An Investigation into Inner Source Software Development: Preliminary Findings from a Systematic Literature Review

Henry Edison
Lero, National University of Ireland Galway
henry.edison@nuigalway.ie

Noel Carroll
Lero, National University of Ireland Galway
noel.carroll@nuigalway.ie

Kieran Conboy
Lero, National University of Ireland Galway
kieran.conboy@nuigalway.ie

Lorraine Morgan
Lero, National University of Ireland Galway
lorraine.morgan@nuigalway.ie

ABSTRACT

Given the value and effectiveness of open source software development to date, practitioners are keen to replicate these practices inside their respective corporations. This application of open source practices inside the confines of a corporate entity has been coined inner source software development. However, while organisations have found ways to directly benefit from revenue streams as a result of leveraging open source practices internally, the current research on inner source is scattered among different areas. Thus gaining clarity on the state-of-the-art in inner source research is challenging. In particular, there is no systematic literature review of known research to date on inner source. We address this challenge by presenting a systematic literature review that identifies, critically evaluates and integrates the findings of 29 primary studies on inner source. Case study approach is the common research approach undertaken in the area. We also identified 8 frameworks/methods, models and tools proposed in the literature to support inner source, as well as a set of benefits and challenges associated with inner source. We envision future work to perform deeper analysis and synthesis on the empirical research on inner source software development.

ACM Classification Keywords

D.2.9 Software Engineering: Management – Software process models; K.6.3 Management of Computing and Information Systems: Software Management – Software development, Software maintenance, Software process

Author Keywords

inner source software development, inner source, systematic literature review, open source

OpenSym '18, August 22–24, 2018, Paris, France

© 2018 ACM. ISBN 978-1-4503-5936-8/18/08…$15.00

DOI: https://doi.org/10.1145/3233391.3233529

INTRODUCTION

Open Source Software (OSS) has been highly prevalent in both practice and research. Given the value and effectiveness of open source development to date, practitioners are keen to replicate these practices inside their respective corporations. This application of open source practices inside the confines of a corporate entity has been coined inner source software (ISS) development [7, 49, 16, 41]. ISS development has been seen as a way in which organisations adopt OSS [17, 19]. Leading organisations such as Lucent Technologies, Nokia, Philips, IBM and HP etc, have provided example of inner source implementations [5], which has led to the emergence of studies on inner source in the scientific literature (e.g. [23, 24, 42, 41, 43]).

While the literature purports that ISS development presents many benefits for internal use, there have been growing research efforts to equally highlight the challenges associated with adopting and scaling inner source practices. However, understanding the state-of-the-art in inner source research is challenging. Capraro and Riehle [5] recently published a review on inner source literature. Although the findings are significant and shed light on the grey area of ISS development, the inner source research field remains dispersed among different research areas. Thus, the objective of this study is to understand the current research that has been carried out on the usage of ISS development or the application of OSS principles within existing organisations. This objective is broken down into the following research questions: *(i) RQ1 - What research methods have been used in studies on ISS development? (ii) RQ2 - What types of contributions are provided by the studies on ISS development? and (iii) RQ3 - What are the reported benefits and challenges associated with ISS development?*

The remainder of this article is structured as follows. Section 2 presents a brief discussion of related work in this area and followed by a presentation of our research approach in Section 3. In Section 4, we present an overview of the characteristics of the primary studies. The key findings of this study are presented in Section 5, and discussed in Section 6. Finally, the conclusion is presented in Section 7.

RELATED WORK

We identified four secondary studies [17, 19, 6, 5] on inner source that were considered relevant to this study. For example, Hauge et al. [17] and Höst and Oručević-Alagić [19] performed literature reviews by following the guidelines provided by Kitchenham and Charters [21]. The study by Crowston et al. [6] could be considered partly systematic, since the relevance can be seen in terms of search strategy, data sources, inclusion/exclusion criteria and data extraction. On the other hand, the study by Capraro and Riehle [5] presented an extensive literature review but showed no evidence of following any systematic guidelines.

Both Hauge et al. [17] and Crowston et al. [6] conducted a search on specific journals or conferences on open source, while Höst and Oručević-Alagić [19] and Capraro and Riehle [5] only focus on software engineering related databases such as Inspec, Compendex, ACM Digital Library and IEEE Xplore. Furthermore, Hauge et al. [17] did not report the used search string and Capraro and Riehle [5] did not reveal the timespan for the search.

Neither Crowston et al. [6] nor Capraro and Riehle [5] used explicit criteria for quality assessment of the primary studies, which hinders interpretation. Contrary to these studies, Hauge et al. [17] and Höst and Oručević-Alagić [19] adopt the quality assessment criteria developed by Dybå and Dingsøyr [8]. For our review, we adopted a comprehensive set of evaluation guidelines based on scientific rigour and industrial relevance proposed by Ivarsson and Gorschek [20].

The aim of the review conducted by Hauge et al. [17] and Höst and Oručević-Alagić[19] was to assess the current research on how organisations adopt OSS, while Crowston et al. [6] investigated the state of research on open source in general. These studies found that there is a lack of studies on open source software practices inside organisations. Capraro and Riehle [5] developed a model of the elements that constitute inner source. The study also presents a classification framework for inner source programs and projects, and a map of known inner source endeavours, as well as qualitative models summarising the benefits and challenges of inner source adoption. All studies identify that there is a trend towards organisational adoption of open source principles in their internal development processes.

In summary, among the existing relevant literature reviews, the study by Capraro and Riehle [5] is only one study that is closer in similarity to our study. One notable difference between the study by Capraro and Riehle [5] and our study is the systematic approach we adopted for our research methodology and the coverage of the search. The study by Capraro and Riehle is more of a literature survey than a systematic literature review (SLR). In addition, our study is much broader since we consider both software engineering (SE) and information systems (IS) literature.

RESEARCH METHODOLOGY

To answer our research questions, we systematically assessed existing evidence related to ISS development using SLR guidelines [21]. An SLR facilitates in identifying and collecting key papers in a specific area of interest, and evaluating and interpreting the reporting discussions and findings [21]. A defined review protocol, search strategy, explicit inclusion and exclusion criteria, and specified information that will be retrieved from primary studies differentiates a systematic review from a conventional literature review [21].

Search Strategy

To help build the search terms, a set of key papers were identified by all authors. Keywords used in these papers were extracted and aggregated and used as the input for the search terms. From the key papers, we extracted a total of 76 keywords (30 unique). The top three keywords are "open source" (23.68%) and "software development" (19.74%), and "inner source" (10.53%).

The search terms were organised into three groups: intervention, control and population, and separated by `AND`-clauses. Table 1 describes the generic search strings. The developed search terms were then searched through in two digital libraries (EngineeringVillage and Scopus) to view its effectiveness. Revisions to the search terms were made until we were able to retrieve all the key papers from these two digital libraries.

A search for relevant literature was conducted on the title, abstract and keyword. We decided to use digital libraries that have good coverage, familiarity, reputation, advanced features and exportability [10]. The digital libraries used to search for relevant literature were relevant to (i) SE research: Compendex, ISI Web of Science and Scopus [9, 3], and (ii) IS research: AIS e-Library. The actual search was performed in August 2017. We collected and analysed articles that have been published until August 2017.

Group	Terms
Intervention (key concepts)	`{inner?source} OR innersource OR {open?source} OR opensource`
Control (context)	`software`
Population (scope)	`organi?ation OR firm OR company OR corporat* OR enterprise OR industr* OR vendor OR institution* OR internal OR inside OR hybrid`

Table 1. Search terms organisation

Selection Strategy

Studies were eligible for inclusion in this study if they are (i) peer-reviewed papers, (ii) written in English, (iii) full-text available, and (iv) focusing on ISS development or the adoption of OSS principles within an organisation. Both theoretical and empirical studies were included in the review process. Similarly, studies conducted in both academic and industry settings were included.

The removal of irrelevant studies and duplicates across digital libraries was conducted as a pre-screening process in the literature review. This was then followed by applying inclusion and exclusion criteria, which was done in two stages. In the first stage, the criteria were applied on the title and abstract level of the papers. In the second stage, the inclusion and exclusion criteria were applied on the full-text of the remaining papers. In every stage, two independent reviewers evaluated the same paper. To be included in the next phase, two reviewers had to be in agreement that a paper met the inclusion criteria. In the cases where both reviewers did not agree on the decision, a third reviewer was called. Full-text papers that met the inclusion and exclusion criteria were then assessed on their quality.

Quality Assessment

In a SLR, study quality assessment was essential to evaluate the existing research topic by using a trustworthy, rigorous and auditable methodology [21]. Hence, the quality assessment was performed independently and the results were not used to decide whether to include or exclude a particular study.

The quality of the primary studies was evaluated based on their scientific rigour and industrial relevance criteria as presented in Ivarsson and Gorschek [20]. We also devised a similar rubric for assessing the rigour of the philosophical papers. Scientific rigour was evaluated by using the following aspects: (i) study context: whether a reviewer can understand and compare it to another context, (ii) study design: whether a reviewer can understand how rigour the research method is applied in the study or to which the theoretical contribution used sound theoretical bases to guarantee the quality of the research, and (iii) validity discussion: to what extend the threats of the study or limitation of the theoretical approach are described and measures to limit them are detailed.

For the industrial relevance, we used the same rubric for both empirical and philosophical studies. Relevance was assessed using the following aspects: (i) subject: whether the subjects in the study were representative of inner source practitioners e.g. students or practitioners, (ii) context: whether the study was conducted in the representative industry setting, (iii) scale: whether the size of the study was realistic or based on a toy example, and (iv) research method: whether the research method employed in the study facilitates investigating real situations and relevant for practitioners. The grading for each aspect was done on three point scale: yes (weighing 1 point - indicating that data for specific aspect is clearly available), somewhat (weighing 0.5 point - indicating that data is vaguely available) and no (weighing 0 point - indicating that data is unavailable).

Data Extraction and Synthesis

To help answer our research questions, data extraction was carried out guided by an extraction form implemented in MS Excel. The following aspects were extracted from the primary studies: (i) type of the studies, classified as: empirical research, experience report, philosophical, opinion, (ii) contributions of the studies: model, theory, framework or method, guidelines, lessons learned, advice or implication, and tools [34], and (iii) research method, including data collection and analysis method, and theoretical lens.

The data extracted from each primary study were integrated in categories representing the research topic addressed, the research method used, the contributions of the study and the benefits and challenges of ISS reported. Frequencies of each component in the categories were recorded. The results were then presented and discussed with the other coauthors.

CHARACTERISTICS OF PRIMARY STUDIES

Search Results

From all digital libraries, we retrieved a total of 13,178 articles. By applying the inclusion/exclusion criteria, we accepted 29 articles as primary studies. The 29 primary studies are listed in Table 2 and referred using their IDs throughout the rest of the paper.

ID	Author(s)	ID	Author(s)
PS1	Alexy et al. [1]	PS16	Pulkkinen et al. [35]
PS2	Dinkelacker et al. [7]	PS17	Riehle et al. [36]
PS3	Gaughan et al. [12]	PS18	Riehle et al. [37]
PS4	Grottke et al. [14]	PS19	Ripatti et al. [38]
PS5	Gurbani et al. [15]	PS20	Sharma et al. [40]
PS6	Gurbani et al. [16]	PS21	Stol et al. [41]
PS7	Hauge et al. [18]	PS22	Stol et al. [42]
PS8	Linåker et al. [25]	PS23	Stol and Fitzgerald [43]
PS9	Lindman et al. [23]	PS24	Theunissen et al. [44]
PS10	Lindman et al. [24]	PS25	Torkar et al. [45]
PS11	Martin and Lippold [26]	PS26	Linden et al. [47]
PS12	Melian and Mähring [27]	PS27	Linden [46]
PS13	Morgan et al. [29]	PS28	Vitharana et al. [48]
PS14	Neus and Scherf [30]	PS29	Wesselius [49]
PS15	Oručević-Alagić and Höst [33]		

Table 2. List of Primary Studies

Publication Sources and Years

The distribution of the years and venues of the primary studies is shown in Fig 1. While the term inner source was introduced in the year 2000 [31], the first research paper in this topic was published two years later (PS2). 14 out of 29 papers (48%) are published in journals i.e. Research Policy, IST, JAIS, TOSEM, etc., whereas 15 papers (52%) are presented in various conferences in both SE and IS, i.e. OSS Symposium, ECIS, PROFES, etc.

Quality of the Primary Studies

Based on the rigour and relevance scores, the primary studies can be considered to be of good quality. The percentile rankings of the quality scores are shown in Fig. 2. The maximum score that a paper could get was 7. Studies with scores below the lower quartile lacked clear information about the study design and threats to validity, as required in the rigour rubric. Typically, these studies were published in practitioners-oriented journals e.g. IEEE Software (PS23, PS26, PS29) or Communication of the ACM (PS6). Studies with minimum scores are philosophical papers. Since the identified or proposed framework or methods have not been studied empirically in industry settings, the relevance scores were zero. Moreover, most of the studies within the interquartile range

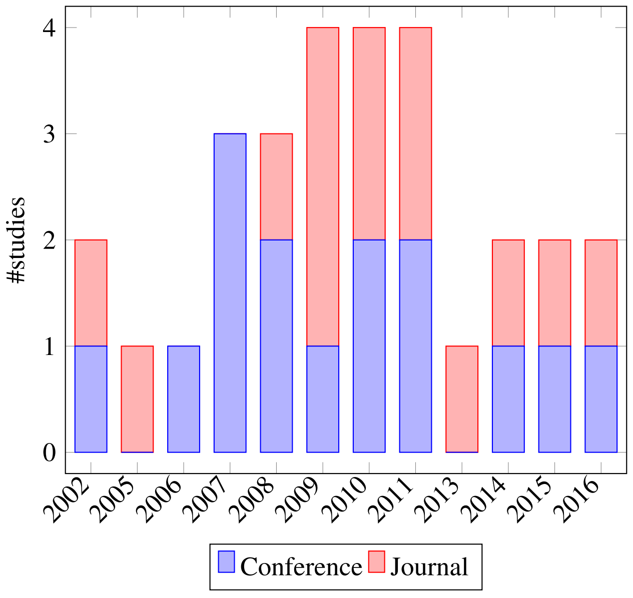

Figure 1. Temporal Distribution of the Primary Studies

did not discuss validity threats and how they were mitigated, which negatively affected the trustworthiness of the reported findings [39].

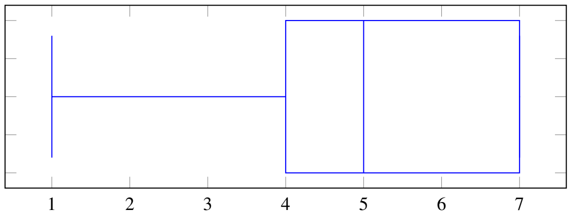

Figure 2. Percentile rankings of the quality scores

FINDINGS

In this section, we present an overview of the body of inner source literature that originated from our literature review. We structure this section according to our research questions.

RQ1 What research methods have been used in studies related to ISS development?

Out of 29 primary studies, 23 papers are empirical research papers. Case study approach was the main common research approach undertaken to investigate inner source approach (13 studies), whilst 2 studies (PS1 and PS7) were employing mixed method, and one study used design science approach (PS19). Some experience report papers were submitted to a special industry experience track of particular conference (PS5, PS11, PS19).

Our primary studies shows that most research on inner source approach has been conducted in established and large multinational organisations, and only one study in an SMEs context (PS7). These organisations come from various business domains, e.g. engineering, software development, medical equipment or telecommunication. They typically have a long history with proprietary oriented and industrial/commercial mode of software development and gradually incorporated open source software and methods into their internal development (PS30). The overview of the empirical research papers is illustrated in Table 3.

Interview and coding technique were the main data collection and analysis method used in the existing research on ISS. The total number of interviews in each studies were ranged between 5 (PS7) and 32 (PS21). The participants involved in interview were typically employees with engineering backgrounds and roles, i.e. software developer, architect, programmer and administrative role, e.g. manager. The working nature of these two roles are also highly affected by the decision of the organisation to adopt inner source approach (PS1).

Out of 13 case study papers, only 6 studies used a theory or conceptual framework as a theoretical lens. Conceptual framework serves as a guide to explain the primary objects of a research project e.g. key factors, constructs or variable and the relationship among them [28]. It is a sensitising and sensemaking device that guides the data collection and analysis processes. Three of them were published in IS conferences (ECIS - PS10, PS13) and journal (JAIS - PS25). In contrast to SE research, the use of theory in IS research is critical [22]. For example, Orlikowski and Lacono [32] argued that IS research is under theorised. Hence, IS researchers aim for strong theoretical contributions and ground their work in theory [22].

RQ2 What types of contributions are provided by the studies on ISS development?

Fig. 3 shows the distribution of the contributions of primary studies. The main contribution of the primary studies was in the form of theory (15 studies) around an ISS approach. Nonetheless, 27% (8 studies) of the contributions provided concrete approaches that could be used to support an ISS approach. These approaches included framework or method for implementing an ISS approach (5 studies), e.g. progressive open source (PS2), corporate open source (PS6), models representing relevant concepts of inner source (2 studies) e.g. theoretical model to promote software reuse (PS28), and inner source business model (PS29) and tool supporting technical infrastructure of inner source (PS19). However, study PS8 concerned about the framework used in the study, which has not been evaluated by others than the authors. Thus, they called for more studies to evaluate the existing frameworks, methods and tools in a different context so that they can be generalised. Other studies provided guidelines (2 studies), lesson learned (3 studies) and advice (1 study) for implementing an ISS approach. The list of the frameworks/methods, models and tools are described in Table 4.

For a successful technology transfer, academic research results are required to be validated statically or dynamically in real setting [13]. Static validation involves presentation of the solution in industry and collecting feedback from practitioners, whilst dynamic validation includes piloting the solution in a real development setting. The purpose of the validation is not to sell the framework or model but to gather valuable feedback regarding usability and scalability of the frameworks/models

ID	Research Approach	Data Collection Method	Data Analysis Method	Theoretical Lens	Context of study
PS1	Mixed method	Interview & survey	Content analysis	–	A multinational engineering firm
PS2	Case study	Not-specified	Not-specified	–	HP
PS3	Interview	Interview	Coding technique	–	7 large multinational companies
PS4	Case study	Data archival	Log transformation	–	A commercial software vendor and Sourceforge.net
PS6	Case study	Not-specified	Not-specified	–	Lucent Technologies
PS7	Mixed method	Interview & survey	Not-specified	–	3 SMEs
PS8	Case study	Data archival & interview	Archival analysis	Inner source framework [41]	An international software development firm
PS9	Case study	Interview	Not-specified	–	Nokia
PS10	Case study	Interview	Coding technique	Institutional entrepreneurship	Philips and Nokia
PS12	Case study	Interview	Coding technique	Actor-Network Theory (ANT)	HP
PS13	Case study	Interview	Coding technique	Open innovation theory [11]	Large medical equipment supplier
PS15	Case study	Focus group	Coding technique	–	Global market leader in software and hardware
PS16	Case study	Participatory observation	Not-specified	Knowledge life-cycle and human aspect of KM	Nokia
PS17	Case study	Interview	Coding technique	–	Three international software companies
PS19	Design science	Prototyping	Not-specified	–	Insta DefSec Ltd.
PS21	Case study	Interview	Coding technique	–	3 large multinational companies
PS22	Case study	Interview	Coding technique	–	A large, globally distributed organisation
PS23	Case study	Interview	Not-specified	–	Philips Healthcare, Lucent
PS25	Case study	Interview	Thematic analysis	Avison and Fitzgerald [2] framework	A large telecommunication company
PS26	Case study	Not-specified	Not-specified	–	2 large European companies
PS28	Case study	Interview	Not-specified	–	IBM
PS29	Case study	Not-specified	Not-specified	–	Philips Healthcare

Table 3. Overview of the empirical research papers

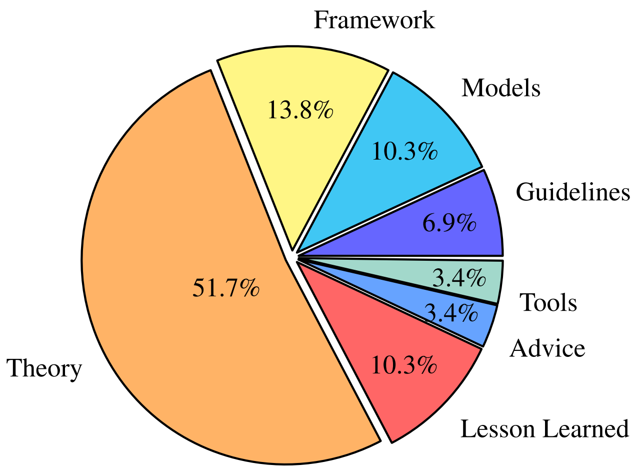

Figure 3. Contributions of the primary studies

and get commitment to implement them across the organisation [13]. Out of 8 approaches, 6 of them had been validated in industry settings (P2, P6, P19, PS21, PS25, P29), while the remaining approaches (PS20, P28) are considered theoretical frameworks.

In the studies PS2, PS6, PS29, the proposed frameworks and model have been dynamically validated in real development projects in the adopting organisations. In the study PS19, the proposed tool had been tested in a real implementation as part of the design science process. In the study P21, the framework was validated using qualitative approach where the data were collected from three multinational organisations. Study PS25 is the only study that performed static and dynamic validation. In the dynamic validation, the organisation adopted the proposed framework in a pilot project.

RQ3 What are the reported benefits and challenges associated with an ISS development?

Table 5 lists the summary of the reported benefits and challenges. We categorise them into four facets: software development, process management, tool and technology and managerial and organisation.

Software Development

Most of the benefits of an ISS approach reported in our primary studies are related to engineering practices to build software. For example, the adoption of an ISS approach enables organisations to improve software quality and reduce technical debt (PS2, PS3, PS8, PS18, PS20, PS25). Since contributors across an organisation can join the community, they significantly broaden the expertise available to a project. Furthermore, they can help fix problems more quickly; either by preventing mistakes or capturing them earlier. This in turn helps a software project reach the goals more quickly and at higher quality (PS2, PS3, PS18, PS20, PS21). The collaboration between various business units increases core team innovativeness in order to satisfy user needs (PS13). Several studies have also reported that an ISS approach reduces development effort as well as cost (PS6, PS7, PS10, PS20). Software reuse maximises the number of projects that can be shared across organisation. Moreover, development costs are shared among the projects or business units using the shared assets.

On the other hand, the common challenge of an ISS approach is related to security aspect (PS2, PS3, PS11, PS27). The openness sought after in an ISS approach makes security and access control to the internal software artefacts e.g. code or design more difficult. For example, project managers and developers are wary of random exposure to their internal product artefacts, e.g. source code and design. For developers, there is a fear of sharing or showing substandard or work. Moreover, once various contractors that are employed within different areas of the organisation are given access, there is the perception that project managers have no control over what security the vendor is using. On the other side, contractors might be reluctant to use an ISS approach due to the fear of loss of intellectual property.

Process Management

The role of contributors in an ISS approach might evolve and shift based on their personal interest. For example, they can shift from developer to maintainer or reviewer, before they will be the project leader or benevolent dictator. This predefined path allows new developer to familiarise themselves with the architecture and to perform tasks with different difficulty levels. Similar to OSS project, the community has a great deal of freedom to choose processes, methods and tools in their works (PS25).

The main challenge associated with process management however, is how to build an effective community within the organisation (PS13, PS21). The study by Stol et al. (PS21) found that there are several issues between the core team and business units in relation to their roles. The core team may be reluctant to adopt contributions from business units, due to the "not invented here" syndrome and non-generic contributions. On the other hand, business units treat the core team as a traditional component supplier. They are also often reluctant to contribute to the shared assets since they consider that development is the responsibility of the core team. The study by Morgan et al. (PS13) also found the challenges in achieving a common vision and aligning objective in an ISS environment. This study also found that even though knowledge sharing is perceived as important in facilitating value creation, it is difficult to get people to invest time or effort in sharing code or building skills and knowledge outside their own domain. The core team and business units only contribute to their areas, as there are often no incentives to contribute beyond them.

Tools and Technology

Our primary studies reveal that an ISS approach has increased information availability and visibility across organisations and defined an entry path for newcomers and new ways of working (PS25). For example, a good software forge indexes information sources, including source code assets or components and allows for centralised searching. This is important for employees with technical roles, insofar as they are able to contribute to further development.

Despite the benefits of software forges, several studies identified challenges related to the tools and technology used in an

ID	Type	Description	Validation
PS2	Framework	Progressive Open Source (POS): consists of three-tier model: (i) Inner Source – refers to the application of the OS approach and benefits to developers within the corporate environment, (ii) Controlled Source - which is outside of the corporate firewall, but restricts access limited to specific corporate partners and (iii) Open Source - refers to the open use of Internet for development, and release of the software source code in a license approved by OSI.	Dynamic
PS6	Framework	Corporate Open Source (COS): evolves in four phases: (i) Initial Development, led by the author of the code, (ii) Ad-hoc Partners, distributes the binary to a wider audience inside the company, (iii) User-initiated Change Request, expanding the class of users within the company to get feedback or wishes for new features, (iv) Establishing a COS Project, as the request for product-specific changes began to accelerate, other within the company started to contribute code and ideas.	Dynamic
PS19	Tool	OpenCart-based platform that acts as a marketplace for promoting software reuse within an organisation. The platform also provides information about the name and version of components, the technical and functional descriptions, the locations and contact persons of the components and prices and licenses if a third party component was included.	Dynamic
PS20	Framework	The framework guides the creation and management of hybrid-OSS communities in organisations, consists of three major elements: (i) community building, (ii) community governance, and (iii) community infrastructure.	–
PS21	Framework	The framework identifies nine important factors that need to be considered when implementing Inner Source. The framework can be used as a probing instrument to assess an organisation on these nine factors so as to gain an understanding of whether or not Inner Source is suitable.	Static
PS25	Framework	The framework to describe a development methodology from its origin to its practical, and to compare two or more development methodologies.	Static and dynamic
PS28	Model	Theoretical model to promote reuse within the organisation.	–
PS29	Model	ISS business model.	Dynamic

Table 4. Frameworks/methods, model and tools to support ISS development

Facet	Benefits	Challenges
Software Development	Better software quality (PS2, PS3, PS8, PS18, PS20, PS25)	Lack of documentation (PS21)
	Reduced development time and time-to-market (PS2, PS20, PS21)	Missing functionality (PS21)
	Shared community debugging (PS2)	Balancing between architectural refactoring and implementing new requirements (PS21)
	Reduced development cost (PS6, PS7, PS10, PS20)	Complexity in using and configuring the shared asset (PS21)
	Increase innovativeness (PS13, PS18)	Security (PS2, PS3, PS11, PS27)
	Avoid duplicate work and promote the reuse of software (PS2, PS3, P10, PS12, PS13, PS17, PS19, PS28)	
Process Management	Define an entry path for newcomers (PS25)	Building an effective community (PS13, PS21)
	Define ways of working (PS25)	
Tool and Technology	Increase information availability and visibility (PS25)	Migration from existing tools and infrastructure (PS2, PS5)
		Maintain code tree, platform, version control and related software engineering tools (PS2)
		Time consuming to search and navigate the software forge (PS2, PS3, PS8, PS16)
Managerial and Organisation	Rapid re-deployment of key developers (PS2, PS12, PS21)	Leadership and task assignment (PS2)
	Improve company's image (PS26)	Achieving a high level of commitment (PS13)
		Cultural resistance to change (PS11, PS14, PS29)

Table 5. Benefits and challenges of ISS approach

ISS approach. Once the software forge attains a certain size, searching and navigating through projects and components details can be time consuming (PS2, PS3, PS8, PS16). Thus, a proper search and navigation infrastructure is important for all contributors. Typically, each software project and group within the organisation has their own infrastructure e.g. version control, bug reports etc., that suits their needs. Therefore, the introduction of uniform toolset and infrastructure in the organisation is a challenges from a technology and user perspective (PS2, PS5). Once they all have migrated, the new IT support is critical to maintain both software e.g. the uptime, running schedule backups and recovery when necessary, and hardware.

Managerial and organisation

In an ISS approach, a community of contributors consists of developers who are familiars with the OSS environment within the organisation e.g. source tree, bug reporting, source management tools, corporate specific coding, commenting and code review process. This makes it easier to identify the talent across organisations (PS21). Hence, an ISS approach creates an opportunity for rapid re-deployment of developers from one project to another and from one product to another (PS2). It also increases the number of parallel development (PS12).

While most organisations today operate in a hierarchical organisational structure, the adoption of an ISS approach creates a virtual organisation (PS2). Contributors may come from different business units within the organisation. Hence it makes it more difficult to manage the talent and skill set at the corporate level. Additionally, it takes more time and effort for people to communicate (PS3), for example, the coding standard must be maintained at corporate level, rather than at a project or group level, and new developers must be trained for maximum use of an ISS approach.

Cultural resistance to change is one of the main challenges for ISS approach adoption (PS11, PS14, PS29). It requires a high level commitment from all stakeholders (PS13). ISS approach changes the relationship within the organisation from a one way dependency to a two way dependency (PS29). Those who are used to developing software, may become the users of the software, and vice versa.

DISCUSSION

Implication for Research and Practice

As far as we are aware of, this is the first attempt to review on inner source research by incorporating the relevant SE and IS literature in a systematic way. Our findings show while a great concentration of empirical research on how organisations adopt ISS development into their internal software development processes, other research areas receive much less attention. One of the implications of these findings for research and practice is the need for more empirical studies on engineering practices, tools and technologies and management practices to support ISS development. Specifically, while ISS development is highly influenced by OSS development, there is a need to translate OSS practices to suit the organisational context to achieve the benefits associated with OSS.

Our results show that majority of our primary studies are investigating ISS approach in the context of large, and globally distributed organisations. More studies that empirically validate existing frameworks, methods or models in different context would be helpful to understand their generalisability. Furthermore, to advance our understanding of the inner source phenomenon, researchers need to draw on theoretical foundations that have been used in prior research on OSS, as well as other theoretical lens that are considered relevant to ISS approach.

The implication for practice also lie in the evidence of the benefits and challenges of ISS development. The findings have shown that the adoption of ISS development helps organisations to improve better quality, time-to-market and innovativeness. However, as suggested by Brown et al. [4], that newcomers should understand the reality of the method through an appropriate enculturation, so that they can recognise what works and what does not work, and thus be aware of changing working processes.

Threats to Validity

Our study is not impervious to threats to validity, which may affect the outcome of this study. In the following section, the threats to validity of this study will be identified and discussed. To mitigate selection bias, we had tested various versions of the search string. We did not use the variation of “open source” for the following reasons. We observed that papers use the term “libre” together with the term “open source”. In addition, papers that use the term OSS or FLOSS must open the abbreviation, which contain “open source”.

To limit subjective bias for an individual reviewer, each paper was reviewed by two reviewers when applying inclusion/exclusion criteria. Prior to the actual selection of the primary studies, all reviewers performed pilot runs with 50 papers. The aim was to see whether all reviewers had the same understanding and perspective on the inclusion/exclusion criteria. Any dissimilarity in assessment between reviewers was discussed in the presence of all reviewers.

CONCLUSION AND FUTURE WORK

Influenced by the success of OSS development, the area of ISS development is gaining more attention from both academic and practitioners. This approach allows organisations to create high quality products in a shorter timeframe by combining heterogeneous development. Unlike existing literature reviews, this review is performed systematically and focuses specifically on ISS development within organisations. Furthermore, our review is interdisciplinary, drawing on both software engineering and information system literature to provide an extensive overview of the ISS phenomenon.

Through our SLR, the study establishes a state of research on ISS development. We found that the case study approach is the common research approach undertaken in the area. While the main contributions of the primary studies is in the form of theory, we also identified existing frameworks/methods, models and tools proposed in the literature to support ISS development as well as a set of benefits and challenges associated with ISS development.

We envision future work could perform a deeper analysis and synthesis on the empirical research on ISS development. Based on this analysis and synthesis, we will further investigate the limitations of the current research on ISS development and establish a research agenda on inner source. To enhance the findings of this review, we intend to conduct a comprehensive survey of practitioners to identify the key challenges involved in ISS development and propose some resolution strategies to overcome the challenges.

ACKNOWLEDGMENTS

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

REFERENCES

1. O. Alexy, J. Henkel, and M. W. Wallin. 2013. From Closed to Open: Job Role Changes, Individual Predispositions, and the Adoption of Commercial Open Source Software Development. *Research Policy* 42, 8 (2013), 1325–1340.
2. D. Avison and G. Fitzgerald. 1995. *Information Systems Development: Methodologies, Techniques and Tools*. McGraw-Hill Education.
3. P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. 2007. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *Journal of Systems and Software* 80, 4 (2007), 571–583.
4. J. S. Brown, A. Collins, and P. Duguid. 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18, 1 (1989), 32–42.
5. M. Capraro and D. Riehle. 2017. Inner Source Definition, Benefits, and Challenges. *Comput. Surveys* 49, 4 (2017).
6. K. Crowston, K. Wei, J. Howison, and A. Wiggins. 2012. Free/Libre Open-source Software Development: What We Know and What We Do Not Know. *Comput. Surveys* 44, 2 (2012), 7:1–7:35.
7. J. Dinkelacker, P. K. Garg, R. Miller, and D. Nelson. 2002. Progressive Open Source. In *Proceedings of 24th ICSE*. 177–184.
8. T. Dybå and T. Dingsøyr. 2008. Strength of Evidence in Systematic Reviews in Software Engineering. In *Proceedings of the International Symposium on ESEM*. 178–187.
9. T. Dybå, T. Dingsøyr, and G. K. Hanssen. 2007. Applying Systematic Reviews to Diverse Study Types: An Experience Report. In *Proceedings of 1st International Symposium on ESEM*. 225–234.
10. M. E. Falagas, E. I. Pitsouni, G. A. Malietzis, and G. Pappas. 2008. Comparison of PubMed, Scopus, Web of Science, and Google Scholar: Strengths and Weaknesses. *FASEB Journal* 22, 2 (2008), 338–342.
11. O. Gassmann and E. Enkel. 2004. Towards a Theory of Open Innovation: Three Core Process Archetypes. In *Proceedings of R&D Management Conference*.
12. G. Gaughan, B. Fitzgerald, and M. Shaikh. 2009. An Examination of the Use of Open Source Software Processes as a Global Software Development Solution for Commercial Software Engineering. In *Proceedings of 35th Euromicro Conference on SEAA*.
13. T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. 2006. A Model for Technology Transfer in Practices. *IEEE Software* 23, 6 (2006), 88–95.
14. M. Grottke, L. M. Karg, and A. Beckhaus. 2010. Team Factors and Failure Processing Efficiency: An Exploratory Study of Closed and Open Source Software Development. In *Proceedings of 34th IEEE Annual COMPSAC*. 188–197.
15. V. K. Gurbani, A. Garvert, and J. D. Herbsleb. 2006. A Case Study of a Corporate Open Source Development Model. In *Proceedings of 28th ICSE*. 472–481.
16. V. K. Gurbani, A. Garvert, and J. D. Herbsleb. 2010. Managing a Corporate Open Source Software Asset. *Commun. ACM* 53, 2 (2010), 155–159.
17. Ø. Hauge, C. Ayala, and R. Conradi. 2010. Adoption of Open Source in Software-Intensive Organizations - A Systematic Literature Review. *Information and Software Technology* 52 (2010), 1133–1154.
18. Ø. Hauge, C.-F. Sørensen, and A. Røsdal. 2007. Surveying Industrial Roles in Open Source Software Development. In *Open Source Development, Adoption and Innovation*. Vol. 234. Springer, Boston, MA.
19. M. Höst and A. Oručević-Alagić. 2011. A Systematic Review of Research on Open Source Software in Commercial Software Product Development. *Information and Software Technology* 53 (2011), 616–624.
20. M. Ivarsson and Tony Gorschek. 2011. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering* 16, 3 (2011), 365–395.
21. B. Kitchenham and S. Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report. Keele University and Durham University.
22. S. Lim, T. Saldanha, S. Malladi, and N. P. Melville. 2009. Theories Used in Information Systems Research: Identifying Theory Network in Leading IS Journals. In *ICIS 2009 Proceedings*. 91.
23. J. Lindman, M. Rossi, and P. Marttiin. 2008. Applying Open Source Development Practices Inside a Company. In *Open Source Development, Communities and Quality*. Vol. 275. Springer, Boston, MA.
24. J. Lindman, M. Rossi, and P. Marttiin. 2010. Open Source Technology Changes Intra-Organizational System Development - A Tale of Two Companies. In *Proceedings of ICIS, paper 151*.

25. J. Linåker, M. Krantz, and M. Höst. 2014. On Infrastructure for Facilitation of Inner Source in Small Development Teams. In *Proceedings of the 15th International Conference on PROFES*. 149–163.

26. G. Martin and A. Lippold. 2011. Forge.mil: A Case Study for Utilizing Open Source Methodologies Inside of Government. In *IFIP Advances in Information and Communication Technology*, Vol. 365. 334–337.

27. C. Melian and M. Mähring. 2008. Lost and gained in translation: Adoption of open source software development at Hewlett-Packard. In *Open Source Development, Communities and Quality*. Vol. 275. Springer, Boston, MA, 93–104.

28. M. B. Miles and M. A. Huberman. 1994. *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE.

29. L. Morgan, J. Feller, and P. Finnegan. 2011. Exploring Inner Source as a Form of Intra-Organisational Open Innovation. In *Proceedings of ECIS 2011, Paper 151*.

30. A. Neus and P. Scherf. 2005. Opening minds: Cultural change with the introduction of open-source collaboration methods. *IBM Systems Journal* 44, 2 (2005), 215–225.

31. Tim O'Reilly. 2000. Open Source and OpenGL. (2000). `http://archive.oreilly.com/pub/a/oreilly/ask_tim/2000/opengl_1200.html`

32. W. J. Orlikowski and C. S. Lacono. 2001. Research Commentary: Desperately Seeking the 'IT' in IT Research – A Call to Theorizing the IT Artifact. *Information Systems Research* 12, 2 (2001), 121–134.

33. A. Oručević-Alagić and M. Höst. 2016. A Two Phase Case Study on Implementation of Open Source Development Practices within a Company Setting. In *Proceedings of International Conference on SEKE*. 63–70.

34. N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. 2014. Software Development in Startup Companies: A Systematic Mapping Study. *Information and Software Technology* 56, 10 (2014), 1200–1218.

35. M. Pulkkinen, O. Mazhelis, P. Marttiin, and J. Merliuoto. 2007. Support for Knowledge and Innovations in Software Development – Community within Company: Inner Source Environment. In *Proceedings of 3rd International Conference on WIST*. 141–150.

36. D. Riehle, M. Capraro, D. Kips, and L. Horn. 2016. Inner Source in Platform-based Product Engineering. *IEEE Transcation on Software Engineering* 42, 12 (2016), 1162–1177.

37. D. Riehle, J. Ellenberger, T. Menahem, B. Mikhailovski, Y. Natchetoi, B. Naveh, and T. Odenwald. 2009. Open Collaboration within Corporations Using Software Forges. *IEEE Software* 26, 2 (2009), 52–58.

38. M. Ripatti, R. Kilamo, K. Salli 1, and T. Mikkonen. 2015. Internal Marketplace as a Mechanism for Promoting Software Reuse. In *Proceedings of 14th SPLST*. 119–133.

39. C. Robson. 2011. *Real World Research*. John Wiley & Sons.

40. S. Sharma, V. Sugumaran, and B. Rajagopalan. 2002. A Framework for Creating Hybrid-Open Source Software Communities. *Information Systems Journal* 12, 1 (2002), 7–25.

41. K.-J. Stol, P. Avgeriou, M. A. Babar, Y. Lucas, and B. Fitzgerald. 2014. Key Factors for Adopting Inner Source. *ACM Transaction on Software Engineering and Methodology* 23, 2 (2014).

42. K.-J. Stol, M. A. Babar, P. Avgeriou, and B. Fitzgerald. 2011. A Comparative Study of Challenges in Integrating Open Source Software and Inner Source Software. *Information and Software Technology* 53, 12 (2011), 1319–1336.

43. K.-J. Stol and B. Fitzgerald. 2015. Inner Source – Adopting Open Source Development Practices in Organizations. *IEEE Software* 32, 4 (2015), 60–67.

44. M. Theunissen, D. Kourie, and A. Boake. 2008. Corporate-, Agile-, and Open Source Software Development: A Witch's Brew or an Elixir of Life. *Balancing Agility and Formalism in Software Engineering* 5082, 84–95 (2008).

45. R. Torkar, P. Minoves, and J. Garrigós. 2011. Adopting Free/Libre/Open Source Software Practices, Techniques and Methods for Industrial Use. *Journal of the Association for Information Systems* 12, 1 (2011).

46. F. van der Linden. 2009. Applying Open Source Software Principles in Product Lines. *UPGRADE* 10, 3 (2009), 32–40.

47. F. van der Linden, B. Lundell, and P. Marttiin. 2009. Commodication of Industrial Software: A Case for Open Source. *IEEE Software* 26, 4 (2009).

48. P. Vitharana, J. King, and H. S. Chapman. 2010. Impact of Internal Open Source Development on Reuse: Participatory Reuse in Action. *Journal of Management Information Systems* 27, 2 (2010), 277–304.

49. J. Wesselius. 2008. The Bazaar Inside the Cathedral: Business Model for Internal Markets. *IEEE Software* 25, 3 (2008), 60–66.