A Mechanism to Explore Proactive Knowledge Retention in Open Source Software Communities

Mehvish Rashid[1,2], Paul M. Clarke[1, 2], Rory V. O'Connor[1, 2]

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

ABSTRACT

Open-source software (OSS) is a type of computer software where in the source code is distributed under a special type of license in which the copyright holder grants users the rights to inspect, alter and redistribute the software. OSS projects are collaborative endeavours which have multiple contributors who are constantly joining, leaving, or changing their role in the project. This ever changing and ever transient nature of OSS project contributors contributes to a contributor turnover induced knowledge loss in OSS projects. In this case 'Knowledge Loss' refers to the phenomenon of the loss of project specific knowledge, experience and expertise in an OSS project, caused by contributors regularly joining and leaving the OSS project. This paper describes the design and development of a robust research methodology and contributes towards the formation of proactive knowledge retention practices in OSS projects to transform contributor's use of knowledge and engagement in knowledge relevant activities including knowledge sharing and knowledge transfer.

Keywords
Software Development Process; Open Source Software, Knowledge Loss, Proactive Knowledge Retention, Mixed Methods Research.

1. INTRODUCTION

The design and development of software intensive systems may be characterised as a human centric knowledge-based activity involves a substantial amount of complexity [1]. There are many different aspects of knowledge management required to successfully implement software development projects, such as domain specific knowledge, system's architecture knowledge, detailed understanding of requirements, computer programming language expertise and knowledge of the development environment [2]. Software development knowledge is information combined with experience, context, interpretation and reflection [3]. In a typical software development project, expert knowledge is generated in a continuous manner through the process of knowledge creation and sharing, which is cyclic in its nature. Therefore, software development organizations often struggle to adapt their processes and businesses [5]. As a result, knowledge management and knowledge retention (whether explicit or implicit) in particular should be a key concern for software development organizations. However, an examination of the literature does not reveal the existence of a formal knowledge retention reference model. We consider that this lack of a model addressing knowledge loss in OSS projects is a significant weakness and therefore the present study sees to address this. The research described in this paper in restricted to the context of OSS and explicitly not that of commercial (or closed source software) which operate under different knowledge retention strategies, as developers are employees and unlike OSS contributors who are volunteers, and therefore more formal knowledge sharing mechanisms can be imposed in commercial situations.

OSS is a term used to describe software, which has been developed and released under an "open source" license that complies with Open Source Definition (OSD). A significant factor sin OSD is "the freedom to use, change, sell or give away the software, the availability of source code and the protection of authors' intellectual property rights" [4]. Accordingly, users may freely access the software source code, including inspection, modification and redistribution of the source code [5, 6]. In recent years there has been an enormous increase in the number of OSS projects. For example [7] indicate that the Source Forge portal alone hosted over 430,000 OSS projects in 2014. OSS projects may be of various sizes, including very large projects and may also involve commercial firms [8]. A

2015 survey reported that almost 78% of companies utilize OSS and 66% of companies have incorporated OSS into software delivered to customers [9].

In contrast to contributors in OSS projects, employees in traditional (closed source) software development organisation may be under contractual obligation to notify their employer before leaving the organisation and to fulfil a notice period during which knowledge transfer concerns can be addressed. The workforce in OSS projects is of a transient nature due to the inevitably high turnover rate [10, 11] as most software developers contribute to OSS projects “for free”, either as a hobby or during their regular working hours even when OSS development is not part of their usual work. Given this nature of involvement OSS contributors further who leave a project may provide no notice to the development community, thus not providing an opportunity for formal knowledge transfer activities. It should be noted that in contrast to this mainstream volunteerism model, some commercial organisations actively encourage employees to contribute to OSS project of interest to the company. Although this changes somewhat the voluntary nature of OSS development, it does not affect the general OSS principle that the source code is available to everybody [12].

The structure of OSS projects is often referred to as being a ‘hierarchical onion-like structure’ (as illustrated in Figure 1), which consists of core developers, co-developers, active users, and passive users [13-15]. In this structure the knowledge distribution among the various contributors is not uniform and the absence of a contributor who is the original owner of the files or system in the project to perform maintenance tasks potentially results in risking files to abandonment [16]. In a typical OSS project there is a small subset of ‘core contributors’ (typically accounting for about 20% if total number of volunteers), make major code contributions, accounting for about 80% of the code in a typical in OSS project [5]. These core contributors are considered to be the most knowledgeable project members, therefore, as depicted in Figure 1, knowledge distribution in OSS projects is non-uniform with a higher concentration of code contributors in the centre of the onion than in the outer layers.

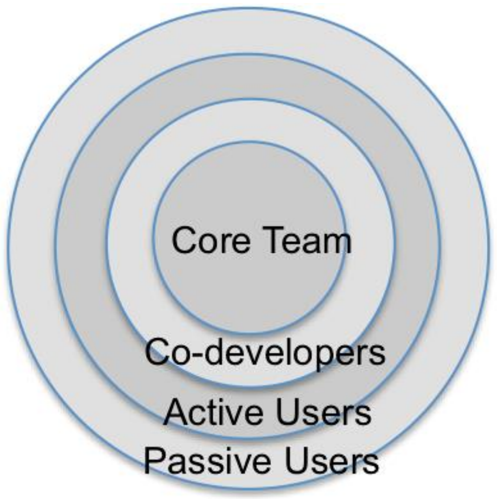

Fig. 1. Onion model representing contributors in OSS projects users [3-5]

When discussing software development knowledge in either open source of closed source projects, it is ecessry to differentiate between three key terms: *data*, *information*, and *knowledge*. *Data* is considered to represent observations and facts without any contextual meaning. In contrast *information* is the result of associating data with a meaningful context [17], where data has been converted into information by being contextualized, categorised, calculated and condensed [17]. *Knowledge* is driven from information [17] and is the product of an individual’s experience and accumulates as a result of communication or inference [18]. Furthermore at a high-level knowledge may be categorised as either *explicit* or *tacit* (or implicit) knowledge [19, 20]. Tacit knowledge comprises of skills learned due to the personal capabilities of contributors and if not documented, remains confined to an individual, whereas explicit knowledge is available in documented form [19]. In either a closed source or open source organisation/project knowledge is created at an organisational level as a result of the interaction between both tacit and explicit knowledge [19].

For any given software development endeavour there are a multitude of potential software development process which can be employed, with the choice of each one dependent on many situational factors process [21]. At a high-level the two major development process choices can be c characterised as traditional (or plan based), which rely primarily on managing explicit knowledge, or agile methods, which primarily rely on managing tacit knowledge [22] and recognises the importance of human interaction in the software development process over written knowledge in formal documentation. Regardless of development process approach utilised, communication is considered a crucial factor, in particular for tacit knowledge exchange, as this relies heavily on informal and predominantly verbal [23]. Understanding the role and nature of tacit knowledge in agile software development is challenging.

Members of software development teams are often referred to as 'knowledge workers' [24-25] who are typically individuals with specialist skill and ability to apply these skills to identify and solve problems. Such knowledge workers' work with intangible cognitive processes, where knowledge, especially tacit knowledge is contained in the minds of individual and is the means of production in software development. Knowledge sharing is therefore a key process in developing software, and since expert knowledge is tacit, the acquisition and transmission of tacit knowledge is significant in the development process [26].

The majority of research relating to knowledge management in OSS projects has concentrated on knowledge shared and knowledge reuse, cross boundary learning in open communities [112]. Where knowledge sharing involving strategic interaction, learning theory on knowledge creation and sharing in online communities, learning driven by criticism and error correction, a social view of learning that have overcome problem of tacit knowledge transformation. However, it is the proposition of this research that the significant issue of Knowledge Retention in OSS projects has not received the attention it deserves in the research literature.

Knowledge retention is effectively the act of building organizational memory and involves capturing knowledge and retaining it in a community or organization so that it can be used. The fundamental nature of OSS projects means that they evolve over a period of time and are developed by multiple contributors who are constantly joining, leaving, or changing their role in the project. The phenomenon of resources joining and leaving in this fashion is referred as 'turnover' [27]. Sudh change (turnover) in contributors in OSS projects led to knowledge loss in OSS projects. In many large OSS projects, a high turnover has been observed leading to the formation of the new development teams [28]. Knowledge loss impacts the productivity of the OSS projects in two ways: 1) The effort required to acquire knowledge to perform the maintenance tasks; and 2) The loss of effort when code is orphaned and removed from the project [39].

In order to write quality software code knowledgeable contributors are required. Searching knowledge is argued to be time consuming and costly [30]. The search efforts can vary depending on the source and the level of details. A post or a query on the project mailing list require less efforts while searching through the results of search engine or examining the clues into source code documentation is time consuming [30]. A study on the GNOME project (a well-known large OSS project sponsored by several companies) reported that 30 months' time is needed for the contributor to understand the software code and to make a contribution [31]. Developers gradually become productive taking more than a year's time on a project to reach productivity plateau [32]. The time to complete distributed tasks is estimated to be three times longer than for co-located tasks [33].

The time required by a new person to learn the inner workings of the project when experienced contributors leave, causes considerable productivity loss [34]. In-depth understanding of software code and interconnecting file structure is not required to complete simple tasks. On the other hand, contributors may have difficulty performing non-trivial tasks due to ‘information blocking’, unavailability of the relevant information to complete a task [34]. The productivity of the contributor and overall project suffers due to the information blocking and a lack of understanding of the code base. According to estimates, information blocking consumes 60% of developers efforts [35].

During the preparation of a release, contributors make changes to align their work with the goals of the release [36]. As abandoned code increases on the project, the numbers of reported defects increase as well [37]. The maintenance of abandoned code is difficult because the team lacks knowledge of its creation and structure [35]. The source code that remains unmaintained (unless a legacy system) has an element of uncertainty for the development team since the contributors who wrote it have left the project [34]. Removal of unmaintained code results in loss of existing functionality and may impact users of the system [36].

In OSS projects, knowledge sharing is an ongoing activity in an intensely people-oriented and self-organised community. As we shall demonstrate, this activity might also be considered to be characterised as both reactive and somewhat disorganised. In such a setup, knowledge is dispersed in the community of contributors interacting on a project and is not limited within a small group [39]. Knowledge sharing is through asynchronous means of communication and with a collection of artifacts, which are publicly available for reuse.

Knowledge retention can be seen as a way of embedding and enabling knowledge within an organisation and a critical factor for sustainable performance [38]. It is an effort-demanding task to identify potential knowledge for the organisation. The structure of the organisation in the context of how well it supports knowledge retention is of importance. Once the person who has the potential knowledge leaves the organisation, it is hard to retain this knowledge.

The focus of this research is on the uniform distribution of knowledge among contributors, by the introduction of continuous knowledge transfer techniques and practices, which we refer to as a proactive knowledge retention strategy for OSS projects. This paper presents the research methodology to investigate the knowledge loss problem in OSS projects articulated in [39, 40]. The research methodology discusses the overall research approach to be employed to devise a proactive knowledge retention strategy in OSS projects.

The knowledge loss due to the loss of experience and expertise on the project impacts productivity and additional time is required to learn the workings of the project when original contributors are no longer accessible [34]. The central hypothesis of this work states that the knowledge loss in OSS projects due to contributor turnover can be reduced by introducing proactive knowledge retention practices in OSS projects that can transform contributor’s use of knowledge and engagement in knowledge relevant activities including knowledge sharing and knowledge transfer.

In OSS projects inevitable turnover due to the transient nature of contributors [11] and absence of contractual bindings for notification before contributors leave, make it difficult to enable any reactive knowledge transfer activity. Therefore, the reactive approach of knowledge retention that may be practised in a traditional software organisation may prove entirely ineffective for OSS projects. The central hypothesis stresses the need for proactive knowledge retention practices in OSS projects arising from the difference of organisational and governance structure between OSS and traditional software organisations. In section 2, we identify our research objectives, while section 3 presents the research methodology and Section 4 outlines the data collection and analysis aspects of our work. Section 5 identifies some research limitations and finally in Section 6, we present a discussion and identify possible areas for future work.

2. RESEARCH OBJECTIVES

In the light of the above discussion, the key research objective is conceptualised in the following statement: *Knowledge loss in OSS projects due to contributor turnover can be reduced by introducing* *proactive Knowledge Retention practices in OSS projects that can transform contributors' use of knowledge and engagement in knowledge relevant activities including knowledge sharing and knowledge transfer.*

The research objective stresses the need for proactive Knowledge Retention practices in OSS projects arising from the difference of organisational and governance structure between OSS and traditional software organisations. OSS projects are dynamic, dispersed, with transient contributors concurrently performing tasks in different roles, and collaborating through technologically mediated channels. The central hypothesis leads to the first research question: **RQ1. What practices can enable effective proactive knowledge retention strategy in OSS projects?**

OSS projects are dynamic, dispersed, with transient contributors concurrently performing tasks in different roles, and collaborating through technologically mediated channels. The challenge is to formulate the strategy for proactive knowledge retention, which can resonate with the idiosyncratic nature of OSS projects without causing an overhead to the productivity of the project and contributors. The OSS communities focus on self-direction and favour intrinsic value system and therefore imposing a strategy that calls for contributors to limit their freedom will lower their enthusiasm of doing well for the society. The next step after identification of retention practices requires their incorporation with the OSS projects work settings, which leads to the second research question [41]: **RQ2. How to incorporate proactive knowledge retention strategy with established work practices in OSS projects?**

In the next section, we describe the methodology to investigate the first research question.

3. RESEARCH METHODOLOGY

Worldviews provide a general philosophical direction to research with common elements having different stances [42]. Philosophical worldviews shape the approach taken for research by influencing research designs and research methods [43]. Worldviews differ in: ontology which is the nature of reality; epistemology, which refers to how we gain knowledge about what we know; axiology which explains role of values in research; methodology determines the process of research; and rhetoric is the language of research [44]. The three worldviews reflected as different philosophical concerns in research are positivism (also called post positivism), interpretivism, and pragmatism. Underlying philosophical concerns further determine the selection of the research method to conduct any research.

- **Positivism**. The positivists advocate in the quantification of their learning through numbers and the use of statistical equations to predict human behaviour [45]. A positivist believes that social life is pretty stable and constant [46]. In such an approach, if the learning of a concept is not possible through quantifiable methods it is generally ignored. The positivists extract simple relationships from a complex real world in numbers without considering the context [46]. Positivists stress deterministic philosophy, reductionism, observation and measurement, and theory verification [43]. In deterministic philosophy, causes determine the effects or outcomes. Reductionism is about reducing ideas into small discrete tests consisting of variables based on hypothesis and research questions. Knowledge development by positivist is through observing and measuring objective reality in numbers, and by verification of laws and theories that govern the world.
- **Interpretivism**. Interpretivist focuses more on human thoughts and actions in social and organisational contexts [47]. Interpretivists (also called constructivists) believe in understanding the context and meaning by taking into account the real setting of the world in which they live and work. Interpretivist led research tends to develop subjective, varied, and multiple meanings about an object enabling them to unfold complex views [43]. The views on the situation being studied are collected from as many participants as possible [43]. For example, in open-ended questioning, a researcher carefully listens to the views of participants and shapes his or her interpretation from cultural and historical experiences [43]. Research led

by an interpretivist generates or inductively develops a theory or pattern from the interpretation [43].

- **Pragmatism**. advocates an alternative world view to positivism and interpretivism and primarily focuses on the problem to be researched and the consequences of the research [42]. Pragmatism offers a middle position or mixed methods research movement with a practical and outcome-oriented method of inquiry based on action and leads by enabling researchers to have better answers to their research questions [48]. Furthermore, pragmatism takes a value-oriented approach to research and reach an agreement about importance of culturally derived values and desired conclusion [48]. Pragmatism as a philosophical underpinning for mixed methods studies, focuses on the research problem [49]. Pragmatism uses pluralistic approaches to derive knowledge about the problem [49] and is concerned with real-world practice.

The philosophical position adopted in this research is that of pragmatism, with a focus on the research problem and enabling empirical research to find answers to the research questions. The pragmatist worldview reflects the direct action oriented approach of a researcher towards the investigation of the research problem at hand. We suggest that this research would benefit by adopting pragmatism and approaching the problem by understanding it in a practical manner and by using multiple methods in research. The goal of this research is to devise a systematic strategy on the use of proactive knowledge retention practices in OSS projects. The development of an overarching proactive knowledge retention strategy requires understanding of the phenomenon and exploration in real life with multiple contexts and with the ability to quantify the concepts.

The view taken in this research is that of taking a middle position between two extremities of being a positivist or an interpretivist. Solely being an interpretivist or positivist is inadequate in terms of addressing the objective of this research. The insights provided by the use of qualitative and quantitative research in a mix method can be integrated into a workable solution under pragmatism [48]. The goal of mixed methods research is not to replace either qualitative or quantitative approach but to use their strengths and minimize the weaknesses of both in single research studies and across studies [48].

Empirical studies are emphasised to provide a scientific basis for software engineering [50] and to investigate the social and cognitive processes surrounding complex software systems [51]. Empirical methods allow for informed and well-grounded decisions and allow the investigation of a phenomenon by experimenting and experiencing it in the real world settings [50]. In general, research methodology is a systematic approach to achieve particular goals of the research. The pragmatist worldview adopted in this research lays the foundation to the research methodology including the research design. Pragmatically inclined researchers focus more on the desired outcomes and solution to the problem. Mixed methods research applies pragmatist system of philosophy where “the researcher mixes or combines quantitative and qualitative research techniques, methods, approaches, concepts or language into a single study” [48]. A method is mainly a set of principles through which empirical data is collected and analysed [52]. Research methods can be classified as qualitative, quantitative or both (mixed) [50].

- **Quantitative Approach** mainly focus on deduction, confirmation, theory or hypothesis testing, explanation, prediction, standardized data collection, and statistical analysis [48]. Quantitative research is based on the use of statistical methods and establishes relationships between variables [53] or quantifies a relationship by comparing two or more groups [42], with an aim of identifying a cause-effect. The focus is on data collection that is largely of a numeric type and the required information is specified in advance and data is gathered using scaled instruments while interpretations are made on the basis of the statistical results [43]. The use of a questionnaire most likely includes a numerical rating scale for quantitative data collection [48]. In order to collect data, researchers can employ an instrument or test, which has a set of questions to evaluate the confidence towards an approach, or use checklist to identify and observe people involved in some task [43].

- **Qualitative Approach.** Qualitative research refers to the study of objects in their natural setting [50], where a qualitative researcher tries to understand the causes while interpreting a phenomenon by accepting that there are multiple interpretations of the explanations given to them by the subjects in the study [54]. Qualitative research believes in a range of different ways of interpretation and understanding views of the subjects on the concerned problem at hand [50]. Data collection involve observing the behaviour of individuals and conducting interviews with individuals where they can talk about a topic openly mostly without the use of specific questions. Researchers make interpretations from the themes or patterns that emerge from the data. Qualitative research theory may be applied after the data collection while following the process of coding (data analyses by labelling and categorising) [55]. Case studies, ethnographies, post-mortem analysis, action research [51], phenomenology, grounded theory, and narrative [43] are primarily qualitative in nature.
- **Mixed Methods Approach** is formally defined as the class of research where the researcher mixes or combines quantitative and qualitative research techniques, methods, approaches, concepts or language into a single study [48]. Mixed methods involves combining qualitative and quantitative research by the collecting qualitative data, which is open-ended and without predetermined responses, and quantitative data, which is closed-ended in nature and the selection of responses is from predetermined list of answers [43]. The mixed methods are valued and thought to neutralize the weakness and bias that arises in research by the usage of just one method [43].

The data collection for the current study is by using mixed methods that involve both qualitative data and quantitative data. A survey instrument will be employed for qualitative and quantitative data collection. The purpose of the data collection using mixed methods will be to analyse and evaluate the proposed practices based on the feedback from the OSS community. Furthermore, the collected feedback from the OSS community will be utilised for improving the proposed practices for proactive knowledge retention in OSS projects. The questionnaire for survey interview will be prepared after examining the literature on practices and developing a set of proposed practices for proactive knowledge retention in OSS projects. Rather than using one type of data collection the combination of both quantitative data (close-ended questions with a larger sample) and qualitative data (open-ended questions with a smaller sample) overcomes the weakness arising from the use of only one data type. The analysis for each type of data collection will be performed separately and then results will be combined to understand the knowledge retention phenomenon in OSS projects.

There are different possible ways in a mixed methods design, to converge or to merge the data from two different types of data collections. The two data collections are analysed separately and then results are explained. The two data collections can be merged by a side-by-side comparison, by transformation procedure and by table or graph [43]. In side-by-side comparison, the results from both types of data collection are reported separately and then findings are compared. In transformation procedure, the qualitative data is transformed to quantitative data by changing the qualitative themes into quantitative variables. The two types of data collection can also be merged in the form of tables and graph while display. In this research, the data analysis for quantitative and qualitative data will be performed independent of each other. The findings from both types of analysis will be combined to understand and elaborate the phenomenon. The results from the analysis of two data collections will be interpreted and produced in writing.

The summary of the research methodology is illustrated in Figure 2, where rectangles represent survey design, data collection, or data analysis, while an oval represents merging the findings, interpreting, or incorporating them in OSS Knowledge Retention practices. The qualitative and quantitative data collection will be conducted in such a way that both types of data are represented equally in the sample size of the survey questionnaire. The two types of data collection will be analysed separately by using side-by-side comparison. The results of both analyses will be merged and reported collectively. The design of the survey questionnaire will ensure that same concepts are measured in collection of both qualitative and quantitative data by removing any inconsistencies. The responses on qualitative and quantitative data will be collected simultaneously through survey questionnaire from each participant

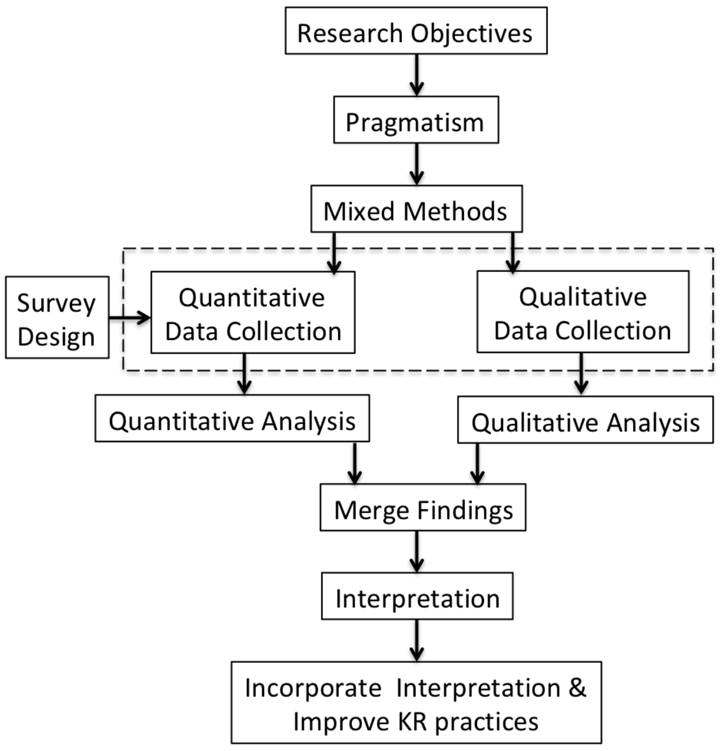

Fig. 2. The Research Methodology to investigate proactive knowledge retention in OSS projects

There are three key validity concerns while using the mixed method research. The first one is of unequal sample size when qualitative and quantitative data is collected [43]. Generally quantitative sample size is larger than qualitative sample size to perform statistical tests. The sample size for a qualitative data collection is small since the intension is to study the sample extensively and gain an in-depth perspective. The second concern is that it can be problematic to compare the findings from two data types with different variables and merge them will lead to incorrect strategy for inquiry [43]. In case of divergence, it can be difficult to explore the results any further. The third concern is to have the same participants in qualitative and quantitative data collection for a better comparison [43].

4 DATA COLLECTION AND ANALYSIS

In order to identify knowledge retention practices, the research proposes the selection of data components is from three data sources: (1) Knowledge retention practices employed in traditional organisations, (2) Knowledge loss mitigation techniques in OSS, and (3) open source guides online. The selection of data components knowledge retention practices from traditional organisation have been identified from the literature. Identification of mitigation techniques for Knowledge Loss is from OSS literature. Additionally, "OSS guides" were identified from online resources, which aggregates best practices from OSS community, and others.

Knowledge retention can be seen as a way of embedding and enabling knowledge within an organisation and a critical factor for sustainable performance [56]. The Knowledge retention strategies used in an organisation advance innovation, organisational growth, efficiency, employee development, and competitive advantage [57]. There are many techniques that facilitate knowledge capture, sharing, and reapplication namely after-action reviews, communities of practice, face-to-face meetings, conferencing, interviews, written reports, use of training or technology-based systems to transfer the knowledge [58] and job rotations [59].

The examination of the literature on knowledge loss in OSS projects resulted in identification of practices on mitigation of knowledge loss in OSS projects [40]. As one of the data sources, the knowledge retention mitigation techniques gathered from the literature review in OSS projects serve as the selected data components in the development of knowledge retention model. Such a model would be comprised of all the major knowledge related practices that OSS projects could adopt if seeking to improve the knowledge exchange among the project community. At the present time, out literature review confirms that there is a substantial – if disparate – body of research examining OSS projects and (separately) knowledge retention in software engineering. Our proposal to introduce a unified model would have clear benefits OSS projects as it would be a single touchpoint for all the acknowledged proactive knowledge retention practices proposed up to this time. In undertaking this work, we expect to systematically distil a model by applying constant comparison and memoing in an effort to robustly and reliably derive an understanding of the precise meaning and relationship between the various factors identified. We furthermore intend to later elicit feedback from the OSS community concerning the structure and content of the proposed model, ultimately leading to a more rounded and acceptable general model for all OSS practitioners.

Open Source Guides have been created and curated by GitHub along with input from outside community reviewers, but they are not exclusive to GitHub products [60]. The goal of open source guide to aggregate community best practices, not what GitHub (or any other individual or entity) thinks is best with the intention to accumulate enough resources for people creating open source projects [60]. Open source guides focus on six aspects in open source community including how to contribute to open source, starting an open source project, finding users for your project, building welcoming communities, best practices for maintainers, and leadership and governance. The review and tagging technique thoroughly applies to the OSS guides online. Consequently, key practices suitable for knowledge retention to benefit the OSS projects and make projects successful were highlighted.

The data components as described above will be extracted from three data sources while keeping a track to original sources. The knowledge retention practices in traditional organisation from presented in literature will be identified and analysed. The mitigation techniques for knowledge loss in OSS will also be identified in an extensive literature. The resulting textual descriptions will be collated in a master repository. The details of practices gathered in the master repository require a systematic analysis and identification into a set of knowledge retention strategies. This will be achieved via through Qualitative Data Analysis.

Qualitative data analysis tends to be inductive in nature and the qualitative data analyst identifies important categories in the data, as well as patterns and relationships, through a process of discovery [61]. Grounded theory principles are applied effectively to conduct qualitative data analysis and identify knowledge retention strategies with rigour. We are proposing five different stages underpinning analysis of selected data components consisting of removing conceptual duplicates, primary classification, categorising classification, revisit change and renaming, and reviews. Central to analysis using grounded theory principles are coding, memoing, and constant comparison. Coding refers to the process of breaking down, conceptualising, and re-assembling data [62]. It is a form of content analysis that is used to find and conceptualise the underlying issues amongst the noise of the data [63]. In the memo writing process, the researcher transforms data into theory and analytically interprets data [64]. Memos are not intended to describe the social worlds of the researcher's data, instead they conceptualize the data in narrative form [64]. In Grounded Theory, constant comparison refers to “the analytic process of comparing different pieces of data for similarities and differences” [64]. This method of analysis “generates successively more abstract concepts and theories through inductive processes of comparing data with data, data with category, category with category, and category with concept” [65].

5 RESEARCH LIMITATIONS

There are some limitations in our research that we wish to explicitly identify and discuss. Any effort towards a knowledge retention model is beset with certain unavoidable constraints: have all of the possible factors of interest been identified, have they been appropriately interrelated, and is the naming and classification system employed suitable. These are constraints that are not easily assuaged. We nonetheless have invested significant effort in attempting to reduce the impact of these concerns to within tolerable limits.

In compiling our list of input sources and data, a substantial and methodology-driven approach has sought to exhaust all published peer-reviewed resources dealing with the subject of knowledge in software projects. This has involved the identification and examination of >1000 individual sources and used a tried and tested snowballing approach to iterate to possible additional sources of information throughout the literature review phase. This is by design and the purpose if to avoid missing some information that might be relevant to the model construction. It is nevertheless important to acknowledge that there are limitations in the snowballing approach, for example if only very few and low-quality papers were selected as the starting point for the snowballing process, then there would be a genuine risk concerning the coverage that the literature review might produce. Similarly, if the focus of the initial research papers was off-topic to some degree then the subsequent literature produced by the snowballing process might also be off topic to some degree. Both of these snowballing limitations could have affected our review also, but we highlight that the key search terms adopted and the extensive nature of our literature review should work to reduce the impact of these limitations to a significant extent.

Having generated, distilled and classified the factors related to knowledge retention, the model will be the subject of intensive internal scrutiny, wherein the name, description and classification of all model content will be carefully examined by a review team, iterating as many times as is required in order to reach general agreement on the content of the model. This step will help to ensure that the naming, associations and classifications in the model will be in the best condition possible following the initial construction of the reference model.

We will finally subject the draft knowledge retention model to external evaluation, using a group of expert practitioners to quality the model not just for utility, but also for understandability and comprehensive scope. Together, we expect that these steps will result in a model that is robust and accurate, but we nevertheless stress that a basic limitation in work of this nature is that general agreement in the population as an entirety is not possible and therefore some debate on the content, structure and effectiveness will invariably arise in the future. The survey instrument itself also introduces some limitations: specifically in terms of ensuring that it is written and structured in a manner that is conducive to clear interpretation, and ultimately, no survey instrument is absolutely perfect, so some issues can arise in this respect. However, our process will seek to reduce the impact of such concerns, for example by insisting on multiple formal internal project reviews of the survey instrument prior to discharge and through the use of piloting to check that used can utilise the survey instrument in a consistent and reliable manner.

6 DISCUSSION & FUTURE WORK

OSS projects are dynamic, dispersed, with transient contributors concurrently performing tasks in different roles, and collaborating through technologically mediated channels. The challenge is to formulate the strategy for proactive knowledge retention, which can resonate with the idiosyncratic nature of OSS projects without causing an overhead to the productivity of the project and contributors. The OSS communities focus on self-direction and favour intrinsic value system and therefore imposing a strategy that calls for contributors to limit their freedom will lower their enthusiasm of doing well for the society.

The main goal of this research is to develop an overarching proactive knowledge retention strategy in OSS projects, something that we suggest is warranted based on a review of the literature and on the volatile nature of the OSS workforce. The mixed method research described in this paper establishes the baseline of the research methodology for the investigation of proactive knowledge retention in OSS projects. Moving onto the next phase to implement the research methodology, we will shift our focus to survey design. The research objectives will underpin the survey design and will align with the data collection in order to accomplish the research goals. In order to design the survey, selection of appropriate data components is required to develop a questionnaire and gather valuable information from OSS communities. A streamlined process is required to identify the data components with a close examination of known and unknown information leading to the formation of an elaborative set of knowledge retention practices suitable for OSS projects. The analysis of the data collected through the anticipated surveys will facilitate in assessment, refinement and improvement of the knowledge retention practices in OSS projects based on the feedback from OSS community.

Moving onto the next phase to implement the research methodology, the focus is on a systematic approach to validate and evaluate the identified knowledge retention strategies. The survey questionnaire will be utilised to gather valuable information from OSS communities. The analysis of the data collected through the anticipated surveys will facilitate in assessment, refinement, and improvement of the knowledge retention practices in OSS projects based on the feedback from OSS community.

The objective of this research is to improve proactive knowledge management as a counter-balance to the transient and sometimes uncommitted nature of OSS personnel. We suggest that the research methodology identified herein is appropriate to this research and that it will deliver tangible benefits for both contributors to OSS projects and consumers of OSS. We also encourage further focus on knowledge retention in general in software development projects and believe that our ultimate work products can be helpful in this respect. Consider for example the current drive towards Continuous Software Engineering (CSE) [66] and its implications in terms of knowledge. Perhaps in these settings some of the knowledge will be tied up in automated scripts that are executed in response to some change in the code base (e.g. a code commit hook). What of the knowledge used to create that script once the creator should depart the organisations? Therefore, we see that our work towards a knowledge retention model for knowledge retention in OSS projects might in the future be adapted for use in more generalised software development settings.

In earlier work the authors have examined the needs of individual situational contexts [67, 68] and it could be that an important situational factor relates to the need for knowledge identification, management and transfer in companies. We suggest that the degree of knowledge exchange should be an important piece of information, and techniques that could help to visualise or understand the desirable amount of knowledge transfer in software settings in general would make for an interesting and useful area for future research; especially as many software companies struggle to attract and secure software development expertise that is in limited supply.

REFERENCES

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

[2] N. Anquetil, K. de Oliveira, M., K. de Sousa, D., and M. Batista Dias, G., "Software maintenance seen as a knowledge management issue," *Inf. Softw. Technol.,* vol. 49, pp. 515-529, 2007.

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

[4] J. Feller and B. Fitzgerald, *Understanding open source software development*: Addison-Wesley London, 2002.

[5] Clarke, P., O'Connor, R.V., Leavy, B., Yilmaz, M.: Exploring the Relationship between Software Process Adaptive Capability and Organisational Performance. IEEE Transactions on Software Engineering, 41(12), pp.1169-1183, doi: 10.1109/TSE.2015.2467388 (2015)

[6] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: theory and measures," *Software Process: Improvement and Practice,* vol. 11, pp. 123-148, 2006.

[7] M. Silic and A. Back, "Open Source Software Adoption: Lessons from Linux in Munich," *IT Professional,* vol. 19, pp. 42-47, 2017.

[8] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/Libre Open-Source Software Development: What We Know and What We Do Not Know," *Acm Computing Surveys,* vol. 44, Feb 2012.

[9] S. BlackDuck. (2015, N.p., 2017. Web. 8 June 2017). *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

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

[11] Yu, Y., Benlian, A., Hess, T.: An empirical study of volunteer members' perceived turnover in open source software projects. In: System Science (HICSS), 2012 45th Hawaii International Conference on, pp. 3396-3405. IEEE, (2012)

[12] Hertel, G., Niedner, S. and Herrmann, S., 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research policy*, *32*(7), pp.1159-1177.

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

[14] Crowston, K., Annabi, H., Howison, J., Masango, C.: Effective work practices for software engineering: free/libre open source software development. Proceedings: ACM workshop on Interdisciplinary software engineering research, pp. 18-26. ACM, USA (2004)

[15] Crowston, K., Howison, J.: The social structure of free and open source software development. First Monday 2, (2005)

[16] Rigby, P.C., Zhu, Y.C., Donadelli, S.M., Mockus, A.: Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a project at Avaya. In: Proceedings of the 2016 International Conference on Software Engineering. (2016)

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

[18] Zack, M.H.: Managing Codified Knowledge. Sloan Management Review 40, 45-58 (1999)

[19]. Nonaka, I., Toyama, R., Konno, N.: SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planning 33, 5-34 (2000)

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

[21] Clarke, P. and O'Connor, R.V., 2012. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, *54*(5), pp.433-447.

[22] S. Nerur, V. Balijepally, Theoretical reflections on agile development methodologies, Commun. ACM, 50 (2007) 79–83.

[23] H. Sharp, H. Robinson, Three 'Cs' of agile practice, in: T. Dingsoyr et al. (Eds.), Agile Software Development: Current Research and Future Directions, Springer, Berlin, 2010, pp. 61–85.

[24] DeMarco, T., Lister, T., 1987. Peopleware: Productive Projects and Teams, Dorset House, New York.

[25] Drucker, P., 1993. Post-Capitalist Society, Harper Business, New York.

[26] Faraj, S., Sproull, L., 2000. Coordinating expertise in software development teams, Management Science, 46 (12), 1554-1568.

[27] Foucault, M., Palyart, M., Blanc, X., Murphy, G.C., Falleri, J.-R.: Impact of developer turnover on quality in open-source software. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 829-841. ACM, Bergamo, Italy (2015)

[28] Robles, G., Gonzalez-Barahona, J.M.: Contributor turnover in libre software projects. IFIP International Federation for Information Processing, vol. 203, pp. 273-286 (2006).

[29] Basri, S. and O'Connor, R., 2012. A study of knowledge management process practices in very small software companies. *American Journal of Economics and Business Administration*, *3*(4), pp.636-644.

[30] von Krogh, G., Spaeth, S., Haefliger, S.: Knowledge reuse in open source software: An exploratory study of 15 open source projects. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pp. 198b-198b. IEEE, (2005)

[31] Herraiz, I., Robles, G., Amor, J.J., Romera, T., González Barahona, J.M.: The processes of joining in global distributed software projects. Proceedings of the 2006 international workshop on Global software development for the practitioner, pp. 27-33. ACM, Shanghai, China (2006)

[32] Zhou, M., Mockus, A.: Developer fluency: achieving true mastery in software projects. Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, pp. 137-146. ACM, Santa Fe, New Mexico, USA (2010).

[33] Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions onSoft. Eng. 29, 481-494 (2003)

[34] Izquierdo-Cortazar, D., Robles, G., Ortega, F., Gonzalez-Barahona, J.M.: Using software archaeology to measure knowledge loss in software projects due to developer turnover. In: System Sciences, 42nd Hawaii International Conference on, pp. 1-10. IEEE, (2009)

[35] Liu, W., Chen, C.L., Lakshminarayanan, V., Perry, D.E.: A design for evidence - based soft research. SIGSOFT Softw. Eng. Notes 30, 1-7 (2005)

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

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

[38] Doan, Q.M., Rosenthal-Sabroux, C., Grundstein, M.: A Reference Model for Knowledge Retention within Small and Medium-sized Enterprises. In: KMIS, pp. 306-311. (2011)

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

[40] Rashid, M., Clarke, P.M. and O'Connor, R.V., 2019. A systematic examination of knowledge loss in open source software projects. *International Journal of Information Management*, *46*, pp.104-123.

[41] Rashid, M., Clarke, P.M. and O'Connor, R.V., 2018, September. An Approach to Investigating Proactive Knowledge Retention in OSS Communities. In European Conference on Software Process Improvement (pp. 108-119). Springer, Cham..

[42] Creswell, J.W., Clark, V.L.P.: Designing and Conducting Mixed Methods Research. SAGE Publications (2011)

[43] Creswell, J.W.: Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. SAGE Publications (2014)

[44] Lincoln, Y.S., Lynham, S.A., Guba, E.G.: Paradigmatic controversies, contradictions, and emerging confluences, revisited. The Sage handbook of qualitative research 4, 97-128 (2011)

[45] Rubin, H.J., Rubin, I.S.: Qualitative interviewing: The art of hearing data. Sage (2011)

[46] Denzin, N.K.: The research act: A theoretical introduction to sociological methods. Transaction publishers (1973)

[47] Klein, H.K., Myers, M.D.: A set of principles for conducting and evaluating interpretive field studies in information systems. MIS quarterly 67-93 (1999)

[48] Johnson, R.B., Onwuegbuzie, A.J.: Mixed Methods Research: A Research Paradigm Whose Time Has Come. Educational Researcher 33, 14-26 (2004)

[49] Tashakkori, A., Teddlie, C.: Sage handbook of mixed methods in social & behavioural research. Sage (2010)

[50] Wohlin, C., Höst, M., Henningsson, K.: Empirical Research Methods in Software Engineering. In: Conradi, R., Wang, A. (eds.) Empirical Methods and Studies in Software Engineering, vol. 2765, pp. 7-23. Springer Berlin Heidelberg (2003)

[51] Shull, F., Singer, J., Sjøberg, D.I.: Guide to advanced empirical software engineering. Springer (2008)

[52] Mackenzie, N., Knipe, S.: Research dilemmas: Paradigms, methods and methodology. Issues in educational research 16, 193-205 (2006)

[53] Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir Software Eng 14, 131-164 (2009)

[54] Denzin, N.K., Lincoln, Y.S.: The Sage handbook of qualitative research. Sage (2011)

[55] Easterbrook, S., Singer, J., Storey, M.-A., Damian, D.: Selecting Empirical Methods for Software Engineering Research. In: Shull, F., Singer, J., Sjøberg, D.K. (eds.) Guide to Advanced Empirical Software Engineering, pp. 285-311. Springer London (2008)

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

[57] T. J. Delong, *Reexamining the career anchor model*, 1982.

[58] D. W. De Long and T. Davenport, "Better practices for retaining organizational knowledge: Lessons from the leading edge," *Employment Relations Today*, vol. 30, pp. 51-63, 2003.

[59] A. G. J. Jansen, *Architectural design decisions*. PhD thesis, Rijks Universiteit Groningen. ISBN: 978-90-367-3494-3, 2008.

[60] N. Eghbal, B. Keepers, S. Wills, and M. Linksvayer. (2018, 15 October 2018). *Open Source Guides*. Available: https://opensource.guide/

[61] R. K. Schutt, *Investigating the social world: The process and practice of research*: Pine Forge Press, 2011.

[62] M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, pp. 3-21, 1990.

[63] O'Connor, R., 2012. Using grounded theory coding mechanisms to analyze case study and focus group data in the context of software process research. In *Research methodologies, innovations and philosophies in software systems engineering and information systems* (pp. 256-270). IGI Global.

[64] L. B. Lempert, *Asking questions of the data: Memo writing in the grounded theory tradition*, 2007.

[65] A. Bryant and K. Charmaz, *The Sage handbook of grounded theory*: Sage, 2007.

[66] O'Connor, R.V., Elger, P., Clarke, P.: Continuous Software Engineering - A Microservices Architecture Perspective. Journal of Software: Evolution and Process, 29(11), 2017, pp.1-12.

[67] Marks, G., O'Connor, R.V., Clarke, P.: The impact of situational context on the software development process - A case study of a highly innovative start-up organization. In: Proceedings of the 17th International SPICE Conference (SPICE 2017), pp.455-466; 4-5 October 2017, Palma de Mallorca, Spain.

[68] Clarke, P., O'Connor, R.V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., Gerrity, M., McGrath, S., Treanor, R.: Exploring Software Process Variation Arising from Differences in Situational Context. In: Proceedings of the 24th European and Asian Conference on Systems, Software and Services Process Improvement (EuroSPI 2017), pp.29-42, 5-8 September 2017, Ostrava, Czech Republic.;