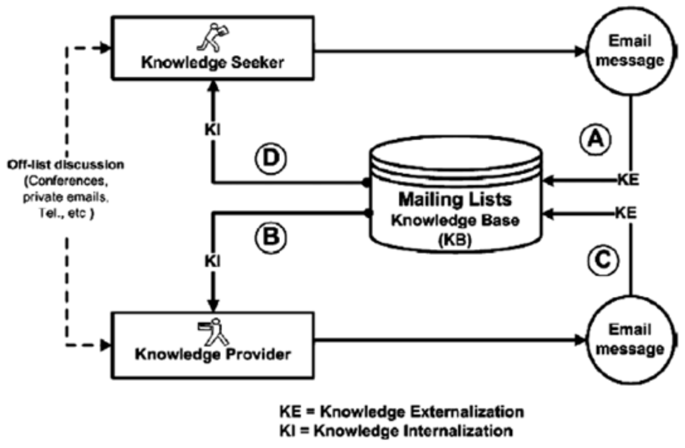
Knowledge Seeker
Email message
Off-list discussion (Conferences, private emails, Tel., etc)
KI
D
A
KE
Mailing Lists Knowledge Base (KB)
KE
B
C
KI
Knowledge Provider
Email message
KE = Knowledge Externalization
KI = Knowledge Internalization

Figure 9: Knowledge Sharing Model (KSM) (Sowe *et al.* 2008)

The model uses a simple measure of two values consisting of e-mail messages posted by knowledge seekers and the number of replies posted by knowledge providers. The analyses of knowledge sharing trends and information contained in e-mails can be utilised to identify what kind of knowledge exchanges are happening in the contributors' community. Moreover, the number of contributions made on the project can determine the expertise level of the group responding to knowledge seekers.

A study on the schema of information types sought in OSS mailing lists asserted that mailing lists are a strong representative of communication in open source software and offer an insight into information seeking needs (Sharif *et al.* 2015). The findings suggest that 42% of information sought on mailing lists is on understanding task implementation and understanding bugs. Mailing lists are a primary source of communication in OSS where knowledge sharing is abundant (Sowe *et al.* 2008). Programmers never meet face to face but coordinate all their activities (Mockus *et al.* 2002). In an OSS project, a mailing list is reported to have unique characteristics of Transactive Memory System (TMS) (Ryan and O'Connor 2012), a multidimensional construct including knowledge differentiation, knowledge location and knowledge credibility (Chen *et al.* 2013). Knowledge differentiation refers to the extent of specialisation in different knowledge domains, knowledge location is about who knows what, and knowledge credibility is assessed by the task performance of individual developers.

Gamification (Yilmaz *et al.* 2016) is another emerging form of knowledge sharing in OSS communities (Vasilescu *et al.* 2014). The community members vote upon the questions and answers posted on a site, the numbers of votes reflect the poster's reputation and a measure of his or her expertise by the potential employer. A gamification element on sites is found to have increased the engagement of the participants and popularity of the site. In OSS communities, gamification is argued to provide a better visibility of contributors activities (Vasilescu *et al.* 2014).

The social media sites also serve for contributors to learn, collaborate, share knowledge and interact with users of software (Vasilescu *et al.* 2014). Contributors contribute on software development sites such as GitHub for coding, Jira to track issues, StackExchange network for open query submission and response, StackOverflow for professional programmers and CrossValidated for statisticians and data miners. Blogs are also considered an effective way of sharing knowledge due to easier access to the internet (Hsu and Lin 2008). Knowledge is also stored in repositories namely: Concurrent Versions System (CVS), Subversion (SVN), Frequently Asked Questions (FAQs), project websites, blogs, bug reporting and bug tracking databases (e.g. Bug Tracking System BTS) and mailing lists (Sowe *et al.* 2008). In Figure 10, we summarise primary studies in a mind map on literature that discusses knowledge creation and knowledge sharing in OSS projects.

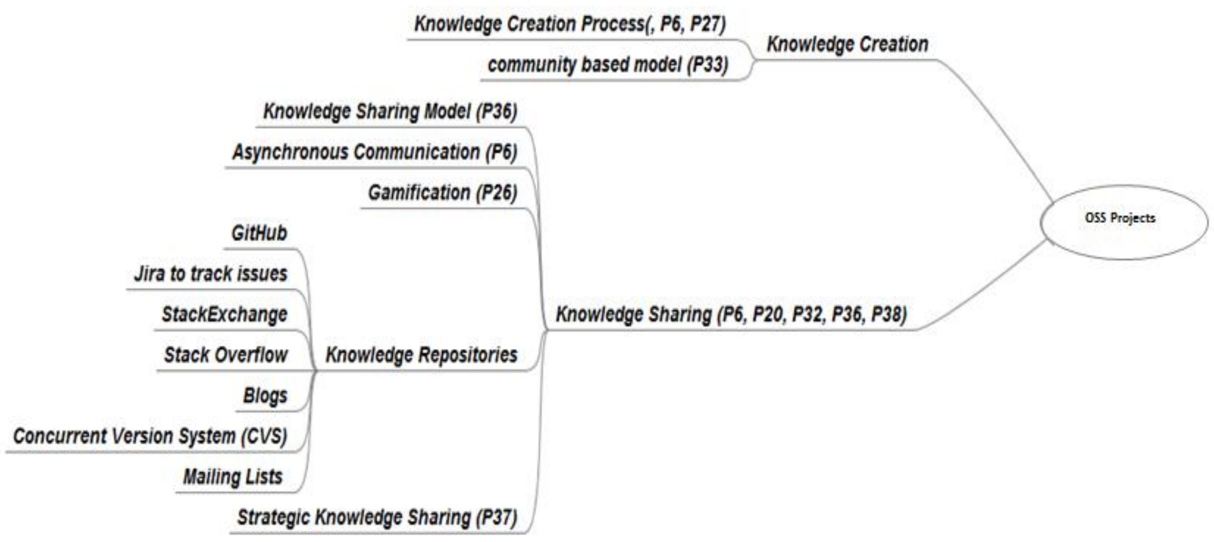
Knowledge Creation Process(, P6, P27)
community based model (P33)
Knowledge Creation
Knowledge Sharing Model (P36)
Asynchronous Communication (P6)
Gamification (P26)
GitHub
Jira to track issues
StackExchange
Stack Overflow
Knowledge Repositories
Blogs
Concurrent Version System (CVS)
Mailing Lists
Strategic Knowledge Sharing (P37)
Knowledge Sharing (P6, P20, P32, P36, P38)
OSS Projects

Figure 10: Mind map of the Knowledge Creation and Sharing in OSS projects

6.2 Knowledge Retention in OSS Projects

The examination of selected literature on knowledge loss in OSS projects resulted in themes including mitigation of knowledge loss in OSS projects. A body of researchers from the software engineering community have focused on the impact of turnover in OSS projects and have highlighted practices to mitigate its effects. We discuss the main practices in this section.

A) Visualisation of Resources: A visualization word cloud has been proposed to show quickly the level of cooperation of the team in the project (Fronza *et al.* 2013). A large variety of data collection is without human intervention and rendered as a wordle. Intensity of colour and size of the letter in a wordle indicate a need for resources. A Visualization word cloud does not cause overhead on the productivity of the contributors.

B) Pair Programming and Shared Code Ownership: In order to mitigate the effects of turnover on the ecosystem, the usage of techniques such as pair programming and shared code ownership are suggested (Mens 2016).

C) Successor: A successor is a person who has relevant expertise and is knowledgeable on the work of other contributors. Identification of successors and involving them as co-owners is presented as a method to reduce the risk associated with developer turnover (Rigby *et al.* 2016). The files with a successor were not at risk of abandonment even when the owning developer left. A successor was there to perform maintenance tasks (Rigby *et al.* 2016).

D) Centralisation: Governance structures in Ericsson is argued to be similar to OSS, and a centralised approach is implemented to secure quality (Britto *et al.* 2016). The situation of knowledge loss faced was because of the recurrent movement of resources in and out of products and constantly changing business needs. In such a situation, the adoption of a centralised approach helped in architectural knowledge stability and its availability to new developers and teams.

E) Removal of Knowledge Barriers: Only a few OSS contributors transit to a higher learning state, due to high learning barriers. Consequently, it can take newcomers up to 60 weeks to become an effective contributor to a OSS project (Adams *et al.* 2009). Knowledge retention in OSS projects can also be improved by the removal of knowledge barriers: namely, lack of technical experience, lack of domain expertise and lack of project practices that hinder the contributions (Steinmacher *et al.* 2015a; Steinmacher *et al.* 2015b).

F) Knowledge Map and TMS: An insight into the area of expertise members, a knowledge map or directory can be used on the project website (Chen *et al.* 2013). Organizations such as IBM have used such a directory as their internal knowledge portal. In order to leverage OSS project teams, a focus is required towards facilitating the TMS development within the teams, based on knowledge location, the usage of the developer mailing list and knowledge credibility (Chen *et al.* 2013). These dimensions are reported to have positive effects on communication quality, improving team performance and reduce the impact of turnover.

G) Diversity of Core Contributors: For an OSS project to survive, a diversity of core developers is required (Wahyudin *et al.* 2007). When a key contributor abandoned an OSS project, it revealed a very fluctuating proportion of developer contribution. A significant imbalance between the contribution and the response from the developers' community was noticed. The reason for the dying project was that a diversity of core contributors were missing from the project (Wahyudin *et al.* 2007). Diversity of core contributors also relates to the underlying concept of uniform knowledge distribution stated next.

H) Uniform Knowledge Distribution: The communication of the OSS project is directed by the core contributors. Their attitudes and involvement in knowledge sharing were linked to the demands of their wider project teams (Licorish and MacDonell 2014). The core contributors bring high levels of skills and cognitive characteristics to their project teams. They start the project and provide high levels of ideas, suggestions, information, comments, instructions and answers to their teams, and are the centre of their project's knowledge activities. The more code changes, core developers perform, the more knowledge they provide (Licorish and MacDonell 2014). However, their least involvement in communication and task changes results into some negative team attitudes. This kind of disruption in communication in OSS projects can hinder knowledge sharing. Another resolution to the non-uniform distribution of knowledge may be the proactive assignment of maintenance tasks on the code written by other contributors. As indicated that contributors who modify codes from other contributors stay longer on the project (Lin *et al.* 2017). This will create a balance of equal development of skills on the OSS project. For example, contributors who normally perform documentation tasks should be assigned some coding tasks.

I) Gamification: A gamified environment has important implications for knowledge management in software engineering (Vasilescu *et al.* 2014) and OSS projects. As observed, Q&A gamification increased the engagement of knowledge providers and the quickness of response. This finding suggests that Q&A site designers should consider gamification elements to increase contributor engagement, which indirectly can help to raise the popularity of their sites. For example, gamification features in Stack Overflow's guarantee that a question will be replied to by enthusiastic experts within minutes of being posted (Zagalsky *et al.* 2016). On Stack Overflow, a crowd approach is used where participants contribute knowledge independently of each other and gamification qualities are used to evaluate who provides the best answer, gains the most points (Zagalsky *et al.* 2016). Knowledge is curated in gamification other than being developed, as is the case with mailing lists. Curation is a mechanism to provide a tool for keeping the channel clear of what seems to be unnecessary information (Zagalsky *et al.* 2016).

J) Improving Code Review Feedback Time for non-Cores: Peer reviews are conducted asynchronously in OSS projects to empower experts who provide feedback to code contributors (Rigby *et al.* 2014). As indicated by a social network analysis of the code review data from eight popular OSS projects, core developers as compared to peripheral contributors have the benefit of receiving quicker feedback, face shorter review intervals and have a higher code acceptance rate (Bosu and Carver 2014). Due to the lack of an established reputation, peripheral developers wait 2 to 19 times (or 12 to 96 hours) longer than core developers to complete the review process. Accordingly, a delay in receiving feedback on reviews may negatively motivate a peripheral or new contributor (Bosu and Carver 2014). An improvement to the timings of the review feedbacks in OSS projects to peripherals can result in a uniform distribution of knowledge, reduce the effects of turnover, and motivate newcomers to stay for a longer duration.

In this section, we explained manifestation of knowledge in OSS projects using the knowledge construction process mentioned in section 6.1. We found that in OSS projects, knowledge creation, and knowledge sharing is abundant through asynchronous communication. In section 6.2, we discussed 10 mitigation techniques to reduce knowledge loss in OSS projects visualised as a mindmap in Figure 11.

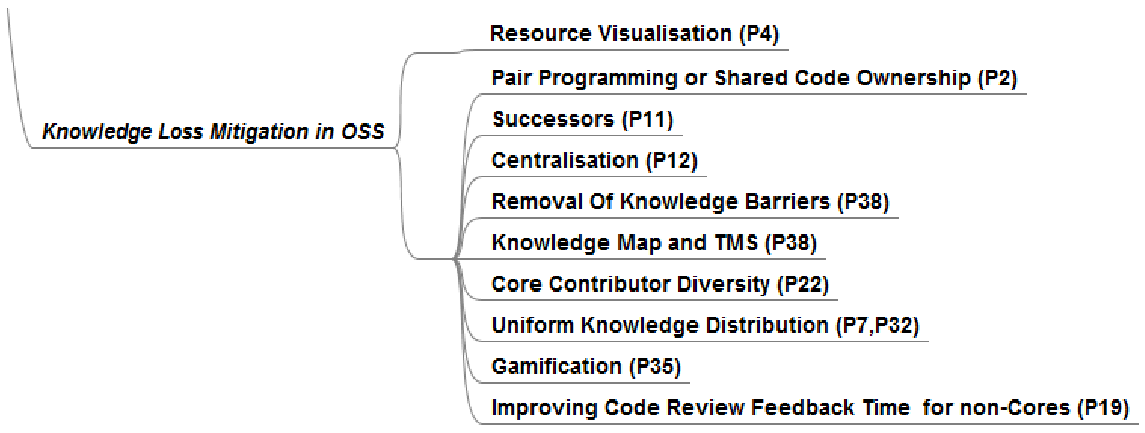
Knowledge Loss Mitigation in OSS
Resource Visualisation (P4)
Pair Programming or Shared Code Ownership (P2)
Successors (P11)
Centralisation (P12)
Removal Of Knowledge Barriers (P38)
Knowledge Map and TMS (P38)
Core Contributor Diversity (P22)
Uniform Knowledge Distribution (P7,P32)
Gamification (P35)
Improving Code Review Feedback Time for non-Cores (P19)

Figure 11: Mind map of mitigation approaches to knowledge loss in OSS projects

In the literature relating to OSS projects the emphasis is on the retention of contributors (Schilling *et al.* 2012; Constantinou and Mens 2017), the stability of contributors (Robles and

Gonzalez-Barahona 2006; Ayushi and Ashish 2014; Lin *et al.* 2017), reducing the associated risks by identifying successors to the maintenance task (Rigby *et al.* 2016) and effects of turnover mainly on quality and productivity (Izquierdo-Cortazar *et al.* 2009; Qin *et al.* 2014; Foucault *et al.* 2015; Rigby *et al.* 2016). One study also reflected on the behaviour of the contributors in OSS projects (Garcia *et al.* 2013), the presence of negative emotions before contributors leave. Empirical work mostly focuses on the contributors in OSS projects. We found 10 impacts of turnover on OSS projects as summarised in Figure 8.

In CSS projects, the contributors with different roles such as developers, analysts, and testers are bound by legal contracts that ascertain that they manage conflicts and demonstrate a certain level of collaboration. In OSS projects, the transient nature of contributors means they may not cooperate well if confronted with conflicts or when they are not motivated (Garcia *et al.* 2013). Furthermore, in OSS projects, our review found no evidence of specific rules or formal process for the initiation of KR process before a contributor leaves.

Typical strategies for KR in CSS are reactive in nature. These strategies include knowledge capture and storage by codification, conducting interviews and identifying lessons learned or best practice from projects that departing employees made contributions to (Daghfous *et al.* 2013). The outcome of these strategies captures a small portion of the knowledge created, acquired and used in the workplace (Daghfous *et al.* 2013). Similar to CSS, the knowledge sharing in OSS projects is reactive. The contributors ask for information when required through the asynchronous means of communication. In OSS projects, proactive knowledge retention practices are required that resonate with the work environment and are non-intrusive, non-invasive, without any overhead on the productivity of contributors, and promise freedom to practise by free will.

While elaborating knowledge collaborations in social media communities, emphasis is on two stages, the creation stage when information is developed and formed, and the retention stage when the created information is preserved and refined through ongoing collaboration (Ransbotham and Kane 2011). While contributor turnover is ongoing even after the community has a successful collaboration, it is suggested that social media communities must create and retain knowledge (Ransbotham and Kane 2011). In the case of OSS communities, it can be implied that KR is also a result of a two-stage process, where the first stage is about knowledge creation and the second stage enables the retention practices to manage knowledge. The knowledge gap created due to turnover of contributors in OSS projects indicates missing KR practices. In Figure 11, ten mitigation techniques to reduce knowledge loss are highlighted. Nine of these mitigation techniques focus on the practices that can be followed in OSS projects. For instance, improving the timing of feedback would be beneficial to boost the intrinsic motivation of non-core contributors and achieve uniform knowledge distribution on projects. The knowledge management literature emphasizes intrinsic motivation in promoting employees' knowledge sharing due to its consistently positive and lasting effect (Pee and Lee 2015). The practice of gamification is another way to share knowledge but also has implications for extrinsic motivation. The prospective employer will choose the contributor with the best answer to a question. Knowledge Management in OSS projects is considered a challenging task. We find abundant knowledge sharing but not any specific practices for KR in OSS projects. Recognising the current situation of knowledge management in OSS projects and realising its importance, we propose to have an addition of a KM evaluation metric to evaluate the health of OSS projects.

The concept of health metrics is proposed for stakeholders to assess whether a project initiative is likely to be sustainable and is worth supporting (Wahyudin *et al.* 2007). A status overview of health in OSS projects is evaluated from the project data available on web repositories. We propose OSS KM evaluation metrics understanding that knowledge relevant activities such as KR are required to reduce the impact of knowledge loss in OSS projects. KM evaluation metrics must be a part of health evaluations metrics in OSS projects. KM should be one of the evaluation factors to assess the knowledge sharing activities of OSS projects for their sustainability in the future. We need additional metrics for KM evaluation in OSS projects, similar to the ones provided to evaluate the health of OSS projects such as developer and user community liveliness, product quality by using defect management of open issues and service delays, communication such as downloads, mailing lists posts and response rate.

8. Limitations of the Study

In this study, we used a snowballing search to find relevant papers instead of a traditional database search. It is arguable that there is a possibility that the snowballing method is weak and does not provide enough coverage of the relevant literature in this study. The snowballing method is shown in literature to provide coverage similar to database searches (Wohlin 2014). The snowballing search strategy can be effectively employed in place of the database search to find relevant papers. In order to overcome the first threat, we rigorously followed the steps of snowballing presented in section 3.

A second possible threat is the misinterpretation of the concepts during data synthesis, in terms of generalization to all OSS projects in the papers. To overcome the second threat, we consider the dynamic nature of OSS projects, which can be purely volunteer based or hybrid with commercial involvements. Not every OSS project community follow the same policies and practices and they vary constantly. The third threat in this study is that the concepts consolidated in this study are from the common understanding of the researchers, therefore themes in this work are subjective and personal.

There is a possibility that we missed some relevant literature while reading the paper titles in BSB and FSB. In the case where the text of the title was not clear, we read abstracts and conclusions of the papers to ensure that we do not disregard any important paper in our search. The objective of this study was to include papers that discussed knowledge loss in OSS projects due to contributor turnover. Papers that use OSS to design technological solutions for knowledge storage were not included in this literature review.

9. Conclusion

The objective of this study is to understand the phenomenon of knowledge loss due to contributor turnover in OSS projects. In order to understand the phenomenon of knowledge loss in OSS projects, we conducted a literature review using snowballing as a search strategy.

We identified 38 papers after filtering from a large number of papers (more than 2000) in a comprehensive review. The papers spread over the period of the year 2000 to 2017. The majority of the papers employed empirical methods as their research methods. We identified 10 impacts from knowledge loss and 10 mitigation techniques to overcome it in OSS projects

OSS projects are considered an extreme case of global software engineering with dynamic, dispersed, and transient contributors collaborating through technologically mediated channels. In comparison to contributors in CSS organisation, contributors in OSS are not contractually bound. To perform maintenance tasks on OSS projects, knowledge is acquired using asynchronous communication where the knowledge seeker asks questions. Delays incurred acquiring certain kind of knowledge to perform a task impacts the productivity of the contributor and the overall project. Further, the lack of knowledge results in poor quality code and increases the number of defects in the code repository.

Organisations invest in KM activities to organise, create, share, reuse, transfer, and retain knowledge. We found knowledge relevant activities in OSS projects namely knowledge creation and knowledge sharing. Knowledge sharing was found to be abundant but there was no evidence of knowledge retention to reduce the impact of knowledge loss in OSS projects. Moreover, knowledge sharing is reactive in nature, initiated by the contributor while looking for task relevant knowledge. We suggest that there is insufficient attention paid to KM in general in OSS, in particular, there would appear to be an absence of proactive measures to reduce the potential impact of knowledge loss. We also propose the need for a KM evaluation metric in OSS projects similar to the ones that evaluate the health of online communities. KM evaluation metrics should be based on the extent of knowledge sharing activities observed in a project. Such a metric could help to inform potential consumers of the OSS of the KM status on a project, something that is non-existent today. We consider it a vital ability for OSS projects to sustain a knowledge-sharing culture that will support the long-term survival and competitiveness of OSS projects.

The future direction for our work is to evaluate further practices that can be incorporated in the OSS project work structure. The work structure of OSS projects is informal, varies in each project, and is dynamic due to transient contributors; therefore, it is a challenging task to generalise a set of practices for all OSS projects.

References

Adams, P.J., Capiluppi, A. and Boldyreff, C. (2009). Coordination and productivity issues in free software: The role of brooks' law, in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, IEEE, 319-328.

Aggestam, L., Söderström, E. and Persson, A. (2010). Seven Types of Knowledge Loss in the Knowledge Capture Process, in *ECIS*.

Al-Emran, M., Mezhuyev, V., Kamaludin, A. and Shaalan, K. (2018). The impact of knowledge management processes on information systems: A systematic review, *International Journal of Information Management*, 43, 173-187, available: http://dx.doi.org/10.1016/j.ijinfomgt.2018.08.001.

Andersen-Gott, M., Ghinea, G. and Bygstad, B. (2012). Why do commercial companies contribute to open source software?, *International Journal of Information Management*, 32(2), 106-117, available: http://dx.doi.org/10.1016/j.ijinfomgt.2011.10.003.

Anquetil, N., de Oliveira, K., M., de Sousa, K., D. and Batista Dias, M., G. (2007). Software maintenance seen as a knowledge management issue, *Inf. Softw. Technol.*, 49(5), 515-529, available: http://dx.doi.org/10.1016/j.infsof.2006.07.007.

Ayushi, R. and Ashish, S. (2014). What Community Contribution Pattern Says about Stability of Software Project?, in *Software Engineering Conference (APSEC), 2014 21st Asia-Pacific*, 1-4 Dec. 2014, 31-34, available: http://dx.doi.org/10.1109/APSEC.2014.88.

Badampudi, D., Wohlin, C. and Petersen, K. (2015). Experiences from using snowballing and database searches in systematic literature studies, in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, ACM, 17.

Bao, L., Xing, Z., Xia, X., Lo, D. and Li, S. (2017). Who Will Leave the Company?: A Large-Scale Industry Study of Developer Turnover by Mining Monthly Work Report, in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 20-21 May 2017, 170-181, available: http://dx.doi.org/10.1109/MSR.2017.58.

Barão, A., de Vasconcelos, J.B., Rocha, Á. and Pereira, R. (2017). A knowledge management approach to capture organizational learning networks, *International Journal of Information Management*, 37(6), 735-740, available: http://dx.doi.org/10.1016/j.ijinfomgt.2017.07.013.

Basili, V.R. (1990). Viewing maintenance as reuse-oriented software development, *IEEE Software*, 7(1), 19-25.

BlackDuck, S. (2015). *Seventy-Eight Percent of Companies Run on Open Source, Yet Many Lack Formal Policies to Manage Legal, Operational, and Security Risk*, available: http://www.northbridge.com/2015FOOSSurveyResultsRelease [accessed N.p., 2017. Web. 8 June 2017].

Bosu, A. and Carver, J.C. (2014). Impact of developer reputation on code review outcomes in OSS projects: An empirical investigation, in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ACM, 33.

Britto, R., Smite, D. and Damm, L.-O. (2016). Software Architects in Large-Scale Distributed Projects: An Ericsson Case Study, *IEEE Software*, 33(6), 48-55.

Capiluppi, A., Gonzalez-Barahona, J.M., Herraiz, I. and Robles, G. (2007). Adapting the "staged model for software evolution" to free/libre/open source software, in *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, Dubrovnik, Croatia, 1294968: ACM, 79-82, available: http://dx.doi.org/10.1145/1294948.1294968.

Capiluppi, A. and Michlmayr, M. (2007). From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects in Feller, J., Fitzgerald, B., Scacchi, W. and Sillitti, A., eds., *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software, June 11–14, 2007, Limerick, Ireland*, Boston, MA: Springer US, 31-44.

Capiluppi, A., Stol, K.-J. and Boldyreff, C. (2012). Exploring the role of commercial stakeholders in open source software evolution, in *IFIP International Conference on Open Source Systems*, Springer, 178-200.

Chen, X., Li, X., Clark, J.G. and Dietrich, G.B. (2013). Knowledge sharing in open source software project teams: A transactive memory system perspective, *International Journal of Information Management*, 33(3), 553-563.

Choi, B., Poon, S.K. and Davis, J.G. (2008). Effects of knowledge management strategy on organizational performance: A complementarity theory-based approach, *Omega*, 36(2), 235-251, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.omega.2006.06.007.

Chong, A.Y.-L., Chan, F.T., Goh, M. and Tiwari, M. (2013). Do interorganisational relationships and knowledge-management practices enhance collaborative commerce adoption?, *International Journal of Production Research*, 51(7), 2006-2018.

Ciborra, C.U. and Andreu, R. (2001). Sharing knowledge across boundaries, *Journal of Information Technology*, 16(2), 73-81.

Clarke, P., O'Connor, R.V. and Leavy, B. (2016). A complexity theory viewpoint on the software development process and situational context, in *Software and System Processes (ICSSP), 2016 IEEE/ACM International Conference on*, IEEE, 86-90.

Colazo, J. and Fang, Y. (2009). Impact of license choice on open source software development activity, *Journal of the American Society for Information Science and Technology*, 60(5), 997-1011.

Conn, S. (2014) *Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments on Pace to Grow 7.6 Percent in 2014* [press release], available: www.gartner.com/newsroom/id/2645115

Constantinou, E. and Mens, T. (2017). An empirical comparison of developer retention in the RubyGems and npm software ecosystems, *Innovations in Systems and Software Engineering*, 13(2), 101-115, available: http://dx.doi.org/10.1007/s11334-017-0303-4.

Costa, E., Soares, A.L. and de Sousa, J.P. (2016). Information, knowledge and collaboration management in the internationalisation of SMEs: A systematic literature review, *International Journal of Information Management*, 36(4), 557-569, available: http://dx.doi.org/10.1016/j.ijinfomgt.2016.03.007.

Crowston, K. (2011). Lessons from volunteering and free/libre open source software development for the future of work in *Researching the Future in Information Systems* Springer, 215-229.

Crowston, K., Annabi, H., Howison, J. and Masango, C. (2004). Effective work practices for software engineering: free/libre open source software development, in *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*, Newport Beach, CA, USA, 1030003: ACM, 18-26, available: http://dx.doi.org/10.1145/1029997.1030003.

Crowston, K. and Howison, J. (2005). The social structure of free and open source software development, *First Monday*, 2(SPEC).

Crowston, K., Howison, J. and Annabi, H. (2006). Information systems success in free and open source software development: theory and measures, *Software Process: Improvement and Practice*, 11(2), 123-148, available: http://dx.doi.org/10.1002/spip.259.

Crowston, K., Wei, K., Howison, J. and Wiggins, A. (2012). Free/Libre Open-Source Software Development: What We Know and What We Do Not Know, *Acm Computing Surveys*, 44(2), available: http://dx.doi.org/10.1145/2089125.2089127.

Dafermos, G.N. (2005). Management and virtual decentralised networks: The Linux project (originally published in Volume 6, Number 11, November 2001), *First Monday*.

Daghfous, A., Belkhodja, O. and C. Angell, L. (2013). Understanding and managing knowledge loss, *Journal of knowledge management*, 17(5), 639-660.

Daniel, S., Midha, V., Bhattacherhjee, A. and Singh, S.P. (2018). Sourcing knowledge in open source software projects: The impacts of internal and external social capital on project success, *The Journal of Strategic Information Systems*, available: http://dx.doi.org/https://doi.org/10.1016/j.jsis.2018.04.002.

Davenport, T.H. and Prusak, L. (1998). *Working knowledge: How organizations manage what they know*, Harvard Business Press.

De Long, D.W. and Davenport, T. (2003). Better practices for retaining organizational knowledge: Lessons from the leading edge, *Employment Relations Today*, 30(3), 51-63.

de Vasconcelos, J.B., Kimble, C., Carreteiro, P. and Rocha, Á. (2017). The application of knowledge management to software evolution, *International Journal of Information Management*, 37(1, Part A), 1499-1506, available: http://dx.doi.org/10.1016/j.ijinfomgt.2016.05.005.

DeBrie, E. and Goeschel, D. (2016) 'Open Source Software Licenses', *The Nebraska Lawyer*.

Dingsøyr, T. (2005). Postmortem reviews: purpose and approaches in software engineering, *Inf. Softw. Technol.*, 47(5), 293-303, available: http://dx.doi.org/10.1016/j.infsof.2004.08.008.

Dingsoyr, T., Bjornson, F.O. and Shull, F. (2009). What Do We Know about Knowledge Management? Practical Implications for Software Engineering, *Software, IEEE*, 26(3), 100-103, available: http://dx.doi.org/10.1109/MS.2009.82.

Dinh-Trong, T. and Bieman, J.M. (2004). Open source software development: a case study of FreeBSD, in *Software Metrics, 2004. Proceedings. 10th International Symposium on*, 14-16 Sept. 2004, 96-105, available: http://dx.doi.org/10.1109/METRIC.2004.1357894.

Doan, Q.M., Rosenthal-Sabroux, C. and Grundstein, M. (2011). A Reference Model for Knowledge Retention within Small and Medium-sized Enterprises, in *KMIS*, 306-311.

Donadelli, S.M. (2015). *The impact of knowledge loss on software projects: turnover, customer found defects, and dormant files*, unpublished thesis, Concordia University, available: http://spectrum.library.concordia.ca/979964/.

Donnellan, B., Fitzgerald, B., Lake, B. and Sturdy, J. (2005). Implementing an open source knowledge base, *IEEE Software*, 22(6), 92-95.

Droege, S.B. and Hoobler, J.M. (2003). Employee turnover and tacit knowledge diffusion: A network perspective, *Journal of Managerial Issues*, 50-64.

Drucker, P.F. (1999). Knowledge-Worker Productivity: The Biggest Challenge, *California Management Review*, 41(2), 79-94.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review, *Information and Software Technology*, 50(9), 833-859.

Felizardo, K.R., Mendes, E., Kalinowski, M., Souza, É.F. and Vijaykumar, N.L. (2016). Using forward snowballing to update systematic reviews in software engineering, in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ACM, 53.

Feller, J. and Fitzgerald, B. (2002). *Understanding open source software development*, Addison-Wesley London.

Fitzgerald, B. (2006). The transformation of open source software, *Mis Quarterly*, 587-598.

Foucault, M., Palyart, M., Blanc, X., Murphy, G.C. and Falleri, J.-R. (2015). Impact of developer turnover on quality in open-source software, in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, Bergamo, Italy, 2786870: ACM, 829-841, available: http://dx.doi.org/10.1145/2786805.2786870.

Fronza, I., Janes, A., Sillitti, A., Succi, G. and Trebeschi, S. (2013). Cooperation wordle using pre-attentive processing techniques, in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (Chase)*, 25-25 May 2013, 57-64, available: http://dx.doi.org/10.1109/CHASE.2013.6614732.

Garcia, D., Zanetti, M.S. and Schweitzer, F. (2013). The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community, in *Cloud and Green Computing (CGC), 2013 Third International Conference on*, Sept. 30 2013-Oct. 2 2013, 410-417, available: http://dx.doi.org/10.1109/CGC.2013.71.

Ge, X., Dong, Y. and Huang, K. (2006). Shared knowledge construction process in an open-source software development community: an investigation of the <i>Gallery</i> community, in *Proceedings of the 7th international conference on Learning sciences*, Bloomington, Indiana, 1150062: International Society of the Learning Sciences, 189-195.

Hagan, D., Watson, O. and Barron, K. (2007). Ascending into order: A reflective analysis from a small open source development team, *International Journal of Information Management*, 27(6), 397-405, available: http://dx.doi.org/10.1016/j.ijinfomgt.2007.08.011.

Hansen, M.T., Nohria, N. and Tierney, T. (1999). What's your strategy for managing knowledge?, *The Knowledge Management Yearbook 2000–2001*, 55-69.

Hemetsberger, A. and Reinhardt, C. (2004). Sharing and creating knowledge in open-source communities: the case of KDE, in *Paper for Fifth European Conference on Organizational Knowledge, Learning, and Capabilities, Innsbruck*.

Herraiz, I., Robles, G., Amor, J.J., Romera, T. and González Barahona, J.M. (2006). The processes of joining in global distributed software projects, in *Proceedings of the 2006 international workshop on Global software development for the practitioner*, Shanghai, China, 1138513: ACM, 27-33, available: http://dx.doi.org/10.1145/1138506.1138513.

Hsu, C.-L. and Lin, J.C.-C. (2008). Acceptance of blog usage: The roles of technology acceptance, social influence and knowledge sharing motivation, *Information & Management*, 45(1), 65-74.

Huang, P. and Zhang, Z. (2016). Participation in Open Knowledge Communities and Job-Hopping: Evidence from Enterprise Software, *Management Information Systems Quarterly*, 40(3), 785-806.

Hutchison, C.S. (2001). Personal knowledge, team knowledge, real knowledge, in *EUROCON'2001*, 4-7 July 2001, 247-250 vol.1, available: http://dx.doi.org/10.1109/EURCON.2001.937805.

Huysman, M. and Lin, Y. (2005). Learn to solve problems: a virtual ethnographic case study of learning in a GNU/Linux users group, *The Electronic Journal for Virtual Organizations and Networks*, 7, 56-69.

Izquierdo-Cortazar, D., Robles, G., Ortega, F. and Gonzalez-Barahona, J.M. (2009). Using software archaeology to measure knowledge loss in software projects due to developer turnover, in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, IEEE, 1-10.

Jalali, S. and Wohlin, C. (2012). Systematic literature studies: database searches vs. backward snowballing, in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, ACM, 29-38.

Jennex, M.E. and Durcikova, A. (2013). Assessing knowledge loss risk, in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, IEEE, 3478-3487.

Jensen, C. and Scacchi, W. (2005). Modeling recruitment and role migration processes in OSSD projects, *ProSim05*, 39.

Joblin, M., Apel, S. and Mauerer, W. (2017). Evolutionary trends of developer coordination: A network approach, *Empirical Software Engineering*, 1-45.

Juntunen, J. and Raisanen, T. (2015). Organisational Knowledge Creation in Open Source Communities, in *10th IWKM* Bratislava, Slovakia.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering in *Technical report, Ver. 2.3 EBSE Technical Report. EBSE* sn.

Kitchenham, B. and Brereton, P. (2013). A systematic review of systematic review process research in software engineering, *Information and Software Technology*, 55(12), 2049-2075.

Koh, J. and Kim, Y.G. (2004). Knowledge sharing in virtual communities: an e-business perspective, *Expert Systems with Applications*, 26(2), 155-166, available: http://dx.doi.org/http://dx.doi.org/10.1016/S0957-4174(03)00116-7.

Kuk, G. (2006). Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List, *Manage. Sci.*, 52(7), 1031-1042, available: http://dx.doi.org/10.1287/mnsc.1060.0551.

Lakhani, K.R. and Von Hippel, E. (2003). How open source software works:"free" user-to-user assistance, *Research Policy*, 32(6), 923-943.

Lam, A. (2000). Tacit knowledge, organizational learning and societal institutions: An integrated framework, *Organization Studies*, 21(3), 487-513.

Lee, C.-P., Lee, G.-G. and Lin, H.-F. (2007). The role of organizational capabilities in successful e-business implementation, *Business Process Management Journal*, 13(5), 677-693.

Lee, G.K. and Cole, R.E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development, *Organization Science*, 14(6), 633-649, available: http://dx.doi.org/10.1287/orsc.14.6.633.24866.

Lee, S., Baek, H. and Jahng, J. (2017). Governance strategies for open collaboration: Focusing on resource allocation in open source software development organizations, *International Journal of Information Management*, 37(5), 431-437, available: http://dx.doi.org/10.1016/j.ijinfomgt.2017.05.006.

Levy, M. and Hazzan, O. (2009). Knowledge management in practice: the case of agile software development, in *Cooperative and Human Aspects on Software Engineering, 2009. CHASE'09. ICSE Workshop on*, IEEE, 60-65.

Licorish, S.A. and MacDonell, S.G. (2014). Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study, *Information and Software Technology*, 56(12), 1578-1596.

Lin, B., Robles, G. and Serebrenik, A. (2017). Developer turnover in global, industrial open source projects: insights from applying survival analysis, in *Proceedings of the 12th International Conference on Global Software Engineering*, IEEE Press, 66-75.

Lindvall, M. and Rus, I. (2003). Knowledge Management for Software Organizations in Aurum, A., Jeffery, R., Wohlin, C. and Handzic, M., eds., *Managing Software Engineering Knowledge*, Berlin, Heidelberg: Springer Berlin Heidelberg, 73-94.

Ling, K., Beenen, G., Ludford, P., Wang, X., Chang, K., Li, X., Cosley, D., Frankowski, D., Terveen, L. and Rashid, A.M. (2005). Using social psychology to motivate contributions to online communities, *Journal of Computer-Mediated Communication*, 10(4), 00-00.

Lungu, M. (2008). Towards reverse engineering software ecosystems, in *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, Sept. 28 2008-Oct. 4 2008, 428-431, available: http://dx.doi.org/10.1109/ICSM.2008.4658096.

Marks, M.A., Mathieu, J.E. and Zaccaro, S.J. (2001). A temporally based framework and taxonomy of team processes, *Academy of Management Review*, 26(3), 356-376.

Mens, T. (2016). An ecosystemic and socio-technical view on software maintenance and evolution, in *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*, IEEE, 1-8.

Michlmayr, M. (2007). *Quality Improvement in Volunteer Free and Open Source Software Projects: Exploring the Impact of Release Management*, unpublished thesis, University of Cambridge.

Mockus, A. (2010). Organizational volatility and its effects on software defects, in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, Santa Fe, New Mexico, USA, 1882311: ACM, 117-126, available: http://dx.doi.org/10.1145/1882291.1882311.

Mockus, A., Fielding, R.T. and Herbsleb, J.D. (2002). Two case studies of open source software development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346.

Nidhra, S., Yanamadala, M., Afzal, W. and Torkar, R. (2013). Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation, *International Journal of Information Management*, 33(2), 333-355, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.ijinfomgt.2012.11.004.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation, *Organization Science*, 5(1), 14-37.

Nonaka, I. and Takeuchi, H. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*, Oxford university press.

Nonaka, I., Toyama, R. and Konno, N. (2000). SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation, *Long Range Planning*, 33(1), 5-34, available: http://dx.doi.org/http://dx.doi.org/10.1016/S0024-6301(99)00115-6.

Octoverse (2016) 'GitHub Octoverse'.

Otte, T., Moreton, R. and Knoell, H.D. (2008). Applied quality assurance methods under the open source development model, in *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, 1247-1252.

Pee, L.G. and Lee, J. (2015). Intrinsically motivating employees' online knowledge sharing: Understanding the effects of job design, *International Journal of Information Management*, 35(6), 679-690, available: http://dx.doi.org/10.1016/j.ijinfomgt.2015.08.002.

Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs, *Cognitive psychology*, 19(3), 295-341.

Petticrew, M. and Roberts, H. (2006). *Systematic Reviews in the Social Sciences: A Practical Guide*.

Pontika, N., Knoth, P., Cancellieri, M. and Pearce, S. (2015). Fostering open science to research using a taxonomy and an eLearning portal, in *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, ACM, 11.

Qin, X., Salter-Townshend, M. and Cunningham, P. (2014). Exploring the Relationship between Membership Turnover and Productivity in Online Communities, in *ICWSM*.

Qumer, A. and Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice, *Journal of Systems and Software*, 81(11), 1899-1919.

Rahardjo Emanuel, A.W. (2014). Statistical analysis of popular open source software projects and their communities, in *Information Technology and Electrical Engineering (ICITEE), 2014 6th International Conference on*, 7-8 Oct. 2014, 1-6, available: http://dx.doi.org/10.1109/ICITEED.2014.7007913.

Ransbotham, S. and Kane, G.C. (2011). Membership turnover and collaboration success in online communities: Explaining rises and falls from grace in Wikipedia, *Mis Quarterly*, 613-627.

Rashid, M., Clarke, P.M. and O'Connor, R.V. (2017). Exploring Knowledge Loss in Open Source Software (OSS) Projects, in *International Conference on Software Process Improvement and Capability Determination*, Springer, 481-495.

Raymond, E. (1999). The cathedral and the bazaar, *Knowledge, Technology & Policy*, 12(3), 23-49.

Rhem, A.J. (2005). *UML for developing knowledge management systems*, CRC Press.

Rigby, P.C., German, D.M., Cowen, L. and Storey, M.-A. (2014). Peer Review on Open-Source Software Projects: Parameters, Statistical Models, and Theory, *ACM Trans. Softw. Eng. Methodol.*, 23(4), 1-33, available: http://dx.doi.org/10.1145/2594458.

Rigby, P.C., Zhu, Y.C., Donadelli, S.M. and Mockus, A. (2016). 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*, Austin, Texas.

Robles, G. and Gonzalez-Barahona, J.M. (2006) 'Contributor turnover in libre software projects', *IFIP International Federation for Information Processing*, 203, 273-286.

Robles, G., Gonzalez-Barahona, J.M. and Michlmayr, M. (2005). Evolution of volunteer participation in libre software projects: evidence from Debian, in *Proceedings of the 1st International Conference on Open Source Systems*, 100-107.

Rus, I. and Lindvall, M. (2002). Knowledge management in software engineering, *IEEE Software*, 19(3), 26.

Ryan, S. and O'Connor, R.V. (2012). Social interaction, team tacit knowledge and transactive memory: empirical support for the agile approach.

Ryan, S. and O'Connor, R.V. (2013). Acquiring and sharing tacit knowledge in software development teams: An empirical study, *Information and Software Technology*, 55(9), 1614-1624.

Scacchi, W. (2004). Free/open source software development practices in the computer game community, *IEEE Software*, 21(1), 59-67.

Scacchi, W. (2007). Free/open source software development, in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, Dubrovnik, Croatia, 1287689: ACM, 459-468, available: http://dx.doi.org/10.1145/1287624.1287689.

Schilling, A., Laumer, S. and Weitzel, T. (2011). 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*, 1620-1630.

Schilling, A., Laumer, S. and Weitzel, T. (2012). Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects, in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 4-7 Jan. 2012, 3446-3455, available: http://dx.doi.org/10.1109/HICSS.2012.644.

Schubert, P., Lincke, D.-M. and Schmid, B. (1998). A global knowledge medium as a virtual community: the NetAcademy concept, *AMCIS 1998 Proceedings*, 207.

Shah, S.K. (2006). Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development, *Management Science*, 52(7), 1000-1014.

Sharif, K.Y., English, M., Ali, N., Exton, C., Collins, J.J. and Buckley, J. (2015). 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, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.infsof.2014.09.003.

Silic, M. and Back, A. (2017). Open Source Software Adoption: Lessons from Linux in Munich, *IT Professional*, 19(1), 42-47, available: http://dx.doi.org/10.1109/MITP.2017.7.

Singh, V., Twidale, M.B. and Rathi, D. (2006). Open source technical support: A look at peer help-giving, in *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, IEEE, 118c-118c.

Sowe, S.K., Karoulis, A. and Stamelos, I. (2005). A constructivist view of knowledge management in open source virtual communities, *Managing Learning in Virtual Settings: The Role of Context, Idea Group, Inc*, 290-308.

Sowe, S.K., Stamelos, I. and Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study, *Journal of Systems and Software*, 81(3), 431-446, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.jss.2007.03.086.

Steinmacher, I., Conte, T., Gerosa, M., A. and Redmiles, D. (2015a). Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects, in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, Vancouver, BC, Canada, 2675215: ACM, 1379-1392, available: http://dx.doi.org/10.1145/2675133.2675215.

Steinmacher, I., Silva, M.A.G., Gerosa, M.A. and Redmiles, D.F. (2015b). A systematic literature review on the barriers faced by newcomers to open source software projects, *Information and Software Technology*, 59, 67-85.

Tiwana, A. (2004). An empirical study of the effect of knowledge integration on software development performance, *Information and Software Technology*, 46(13), 899-906, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.infsof.2004.03.006.

Tyagi, S., Cai, X., Yang, K. and Chambers, T. (2015). Lean tools and methods to support efficient knowledge creation, *International Journal of Information Management*, 35(2), 204-214, available: http://dx.doi.org/10.1016/j.ijinfomgt.2014.12.007.

Vasilescu, B., Serebrenik, A., Devanbu, P. and Filkov, V. (2014). How social Q&A sites are changing knowledge sharing in open source software communities, in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, Baltimore, Maryland, USA, 2531659: ACM, 342-354, available: http://dx.doi.org/10.1145/2531602.2531659.

Venkitachalam, K. and Willmott, H. (2017). Strategic knowledge management—Insights and pitfalls, *International Journal of Information Management*, 37(4), 313-316, available: http://dx.doi.org/10.1016/j.ijinfomgt.2017.02.002.

Venzin, M., Von Krogh, G. and Roos, J. (1998). Future research into knowledge management, *Knowing in firms: Understanding, managing and measuring knowledge*, 26-66.

Viana, D., Conte, T., Marczak, S., Ferreira, R. and Souza, C.d. (2015). Knowledge Creation and Loss within a Software Organization: An Exploratory Case Study, in *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, 5-8 Jan. 2015, 3980-3989, available: http://dx.doi.org/10.1109/HICSS.2015.477.

Vinogradova, V. and Galandere-Zile, I. (2005). *Where is the border between an information system and a knowledge management system?*, unpublished thesis, Univerza na Primorskem, Fakulteta za management.

von Krogh, G., Spaeth, S. and Haefliger, S. (2005). 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*, IEEE, 198b-198b.

von Krogh, G., Spaeth, S. and Lakhani, K.R. (2003). Community, joining, and specialization in open source software innovation: a case study, *Research Policy*, 32(7), 1217-1241, available: http://dx.doi.org/http://dx.doi.org/10.1016/S0048-7333(03)00050-7.

Wahyudin, D., Mustofa, K., Schatten, A., Biffl, S. and Min Tjoa, A. (2007). Monitoring the "health" status of open source web-engineering projects, *International Journal of Web Information Systems*, 3(1/2), 116-139.

Wang, X. and Lantzy, S. (2011). A systematic examination of member turnover and online community health, in *ICIS 2011 Proceedings. 25*.

Wohlin, C. (2014). 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*, ACM, 38.

Xu, B. (2006). *Volunteers' Participative Behaviors in Open Source Software Development: The Role of Extrinsic Incentive, Intrinsic Motivation and Relational Social Capital*, unpublished thesis, Texas Tech University, available: http://repositories.tdl.org/ttu-ir/handle/2346/1284.

Ye, Y., Nakakoji, K., Yamamoto, Y. and Kishida, K. (2005). The co-evolution of systems and communities in free and open source software development, *Free/open source software development*, 59-82.

Yilmaz, M., Yilmaz, M., O'Connor, R.V. and Clarke, P. (2016). A gamification approach to improve the software development process by exploring the personality of software practitioners, in *International Conference on Software Process Improvement and Capability Determination*, Springer, 71-83.

Yu, Y., Benlian, A. and Hess, T. (2012). An Empirical Study of Volunteer Members' Perceived Turnover in Open Source Software Projects, in *2012 45th Hawaii International Conference on System Sciences*, 4-7 Jan. 2012, 3396-3405, available: http://dx.doi.org/10.1109/HICSS.2012.97.

Zack, M.H. (1999). Developing a knowledge strategy, *California Management Review*, 41(3), 125-145.

Zack, M.H. (1999). Managing Codified Knowledge, *Sloan Management Review*, 40(4), 45-58.

Zagalsky, A., Teshima, C.G., German, D.M., Storey, M.-A. and Poo-Caamaño, G. (2016). How the R community creates and curates knowledge: a comparative study of stack overflow and mailing lists, in *Proceedings of the 13th International Workshop on Mining Software Repositories*, ACM, 441-451.

Zheng, X., Zeng, D., Li, H. and Wang, F. (2008). Analyzing open-source software systems as complex networks, *Physica A: Statistical Mechanics and its Applications*, 387(24), 6190-6200, available: http://dx.doi.org/10.1016/j.physa.2008.06.050.

Zhou, M. and Mockus, A. (2015). Who Will Stay in the FLOSS Community? Modeling Participant's Initial Behavior, *IEEE Transactions on Software Engineering*, 41(1), 82-99, available: http://dx.doi.org/10.1109/TSE.2014.2349496.

Zhou, Q. (2009). The Impact of Job Satisfaction Affect on Turnover Intention: An Empirical Study Based on the Circumstances of China, in *2009 Second International Conference on Education Technology and Training*, 13-14 Dec. 2009, 220-223, available: http://dx.doi.org/10.1109/ETT.2009.80.

Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge, *Journal of the American Society for Information Science and Technology*, 58(4), 479-493, available: http://dx.doi.org/10.1002/asi.20508.

Appendix I: Primary Studies

P1. Wang, X. and Lantzy, S. (2011). A systematic examination of member turnover and online community health, in *ICIS 2011 Proceedings. 25*.

P2. Mens, T. (2016). An ecosystemic and socio-technical view on software maintenance and evolution, in *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*, IEEE, 1-8.

P3. Constantinou, E. and Mens, T. (2017). An empirical comparison of developer retention in the RubyGems and npm software ecosystems, *Innovations in Systems and Software Engineering*, 13(2), 101-115, available: http://dx.doi.org/10.1007/s11334-017-0303-4.

P4. Fronza, I., Janes, A., Sillitti, A., Succi, G. and Trebeschi, S. (2013). Cooperation wordle using pre-attentive processing techniques, in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (Chase)*, 25-25 May 2013, 57-64, available: http://dx.doi.org/10.1109/CHASE.2013.6614732.

P5. Joblin, M., Apel, S. and Mauerer, W. (2017). Evolutionary trends of developer coordination: A network approach, *Empirical Software Engineering*, 1-45.

P6. Rashid, M., Clarke, P.M. and O'Connor, R.V. (2017). Exploring Knowledge Loss in Open Source Software (OSS) Projects, in *International Conference on Software Process Improvement and Capability Determination*, Springer, 481-495.

P7. Lin, B., Robles, G. and Serebrenik, A. (2017). Developer turnover in global, industrial open source projects: insights from applying survival analysis, in *Proceedings of the 12th International Conference on Global Software Engineering*, IEEE Press, 66-75.

P8. Foucault, M., Palyart, M., Blanc, X., Murphy, G.C. and Falleri, J.-R. (2015). Impact of developer turnover on quality in open-source software, in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, Bergamo, Italy, 2786870: ACM, 829-841, available: http://dx.doi.org/10.1145/2786805.2786870.

P9. Nidhra, S., Yanamadala, M., Afzal, W. and Torkar, R. (2013). Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation, *International Journal of Information Management*, 33(2), 333-355, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.ijinfomgt.2012.11.004.

P10. Huang, P. and Zhang, Z. (2016). Participation in Open Knowledge Communities and Job-Hopping: Evidence from Enterprise Software, *Management Information Systems Quarterly*, 40(3), 785-806.

P11. Rigby, P.C., Zhu, Y.C., Donadelli, S.M. and Mockus, A. (2016). 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*, Austin, Texas.

P12. Britto, R., Smite, D. and Damm, L.-O. (2016). Software Architects in Large-Scale Distributed Projects: An Ericsson Case Study, *IEEE Software*, 33(6), 48-55.

P13. Izquierdo-Cortazar, D., Robles, G., Ortega, F. and Gonzalez-Barahona, J.M. (2009). Using software archaeology to measure knowledge loss in software projects due to developer turnover, in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, IEEE, 1-10.

P14. Rastogi, A. and Sureka, A. (2014). What community contribution pattern says about stability of software project?, in 2014, IEEE, 31-34.

P15. Bao, L., Xing, Z., Xia, X., Lo, D. and Li, S. (2017). Who Will Leave the Company?: A Large-Scale Industry Study of Developer Turnover by Mining Monthly Work Report, in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 20-21 May 2017, 170-181, available: http://dx.doi.org/10.1109/MSR.2017.58.

P16. Schilling, A., Laumer, S. and Weitzel, T. (2012). Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects, in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 4-7 Jan. 2012, 3446-3455, available: http://dx.doi.org/10.1109/HICSS.2012.644.

P17. Ransbotham, S. and Kane, G.C. (2011). Membership turnover and collaboration success in online communities: Explaining rises and falls from grace in Wikipedia, *Mis Quarterly*, 613-627.

P18. Garcia, D., Zanetti, M.S. and Schweitzer, F. (2013). The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community, in *Cloud and Green Computing (CGC), 2013 Third International Conference on*, Sept. 30 2013-Oct. 2 2013, 410-417, available: http://dx.doi.org/10.1109/CGC.2013.71.

P19. Bosu, A. and Carver, J.C. (2014). Impact of developer reputation on code review outcomes in OSS projects: An empirical investigation, in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ACM, 33.

P20. Mockus, A., Fielding, R.T. and Herbsleb, J.D. (2002). Two case studies of open source software development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346.

P21. Robles, G. and Gonzalez-Barahona, J.M. (2006) 'Contributor turnover in libre software projects', *IFIP International Federation for Information Processing*, 203, 273-286.

P22. Wahyudin, D., Mustofa, K., Schatten, A., Biffl, S. and Min Tjoa, A. (2007). Monitoring the "health" status of open source web-engineering projects, *International Journal of Web Information Systems*, 3(1/2), 116-139.

P23. Otte, T., Moreton, R. and Knoell, H.D. (2008). Applied quality assurance methods under the open source development model, in *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, 1247-1252.

P24. Crowston, K., Wei, K., Howison, J. and Wiggins, A. (2012). Free/Libre Open-Source Software Development: What We Know and What We Do Not Know, *Acm Computing Surveys*, 44(2), available: http://dx.doi.org/10.1145/2089125.2089127.

P25. Capiluppi, A., Gonzalez-Barahona, J.M., Herraiz, I. and Robles, G. (2007). Adapting the "staged model for software evolution" to free/libre/open source software, in *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, Dubrovnik, Croatia, 1294968: ACM, 79-82, available: http://dx.doi.org/10.1145/1294948.1294968.

P26. Vasilescu, B., Serebrenik, A., Devanbu, P. and Filkov, V. (2014). How social Q&A sites are changing knowledge sharing in open source software communities, in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, Baltimore, Maryland, USA, 2531659: ACM, 342-354, available: http://dx.doi.org/10.1145/2531602.2531659.

P27. Nonaka, I., Toyama, R. and Konno, N. (2000). SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation, *Long Range Planning*, 33(1), 5-34, available: http://dx.doi.org/http://dx.doi.org/10.1016/S0024-6301(99)00115-6.

P28. Robles, G., Gonzalez-Barahona, J.M. and Michlmayr, M. (2005). Evolution of volunteer participation in libre software projects: evidence from Debian, in *Proceedings of the 1st International Conference on Open Source Systems*, 100-107.

P29. Rigby, P.C., German, D.M., Cowen, L. and Storey, M.-A. (2014). Peer Review on Open-Source Software Projects: Parameters, Statistical Models, and Theory, *ACM Trans. Softw. Eng. Methodol.*, 23(4), 1-33, available: http://dx.doi.org/10.1145/2594458.

P30. Martinez-Romo, J., Robles, G., Gonzalez-Barahona, J.M. and Ortuño-perez, M. (2008) 'Using social network analysis techniques to study collaboration between a FLOSS community and a company', *IFIP International Federation for Information Processing*, 275, 171-186.

P31. Steinmacher, I., Silva, M.A.G., Gerosa, M.A. and Redmiles, D.F. (2015). A systematic literature review on the barriers faced by newcomers to open source software projects, *Information and Software Technology*, 59, 67-85.

P32. Licorish, S.A. and MacDonell, S.G. (2014). Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study, *Information and Software Technology*, 56(12), 1578-1596.

P33. Lee, G.K. and Cole, R.E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development, *Organization Science*, 14(6), 633-649, available: http://dx.doi.org/10.1287/orsc.14.6.633.24866.

P34. Qin, X., Salter-Townshend, M. and Cunningham, P. (2014). Exploring the Relationship between Membership Turnover and Productivity in Online Communities, in *ICWSM*.

P35. Zagalsky, A., Teshima, C.G., German, D.M., Storey, M.-A. and Poo-Caamaño, G. (2016). How the R community creates and curates knowledge: a comparative study of stack overflow and mailing lists, in *Proceedings of the 13th International Workshop on Mining Software Repositories*, ACM, 441-451.

P36. Sowe, S.K., Stamelos, I. and Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study, *Journal of Systems and Software*, 81(3), 431-446, available: http://dx.doi.org/http://dx.doi.org/10.1016/j.jss.2007.03.086.

P37. Kuk, G. (2006). Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List, *Manage. Sci.*, 52(7), 1031-1042, available: http://dx.doi.org/10.1287/mnsc.1060.0551.

P38. Chen, X., Li, X., Clark, J.G. and Dietrich, G.B. (2013). Knowledge sharing in open source software project teams: A transactive memory system perspective, *International Journal of Information Management*, 33(3), 553-563.