The Journal of Systems and Software 163 (2020) 110520

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Inner source software development: Current thinking and an agenda for future research

Henry Edison*, Noel Carroll, Lorraine Morgan, Kieran Conboy

Lero, National University of Ireland Galway, Galway, Ireland

ARTICLE INFO

Article history:
Received 5 March 2019
Revised 13 December 2019
Accepted 11 January 2020
Available online 15 January 2020

Keywords:
Inner source
Inner source software development
Research agenda
Systematic literature review

ABSTRACT

Context: Inner source software development (ISSD) has been viewed as an alternative approach in which organisations adopt open source software development (OSSD) practices and exploit its benefits internally.

Objective: In this paper, we aim to provide an extensive review of current research on ISSD and to establish a research agenda on this domain.

Method: The review is primarily performed using a systematic literature review protocol.

Results: We identified, critically evaluated and integrated the findings of 37 primary studies, describing 25 empirical research papers, 10 frameworks/methods, models and tools to support the implementation of inner source, as well as a set of benefits and challenges associated with ISSD.

Conclusion: This study presents four main contributions. First, the study provides an in-depth review of ISSD to date, i.e. the evolution of research across inner source, contributions of existing research developments, and theories, models and frameworks used to study inner source. Second, our review applies the OSSD approach framework as the lens to analyse ISSD. Third, the review updates the key challenges associated with ISSD from a management perspective. The final contribution is the establishment of a research agenda to advance knowledge on ISSD.

1. Introduction

Open source software (OSS) has significant commercial value and is now mainstream across software markets (Feller and Fitzgerald, 2000; Capraro and Riehle, 2017). This model of software development is typically characterised by voluntarism, meritocracy, modular architecture, distributed global teams and openness of the source code for modification and distribution (Fitzgerald, 2006). Moreover, the collaborative development model of OSS delivers high quality software products with a faster development cycle (e.g. Linux, Apache, Eclipse, Mozilla) while simultaneously reducing development costs and risks (Feller and Fitzgerald, 2000). As a result, organisations are keen to adopt and replicate this model in their internal development processes and exploit its benefits. In the literature, this phenomenon is referred to as "inner source" (Dinkelacker et al., 2002; Wesselius, 2008; Gurbani et al., 2010; Stol et al., 2014), "progressive open source" (Dinkelacker et al., 2002), or "corporate open source" (Gurbani et al., 2006). Leading organisations such as Lucent Technologies, Nokia, Philips Healthcare, Bosch, Paypal, IBM and HP etc., have provided examples of inner source software development (ISSD) implementations in their respective organisations (Capraro and Riehle, 2017). Indeed, the growth of inner source adoption has also led to the emergence of inner source research in the scientific literature (see for example Lindman et al., 2008; Lindman et al., 2010; Stol et al., 2011; Stol et al., 2014; Stol and Fitzgerald, 2015). Nonetheless, ISSD does not have clearly defined tasks, artefacts or roles as in other contemporary methods, such as Scrum (e.g. Scrum meeting, backlog, Scrum master) or Kanban (e.g. Kanban board) (Stol et al., 2014). A number of common ISSD practices, however, are recognised, including universal access to source code and documentation, transparent fishbowl development environment, peer-review of contributions through organisation-wide scrutiny of contributions, informal communication channels (e.g., mailing lists, wiki, etc.), self-selection of motivated contributors, frequent releases and early feedback, around the clock development (Morgan et al., 2011; Stol et al., 2014).

* Corresponding author.
E-mail addresses: henry.edison@nuigalway.ie (H. Edison), noel.carroll@nuigalway.ie (N. Carroll), lorraine.morgan@nuigalway.ie (L. Morgan), kieran.conboy@nuigalway.ie (K. Conboy).

https://doi.org/10.1016/j.jss.2020.110520

Previous literature reviews on open source (Hauge et al., 2010; Höst and Oručević-Alagić, 2011; Crowston et al., 2012) indicate that there is a trend towards organisational adoption of open source principles in their internal development processes and call for more empirical research in this area. Indeed, we have observed an increased focus on inner source from practitioners over the past four years. For example, the InnerSource Commons (ISC)[1] was founded in 2015 and is an active community of over seventy companies, academic institutions and government agencies with the goal of creating and sharing knowledge about inner source adoption. This group has also organised eight inner source summits from 2016 to 2019, hosted in both European and USA locations, which is testament to the increased practitioner interest in the area. In practice, however, the adoption of such practices is unique and varies across organisations (Stol et al., 2011), hence we believe a systematic literature review is necessary to synthesise variances in existing research. Furthermore, a consequence of the growing number of empirical studies in ISSD is the need to adopt systematic approaches to assessing and aggregating research outcomes in order to provide a balanced and objective summary of research evidence for inner source. This invites the research community to adopt an evidence-based approach to "provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision-making process regarding the development and maintenance of software" (Kitchenham et al., 2004, p.2).

With this in mind, the objective of our research is to further synthesise and update the literature review on inner source. We complement existing reviews by identifying the current status of this research domain and paving the way for more empirical studies on inner source. The most recent literature review on inner source was carried out by Capraro and Riehle (2017), reviewing a total of 43 primary studies from 2001 to 2014. We build on this work by (i) systematically reviewing current developments in inner source (over the past five years, for example, there has been an increase in the adoption of various inner source practices, tools, and technologies reported across more recent publications which needs to be assessed in terms of research evidence on inner source) and (ii) addressing different research questions not covered in previous studies. A revisit on this area can reveal how research on inner source (e.g. research focus and contributions) has changed over the past number of years and serve as a basis to establish a future research agenda. Moreover, for practitioners, knowledge about the concept of inner source, its practices and tools, and its challenges and benefits can help them to better understand their organisational fit with inner source. To achieve our research objective, we formulated the following research questions:

- *RQ1 – How has inner source research changed over time in terms of focused knowledge areas and contributions?* This research question establishes the state of the art on empirical research in inner source, including the evolution of research focus, contributions, and theories used to study inner source.
- *RQ2 – What are the main factors that characterise ISSD?* As each implementation of inner source is unique to the organisation, this research question establishes the state of practice of inner source reported in the literature.
- *RQ3 – What are the reported benefits and challenges associated with ISSD?* This research question provides an update on the identified benefits and barriers relevant to inner source.
- *RQ4 – What are the research gaps in the current research on inner source?* Based on the findings of RQ1, RQ2 and RQ3, this research question identifies significant gaps across the inner source literature and provides a roadmap for future endeavours in inner source research.

This study is an extension to our research paper published and presented in the 14th International Symposium on Open Collaboration (OpenSym) 2018 (Edison et al., 2018). Building on feedback from the inner source community, we performed a deeper analysis for answering RQ3, and added new research questions around the state of the art (RQ1), characterising inner source (RQ2), and research gaps (RQ4). To answer our research questions, we conducted a systematic literature review (SLR) as proposed by Kitchenham and Charters (2007) to accumulate primary studies aiming to improve the understanding and to ascertain the validity and reliability of claims and propositions relating to inner source. Moreover, a SLR facilitates in identifying and collecting key papers in a specific area of interest (such as inner source) and evaluating and interpreting the reported discussions and findings (Kitchenham and Charters, 2007).

This study presents four main contributions, as follows:

- The SLR provides an in-depth review of inner source research reported in the literature, e.g. the evolution of research across inner source, the contributions of existing research developments, and methods and theories used to study inner source. This review indicates that even though inner source is not a well-defined methodology, the majority of research has centred on its adoption and adaptation across various contexts. Other knowledge areas such as software requirements, software testing, configuration management, software engineering (SE) economics, etc. receive less attention. This also highlights some concerns around the evaluation efforts of inner source and the lack of quantifiable evidence to clearly report on its benefits.
- The review has applied an OSS framework (Feller and Fitzgerald, 2000) as a theoretical lens to analyse our findings on inner source. The use of the OSS framework (based on Zachman's IS architecture and Checklandâs CATWOE framework (Feller and Fitzgerald, 2000)) has supported key research in the development of OSS (Aksulu and Wade, 2010), making it a suitable approach to guide inner source research developments. Specifically, this framework assisted us in maintaining a balanced examination of inner source approaches from five different perspectives, i.e. 'what', 'why', 'when and where', 'how', and 'who'.
- Our review also updates the key challenges associated with inner source from a management perspective. For example, our SLR study identified new challenges reported across inner source relating to the lack of guidance for implementing inner source, the lack of metrics for evaluating inner source initiatives, and the management of tensions during inner source adoption or transformations.
- Finally, based on the findings of RQ1, RQ2 and RQ3, our fourth contribution of this SLR identifies five key themes to formulate a research agenda on inner source. Researchers could use this agenda to better position and align their work in this area.

The remainder of this article is structured as follows. Section 2 presents a brief discussion of related work in this area. This is 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. The discussion of the results are presented in Section 6 followed by a conclusion in Section 7.

[1] ISC supports practitioners and those who want to learn about inner source by a broad array of research developments and activities (http://innersourcecommons.org/).

Table 1
Overview of related literature reviews.

Facet	Hauge et al. (2010)	Höst and Oručević-Alagić (2011)	Crowston et al. (2012)	Capraro and Riehle (2017)	Our study
Objective	"To identify what we know about how organisations adopt OSS."	"...to understand the result of the research that has been carried out on the usage of open source software and open source software development in proprietary software development organisations."	"...to synthesise the empirical literature to date in order to clarify what we know and do not know about community-based FLOSS development and to suggest promising directions for further work"	"...assessing the state of the research and introducing a set of qualitative IS models that provide a unified view of IS research results."	To further synthesise and update the literature review on inner source
Review protocol	Systematic	Systematic	Partially	No	Systematic
Data sources	24 journals and 7 conferences in the area of SE and IS	Inspec and Compendex	opensource.mit.edu, FLOSS special issues and tracks in journals and conferences, AoM Journal, Association of Information Systems, ABI/Inform and Web of Science	Google Scholar, ACM Digital Library, and IEEEXplore	Compendex, ISI Web of Science, Scopus, AIS e-Library
Inclusion/ exclusion criteria	Yes	Partially	No	No	Yes
Quality assessment	Yes	Partially	No	No	Yes
Data extraction	Yes	Partially	Yes	No	Yes

2. Related work

Relevant reviews were identified by conducting a series of search runs that consisted of different combination of search terms in Compendex and Scopus digital libraries. The search terms used were "open source", "inner source", "systematic literature review", and "literature review". After identifying the related reviews, we also searched for additional reviews that are related to our study from the reference list.

We identified four secondary studies (Hauge et al., 2010; Höst and Oručević-Alagić, 2011; Crowston et al., 2012; Capraro and Riehle, 2017) on inner source that were considered relevant to this study. A summary of the related work is presented in Table 1. For example, the study by Hauge et al. (2010) and Höst and Oručević-Alagić (2011) performed literature reviews by following the guidelines provided by Kitchenham and Charters (2007). The study by Crowston et al. (2012) 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 (2017) presents an extensive literature review but does not report a SLR approach to deliver their findings.

Both of the reviews by Hauge et al. (2010) and Crowston et al. (2012) conducted a search on specific journals or conferences on open source, while the reviews by Höst and Oručević-Alagić (2011) and Capraro and Riehle (2017) focused on SE related databases such as Inspec, Compendex, ACM Digital Library and IEEE Xplore. Furthermore, the review by Hauge et al. (2010) did not report on the used search string.

Neither of the reviews Crowston et al. (2012) nor Capraro and Riehle (2017) reported explicit criteria for quality assessment of the primary studies. Contrary to these reviews, Hauge et al. (2010) and Höst and Oručević-Alagić (2011) adopt the quality assessment criteria developed by Dybå and Dingsoyr (2008). For our review, we adopted a comprehensive set of evaluation guidelines based on a scientific rigour and industrial relevance rubric (Ivarsson and Gorschek, 2011).

The aim of the reviews conducted by Hauge et al. (2010), Höst and Oručević-Alagić (2011) and Crowston et al. (2012) was to assess the current research on how organisations adopt open source software development (OSSD). They found that inner source is one way of adopting OSS for internal use. They also found that most of the studies on this area were in the form of experience reports and only two or three scientific papers, indicating that academic research on this area is lagging behind. However, they are different in terms of data sources used for the search process. For example, the review by Hauge et al. (2010) and Crowston et al. (2012) focused on specific repositories on open source, whereas the review by Höst and Oručević-Alagić (2011) used Inspec and Compendex database. The review by Capraro and Riehle (2017) searched both scientific papers and grey literature on inner source research and practice.

To synthesise and build on previous research, we conduct a SLR since it provides us with a methodologically rigorous review of research results. Thus, the synthesis of the best quality scientific studies on a specific topic or research question contributes to support the development of evidence-based guidelines for practitioners (Kitchenham et al., 2009).

3. Research methodology

3.1. Search strategy

To help build the search terms, a set of seed papers (see Table 2) were identified manually by all authors (hereon referred to as the reviewers). Keywords used in these papers were extracted, aggregated and used as the input for the search terms. In order to verify the quality of the search string, we conducted

Table 2
Seed papers.

ID	Author(s)	ID	Author(s)
SP1	Dinkelacker et al. (2002)	SP9	Melian and Mähring (2008)
SP2	Gaughan et al. (2009)	SP10	Morgan et al. (2011)
SP3	Gurbani et al. (2006)	SP11	Riehle et al. (2009)
SP4	Gurbani et al. (2010)	SP12	Sharma et al. (2002)
SP5	van der Linden et al. (2009)	SP13	Stol et al. (2011)
SP6	van der Linden (2009)	SP14	Stol et al. (2014)
SP7	Lindman et al. (2008)	SP15	Stol and Fitzgerald (2015)
SP8	Lindman et al. (2010)	SP16	Wesselius (2008)

Table 3
Search terms organisation.

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

a trial search on EngineeringVillage and Scopus. We compared the seed papers with the result set of the trial search. The search string captured 12 out of 16 papers. Some seed papers were not retrieved as they used different keywords for the population, e.g. inside and industry. Thus, we added these terms to the search string. In the final trial, we were able to retrieve all 16 seed papers.

Table 3 describes the generic search string. We did not use the variations 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, Free Open Source Software (FOSS) or Free/Libre Open Source Software (FLOSS) must open the abbreviation, which contains "open source". However, as the term "inner source" is generic and can be found in other domains, we limit the context to the software field. Moreover, we also scope the search process to find papers that discuss inner source or open source within an organisation, rather than a community.

A search for relevant literature was conducted on the metadata, in particular on the title, abstract and keyword. We decided to use electronic databases that have good coverage, familiarity, reputation, advanced features and exportability (Falagas et al., 2008). The electronic databases used to search for relevant literature were relevant to: (i) SE research: Compendex, ISI Web of Science, and Scopus (Dybå et al., 2007; Brereton et al., 2007) and (ii) information systems (IS) research: AIS e-Library.

Table 4 shows the search string used in each digital library and the corresponding search results. The first digital library search was done in August 2017. The second search was performed on Dec 2018 to cover the gap between August 2017 - Dec 2018. We performed a manual search by checking forward and backward references for each primary study. To complement the search, we also performed another search using Google Scholar. Taken together, we retrieved 14,925 articles from all digital libraries. We limited the search to only computer science, engineering or business economics research areas.

3.2. Selection strategy

The selection process of the primary studies was composed by three phases, as shown in Fig. 1. The initial step in this process was removing duplicates (based on similarity of title and authors), non-English and non-research papers. A total of 632 papers were found and removed. The next step was to remove other irrelevant papers based on the criteria for inclusion and exclusion, as shown in Table 5.

Our search strings yielded broad search results. The broad search was intended to identify as many papers as possible, since the nomenclature in the area is not well standardised. In addition, research on inner source is a cross disciplinary study, so terms used to describe this phenomenon are inconsistent. Thus, in the second phase, we removed 12,962 irrelevant studies, that mostly focused on the use of open source components to develop new software in various contexts, e.g. astronomy, enterprise resource planning, hydrology, telecommunication, and so forth.

In the next phase, selection criteria were performed on 1,331 remaining papers. Two reviewers evaluated each paper. The second, third, and fourth reviewers evaluated one-third of the papers each, whilst the first reviewer evaluated all of the papers. The papers were sorted alphabetically by the author's name before they were distributed to the reviewers. To be included, two reviewers had to be in agreement on a paper relevance. In the cases where both reviewers did not agree on the decision, a third reviewer vote was required. At the end of this phase, 1,293 papers were excluded.

The third phase was to apply inclusion and exclusion criteria on the full-text of the remaining 38 papers. We found two similar papers, which were published in two different conferences. We contacted the authors and received confirmation that one of the papers was not published in a peer-reviewed venue. Therefore, we excluded the non-peer reviewed paper and included 37 primary studies.

3.3. Quality assessment

In a SLR study quality assessment is essential to evaluate the existing research topic by using a trustworthy, rigorous, and auditable methodology (Kitchenham and Charters, 2007). In this study, the quality assessment was performed independently by each reviewer.

To assess the quality of primary studies, we adopted the scientific rigour and industrial relevance rubric (Ivarsson and Gorschek, 2011). For the scientific rigour, the quality was evaluated

Table 4
Search results.

Database	Generic search string	Number of articles
Compendex	(((((organi?ation OR firm OR company OR corporat* OR enterprise OR industr* OR vendor OR institution OR internal OR inside OR hybrid)) WN KY) AND (((inner?source OR innersource OR open?source OR opensource)) WN KY)) AND ((software) WN KY))	6,235
Web of Science	TOPIC:(organi?ation OR firm OR company OR corporat* OR enterprise OR industr* OR vendor OR institution OR internal OR inside OR hybrid) AND TOPIC: (''inner source'' OR ''inner-source'' OR innersource OR ''open source'' OR ''open-source'' OR opensource) AND TOPIC: (software)	3,089
Scopus	TITLE-ABS-KEY(organi?ation OR firm OR company OR corporat* OR enterprise OR industr* OR vendor OR institution OR internal OR inside OR hybrid) AND TITLE-ABS-KEY(''inner source'' OR ''inner-source'' OR innersource OR ''open source'' OR ''open-source'' OR opensource) AND TITLE-ABS-KEY(software)	5,458
AIS e-library	abstract:(organisation OR organization OR firm OR company OR corporat* OR enterprise OR industr* OR vendor OR institution OR internal OR inside OR hybrid) AND abstract:(''inner source'' OR ''inner-source'' OR innersource OR ''open source'' OR ''open-source'' OR opensource) AND abstract:software	136
Manual Search		7
	Total articles found	14,925

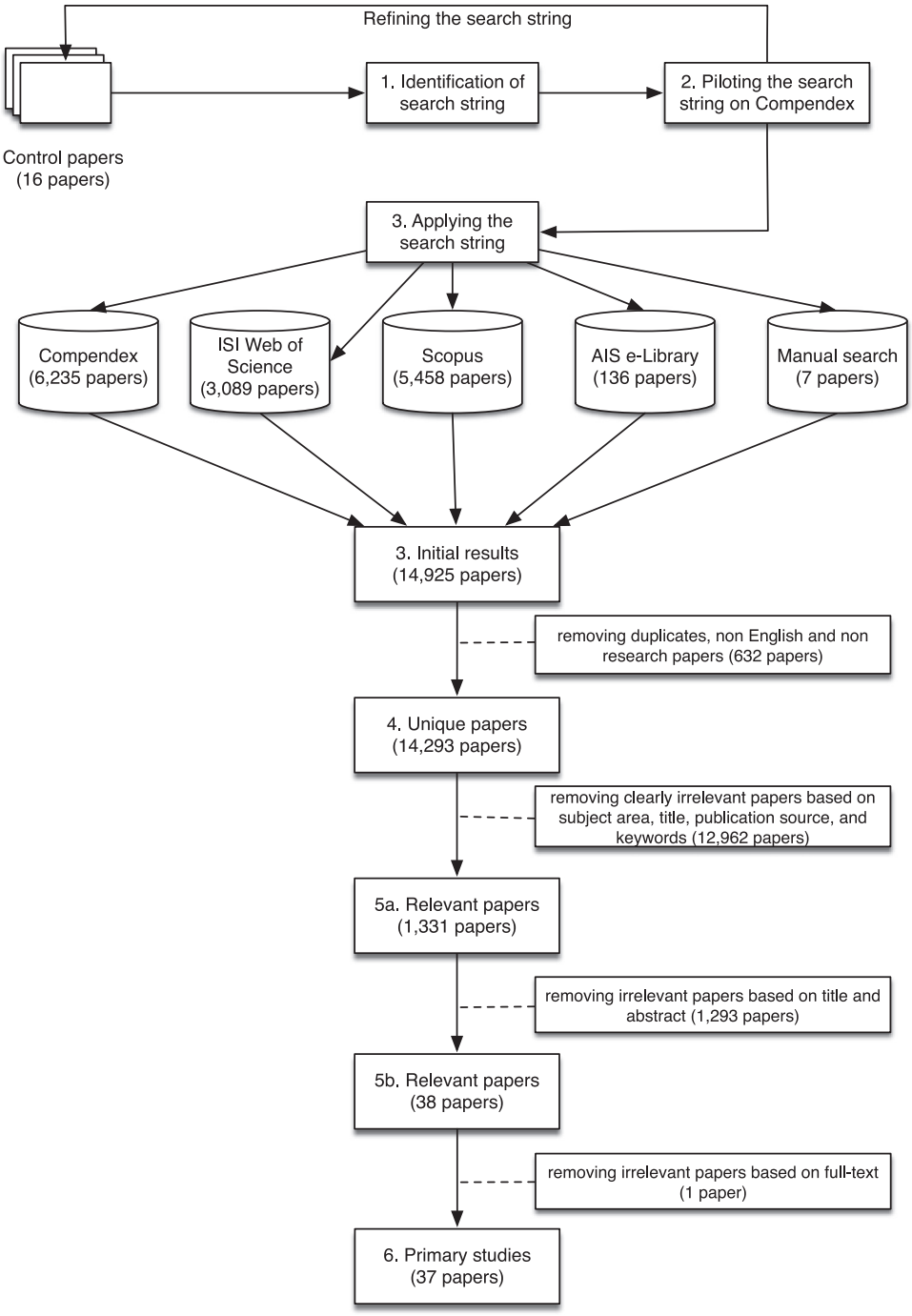

Fig. 1. Selection process of primary studies.

Table 5
Inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
Written in English	Non-English papers
Peer-reviewed papers	Non-scientific papers e.g. editorial, presentation, call for papers, keynote speeches, preface, panel report etc.
Full-text available	Full-text not available
Studies on inner source or the adoption of OSS processes/practices within an organisation	Studies on community-based OSS project e.g. "Apache and eclipse: Comparing open source project incubators."
	Studies that only focus on the use of OSS components to develop new software or tools e.g. "An open source Java framework for biometric web authentication based on BioAPI"
	Studies that evaluate the quality of OSS e.g. "A study of concurrency bugs in an open source software"

Table 6
Extracted Data.

	Property	Research Question
D1	Research Method	RQ1
D2	Knowledge area	RQ1
D3	Contributions of the studies	RQ1
D4	Definition, motivation, process, actors	RQ2
D5	Challenges & benefits	RQ3
D6	Research gaps	RQ4

using a three point scale: strong (1), medium (0.5), and weak (0), on the following aspects: (i) study context: the degree to which the reviewer can understand and compare it or the theory to another context, (ii) study design: the degree to which the reviewer can understand how rigorous the research method is applied in the study or how theoretical bases sound, and (iii) validity discussion: to what extent 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 theoretical 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 researchers, (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 which are relevant for practitioners.

3.4. Data extraction and synthesis

To help answer our research questions, a form for data extraction was constructed. The data extraction was performed during the full-text review. The data extracted from the primary studies are listed in Table 6.

We recorded data for D1 to D3 to answer the first research question. We used SWEBOK as it allows us to compare the existing research and to identify changes over time (Klotins et al., 2015; Berg et al., 2018). SWEBOK provides the boundary of SE with respect to other disciplines (Bourque and Fairley, 2014). It contains 15 knowledge areas in SE discipline. To enrich our findings, we keep track of coverage, e.g. how many knowledge areas are covered by the primary studies. We categorised the primary studies according to the applied research method, including the data collection and analysis method and the theoretical lens. We extracted the mentioned research method without interpreting the content of the study. If the focus of the study is directed toward reporting industrial experiences without stating research questions or research methods, it is classified as an experience report.

To answer RQ2, we used the OSS approach framework (Feller and Fitzgerald, 2000) to guide the data extraction and synthesis. The OSS approach was derived from Zachman's IS architecture and Checkland's CATWOE framework and specifically designed to analyse the OSS approach. As described in Section 1, inner source presents an alternative approach of OSS adoption in an organisation (Hauge et al., 2010; Höst and Oručević-Alagić, 2011), thus we argue that the framework is a suitable fit to study inner source. It assists us in maintaining a balanced examination of inner source from five different perspective, i.e. 'what', 'why', 'when and where', 'how', and 'who'. The adapted framework analysis for the inner source approach is shown in Table 7. To answer RQ3, we aggregated and complied the corresponding challenges and benefits of inner source. We also collected research gaps identified in each primary study to answer RQ4.

4. Characteristics of primary studies

The 37 primary studies are listed in Table 8 and referred to using their IDs throughout the rest of the paper.

4.1. Publication sources and years

The distribution of the years and venues of the primary studies is shown in Fig. 2. While the term inner source was introduced in 2000 (O'Reilly, 2000), the first research paper in this topic was published two years later (PS2). All primary studies were dated from year 2002 onwards. There were peaks in inner source studies between 2007 to 2011, in which four studies were published each year during this period. Fig. 2 shows that in recent years there has been a steady number of publications in this area. 14 out of 37 papers (38%) are published in journals i.e. Research Policy, IST, JAIS, TOSEM, etc., whereas 23 papers (62%) are presented in various conferences in both SE and IS, i.e. OSS Symposium, ECIS, PROFES, etc.

4.2. Quality of the primary studies

In general, 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. 3. The maximum score that a paper could get was 7 (see Section 3.3). Studies with scores below the lower quartile lacked clear information about the study design and threats to validity, as re-

Table 7
Framework analysis for the ISSD (adapted from (Feller and Fitzgerald, 2000)).

What (Transformation)
What defines a software product/project as inner source?
What types of products/projects tend to be inner source?
Why (World View)
What are the technological motivations for ISSD?
What are the economic motivations for ISSD?
What are the socio-political motivations for ISSD?
When and Where (Environment)
What are the temporal dimensions of ISSD?
What are the spatial/geographic dimensions of ISSD?
How
How is the ISSD process organised?
What tools are used to support the ISSD?
Who (Client, Actor, Owner)
What are the characteristics of the individual developers contributing to inner source projects?
What are the characteristics of the organisations adopting inner source?
What are the characteristics of the users of inner source software?

Table 8
List of Primary Studies.

ID	Author(s)	ID	Author(s)	ID	Author(s)
PS1	Alexy et al. (2013)	PS14	Neus and Scherf (2005)	PS27	van der Linden (2009)
PS2	Dinkelacker et al. (2002)	PS15	Oručević-Alagić and Höst (2016)	PS28	Vitharana et al. (2010)
PS3	Gaughan et al. (2009)	PS16	Pulkkinen et al. (2007)	PS29	Wesselius (2008)
PS4	Grottke et al. (2010)	PS17	Riehle et al. (2016)	PS30	Carroll et al. (2018)
PS5	Gurbani et al. (2006)	PS18	Riehle et al. (2009)	PS31	Eckert et al. (2017)
PS6	Gurbani et al. (2010)	PS19	Ripatti et al. (2015)	PS32	Capraro et al. (2018)
PS7	Hauge et al. (2007)	PS20	Sharma et al. (2002)	PS33	Bank et al. (2017)
PS8	Linåker et al. (2014)	PS21	Stol et al. (2014)	PS34	Schreiber et al. (2014)
PS9	Lindman et al. (2008)	PS22	Stol et al. (2011)	PS35	Smith and Garber-Brown (2007)
PS10	Lindman et al. (2010)	PS23	Stol and Fitzgerald (2015)	PS36	Martin and Hoffman (2007)
PS11	Martin and Lippold (2011)	PS24	Theunissen et al. (2008)	PS37	Kenny et al. (2017)
PS12	Melian and Mähring (2008)	PS25	Torkar et al. (2011)		
PS13	Morgan et al. (2011)	PS26	van der Linden et al. (2009)		

Table 9
Knowledge areas and primary studies.

Knowledge Areas	Coverage	Topic	Sources
SE Process	5/5	SE Definition	PS2, PS5, PS6
		Software Life Cycle	PS3, PS7, PS8, PS9, PS10, PS11, PS12, PS15, PS17, PS21, PS22, PS23, PS24, PS25, PS26, PS27, PS29, PS35, PS36
		Software Process Assessment and Improvement	PS31
		Software Measurement	PS32
		Software Process Tools	PS16, PS18, PS19, PS28, PS34
SE Professional Practice	1/7	Group Dynamics and Psychology	PS1, PS4, PS13, PS14, PS20, PS30, PS33
Computing Foundation	1/17	Algorithm and Foundation	PS37

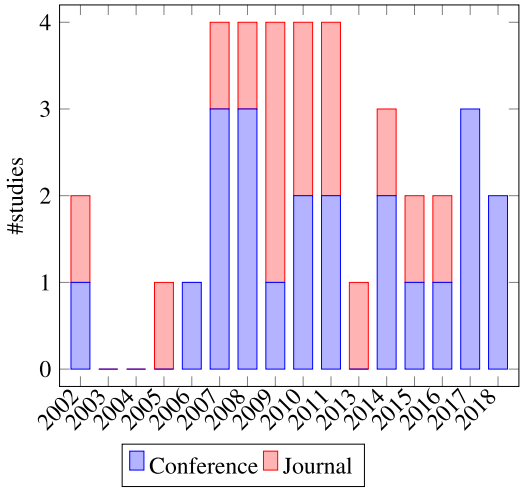

Fig. 2. Publication venues and year.

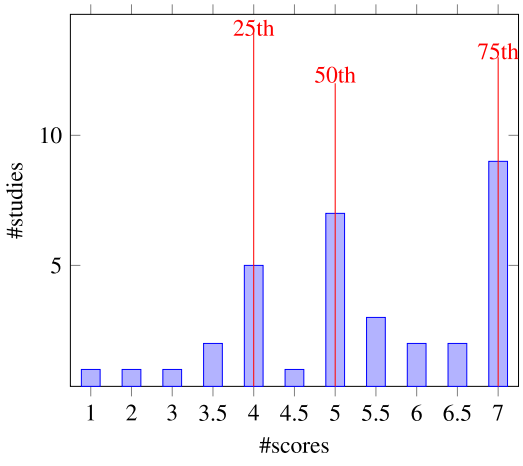

Fig. 3. Distribution of the quality scores.

quired in the rigour rubric. Typically, these studies were published in practitioner-oriented journals e.g. IEEE Software (PS22, PS25, PS28) or Communication of the ACM (PS5). Studies with minimum scores are philosophical papers. Since, the identified or proposed frameworks or methods have not been studied empirically in industry settings, the relevance scores were zero. Moreover, most of the studies within the interquartile range did not discuss validity threats and how they are mitigated, which negatively affected the trustworthiness of the reported findings (Robson, 2011).

5. Results

In this section, we present the findings of our literature review. We structure this section according to our research questions.

5.1. RQ1 – How Has Inner Source Research Changed Over Time in Terms of Focused Knowledge Areas and Contributions?

In this section, we present an overview of the body of inner source knowledge in terms of the SWEBOK knowledge areas (Section 5.1.1) and the contribution of the primary studies (Section 5.1.2).

5.1.1. Knowledge areas

Table 9 summarises the identified primary studies and respective SWEBOK knowledge areas (KAs). The coverage column shows how many topics under the KAs are covered by the articles. For example, the software design KA covers seven related topics: software design fundamentals, key issues in software design, software structure and architecture, user interface design, software design quality analysis and evaluation, software design notations, software design strategies and methods, and software design tools. Coverage

Table 10
Overview of theoretical foundations for studying inner source.

Theory/Framework/Model	Description	Application	Sources
Inner source framework (Stol et al., 2014)	The framework presents nine important factors that need to be considered when implementing inner source.	The framework was used to assess whether inner source would fit the case organisation or not, and what challenges would emerge.	PS8
Institutional theory on enrolling group interest (Scott, 2001)	The theory suggests that the process of understanding different interest of the different groups to enrol becomes essential to understanding how institutions evolve.	The theory was used to explain how organisations evolve when inner source was institutionalised, in terms of the change in the structure and job role.	PS1, PS10
Actor network theory (ANT) (Latour, 1986)	The theory addresses how technology influences and is influenced by social elements on a setting over time. The ideas, values, and intentions of social actors become inscribed in a technology through interactions.	The theory was used to explain the translation of OSSD process into a corporate context.	PS12
Open innovation theory (Gassmann and Enkel, 2004)	There are three core open innovation processes: the outside-in process (bring external knowledge inside the company), the inside-out process (spreading knowledge outside the company), and the coupled process (combine the outside-in and inside-out process).	The theory was used to examine inner source as a form of open innovation within an organisation.	PS14
Knowledge life cycle model (Maier et al., 2005; Ghalib, 2004)	The model describes 8 phases of how knowledge is created, enriched and cumulated.	The model was used to examine how an inner source portal (for sharing assets) supports knowledge processes and innovation.	PS16
Methodology comparison theory (Avison and Fitzgerald, 1995)	The theory suggests that comparing two methodologies is beneficial to understand the nature of methodology and to guide its selections for a particular suitable context. Thus, theory provides 26 'ideal-type' criteria that a methodology should fulfil.	The theory was used to analyse open source methodology and Streamline methodology used by Ericsson.	PS25
Tension theory (Huxham and Beech, 2003; Smith and Lewis, 2011)	The theory suggests that tensions can be either disruptive where for example they can lead to some form of breakdown, or they can be beneficial, by fostering competition and challenges with management involved in the process of continually resolving tensions.	The theory was used to explain different types of tensions that arise from the adoption of inner source in organisations.	PS30
CMMI-DEV (SEI, 2010)	The model is a process improvement training and appraisal program providing guidance on applying best practices in a software development organisation.	The model was used to build a maturity model of inner source implementation.	PS31

1/7 in the software design KA means that only one topic is covered, while the other six are not addressed at all.

Our primary studies reveal that the current research on inner source addresses only three KAs in SE. The majority of the primary studies covered all topics in SE process KA (29 papers), followed by SE professional practices KA (one topic, addressed by seven papers), software construction KA (one topic, addressed by two papers) and computing foundation KA (one topic, addressed by one paper).

SE process KA aims to support work engineering activities to develop, maintain and operate software (Bourque and Fairley, 2014). It includes five topics: SE definition, software life cycle, software process assessment and improvement, software measurement and software process tools. The SE definition topic is concerned with a definition of software process, software process management and software process infrastructure. PS2, PS5 and PS6 partially addressed this topic by emphasising on the infrastructure to support development activities. These papers are considered the first papers that defined an implementation of an inner source approach, including the need of infrastructure to support the development activities. Since then, more research was conducted to adapt the proposed inner source implementation in another context, e.g. a global distributed environment (PS8), government organisations (PS10), a small team (PS9), or different domains like hardware and software or medical devices (PS11, PS12, PS15, PS17, PS21, PS22, PS23, PS24, PS25, PS26, PS27, PS29, PS35, PS36).

Several studies addressed the SE process tools topic by investigating the use of various tools to support and manage inner source processes. This includes knowledge management systems (PS16), a repository or directory (PS18, PS19, PS28, PS34) to store documentation, software components, test cases, and problem reports. While research on inner source encourages the reuse of components (Software Construction KA), considering a huge number of available components in the repository, selecting the right components could be difficult. The importance of individual components can differ in scale and technology. Research by Kenny et al. (2017) offers a way of weighing the relative importance of individual components within a system architecture.

Similar to OSS development, activities in ISSD are also performed by a community. Thus, each contributor must be able to collaborate with other stakeholders, e.g. customers, coworkers or suppliers. However, transitioning from a closed to an open source development puts software engineers under pressure as they need to embrace more open and collaborative principles. Current research on group dynamics and psychology (SE Professional Practice KA) mainly investigates the effect of such a transition and pressure in the community (PS1, PS4, PS30). Some studies focus on how to establish an effective community (PS20) or the collaboration pattern that can be implemented to support inner source (PS33) or increase innovation (PS13).

Our primary studies also reveal that the majority of empirical studies on inner source employ a qualitative approach, thus present qualitative results. Case studies and interviews are the main data collection method. Most studies on inner source have been conducted in established and large multinational organisations, with only two studies in a small and medium enterprises

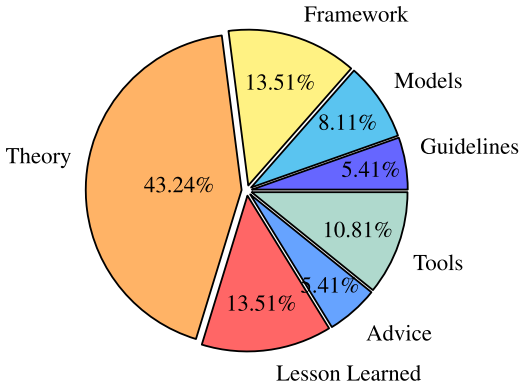

Fig. 4. Contributions of the primary studies.

(SMEs) context (PS7, PS36). Typically, the large organisations come from various business domains, e.g. engineering, software development, medical equipment or telecommunication. They have a long history with proprietary oriented and industrial/commercial mode of software development, and gradually incorporated OSS and methods into their internal development (PS30).

There is no predominant theory, framework, or model used to examine inner source. Table 10 presents a summary of the theories, frameworks, and models adopted in the primary studies and explains how they are applied to examine inner source. Among the 25 empirical studies, one fourth adopted a theory-guided approach to study inner source. For example, PS1 and PS10 used institutional theory on enrolling group interest (Scott, 2001) to explain how organisations evolve when implementing inner source. The study by PS12 used actor network theory (Latour, 1986) to identify which open source characteristics are lost in the translation to inner source, and which ones remain.

5.1.2. Contribution of the primary studies

The distribution of the contribution of primary studies is illustrated in Fig. 4. The main contribution of the primary studies was in the form of theory (16 studies) around an inner source approach. Nonetheless, 32.43% (12 studies) of the contributions provided concrete approaches that could be used to support an inner source approach. These approaches included frameworks or methods for implementing inner source (5 studies), e.g. progressive open source (PS2), corporate open source (PS6), models representing relevant concepts of inner source (3 studies) e.g. a theoretical model to promote software reuse (PS28), an inner source business model (PS29), an inner source maturity model (PS31), and tools supporting technical infrastructure of inner source (PS8, PS19, PS34, PS37). Other studies provided guidelines (2 studies), lesson learned (5 studies), and advice (2 studies) for implementing inner source.

For a successful technology transfer, academic research results are required to be validated statically or dynamically in real settings (Ivarsson and Gorschek, 2011). Static validation involves the 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 and to secure commitment to implement them across the organisation (Gorschek et al., 2006). Ten out of twelve frameworks/methods, models and tools had been validated in industry settings (PS2, PS6, PS8, PS19, PS21, PS25, PS28, PS31, PS34, PS37), while the remaining approaches (PS20, P29) are considered theoretical frameworks. The list of frameworks/methods, models, and tools are described in Table 11.

5.2. RQ2 – What Are the Main Factors that Characterise ISSD?

In this section, the characteristics of ISSD are analysed in detail using each category of the derived framework. The overall analysis is summarised in Table 12.

5.2.1. What (Transformation)

Based on our primary studies, terms like “inner source” or “inner source software” have been used interchangeably to describe the phenomenon of adopting OSS development practices within the confines of an organisation (PS22). While “inner source” refers to the phenomenon of the OSS practices development adoption, “inner source software development” refers to the actual notion of development of a specific inner source software (PS22). Synonyms for these terms are “progressive open source” (PS2), “corporate open source” (PS5), “internal marketplace” (PS19), “internal open source” (PS27).

Inner source is not a well-defined methodology, but rather a development philosophy, which promotes open collaboration, egalitarianism, meritocracy, and self-organisation (PS2, PS18, PS21, PS23). Inner source uses practices from OSS communities that will add value to the existing software development process, including (i) universal access to all development artefacts, (ii) independent peer review, (iii) informal communication channels, (iv) self-selection of tasks to improve, and (v) early feedback and frequent releases (PS23). However, the implementation of inner source in each organisation varies in terms of which and how the practices are adopted (Stol et al., 2011).

Research by Lindman et al. (PS10) argued that inner source does not resemble, what it is referred to as a “classical OSS movement”, which is based on voluntarism, peer-recognition and public discussion, but supports designated projects based on work contracts. In addition, costs are made visible and the sharing between units is based on agreement. The study also found that inner source changes the reward and communication structure, which result in more competition for resources between business units.

Inner source software refers to the software product that is developed within an inner source environment (PS22). The types of software developed in an infrastructure-based model are tools and utilities, e.g. compilers and shells, whilst in a project-based model, the types of software are business critical assets, e.g. platform in a software product line.

While collaboration has been seen as one of the main constructs of an inner source approach, few studies address on how to measure it. For example, PS2 and PS18 used the number of inner source projects in the portfolio. Moreover, PS9 and PS25 proposed the number of contributors in each inner source projects to measure collaboration. The research by Vitharana et al. (PS28) highlighted that when it comes to measurement in an inner source project, it needs to account the participation from developers across division or organisation units. To fill this gap, the recent research by Capraro et al. (PS32) proposed a patch-flow measurement to quantify collaboration. Patch-flow measures the contribution made by a developer, who is external to a project. A significant patch-flow indicates an intense collaboration, which means the development project actively receives contributions across teams and divisions.

5.2.2. Why (World View)

In order to stay competitive, organisations are looking for ways to improve their software development processes. One of the ways to do so is by imitating and extending the methods that have proven to be successful within the organisation (PS3, P12, PS22). Over the last decade, the success of OSS has gained much attention, in terms of software maintenance, code reusability and higher quality. As the OSS approach acceptance has increased

Table 11
Overview of the Frameworks/Methods, Models and Tools for Implementing Inner Source.

ID	Type	Description	Validation
PS2	Framework	Progressive Open Source (POS), consists of a three-tier model: (i) inner source – refers to the application of the open source approach and benefits developers within the corporate environment, (ii) Controlled Source - which is outside of the corporate firewall but restricts access to limited specific corporate partners, and (iii) Open Source - refers to the open use of the 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, and (iv) Establishing a COS Project – as the request for product-specific changes began to accelerate, others within the company started to contribute code and ideas.	Dynamic
PS8	Tool	Software forge (component library)	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, consisting 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 to gain an understanding of whether or not Inner Source is suitable.	Static
PS25	Framework	The framework is used to describe a development methodology from its origin to its practical application, and to compare two or more development methodologies.	Static and Dynamic
PS28	Model	Theoretical model to promote software reuse within the organisation.	Dynamic
PS29	Model	Inner source business model.	–
PS31	Model	Maturity model of inner source implementation.	Static
PS34	Tool	DLR Software Portal, a directory of existing software.	Dynamic
PS37	Tool	CALAIS – A Component Analysis Learning Algorithm for Inner Source	Static

Table 12
Framework analysis for ISSD.

What (Transformation)	
What defines a software product/project as inner source?	In general, inner source software is the product developed at an organisation that has adopted OSS development practices. The source code is open and visible for everyone in the organisation. The software is developed by developers across divisions or units.
What types of products/projects tend to be inner source?	Currently, the inner source approach has been used to develop discrete software packages (e.g. utilities, tools, compilers, shells, etc.) and business critical assets (e.g. platform of a software product line).
Why (World View)	
What are the technological motivations for ISSD?	The technological drivers for inner source include improving product line engineering and promoting the reuse of software to build high quality software and supporting global software development.
What are the economic motivations for ISSD?	Business drivers for inner source include the need for shared software development cost, effort and risk.
What are the socio-political motivations for ISSD?	Human motivation for inner source include the desire for meaningful work through the opportunity to contribute and knowledge sharing, peer reputation, and the desire to develop valuable technical skill.
When and Where (Environment)	
What are the temporal dimensions of ISSD?	ISSD is characterised by time-based releases and less frequent releases.
What are the spatial/geographic dimensions of ISSD?	Similar to OSSD, ISSD is characterised by distributed developer teams across organisational and geographical boundaries.
How	
How is the ISSD process organised?	An inner source program can be established as an infrastructure-based and a project-based model. In infrastructure-based, an organisation provides the required infrastructure to support the development process, whilst in project-based, a core team is responsible for developing the share assets (e.g. the platform for product lines).
What tools are used to support the ISSD?	Tools typically used in OSS projects are also used in an inner source environment, e.g. web servers, wiki-tools (for knowledge sharing), mailing lists (for question and answer with architects), issue trackers (for report and communication problem), and code archives.
Who (Client, Actor, Owner)	
What are the characteristics of the individual developers contributing to inner source projects?	The contributors of ISSD are typically developers who are familiar with OSS and with an interest in the project. Depending on their contribution and level of knowledge, any developer can be either a benevolent dictator or trusted lieutenant.
What are the characteristics of the organisations adopting inner source?	Typical organisations that adopt inner source are established and large organisations that have been using or are involved in OSS and develop a range of software products mainly to large customers that require confidentiality, high quality and predictable delivery.
What are the characteristics of the users of inner source software?	To date, inner source software users have primarily been everyone within the organisation who is interested in the product or project. Expert users can contribute further on the development process.

(Fitzgerald, 2006), organisations have incorporated open source into their existing development process (PS26). Even though there is no standard set of OSS practices, many organisations are adopting OSS practices within their settings. In addition, several success stories and lessons learned have been reported in small and large organisations that have adopted open source practices internally.

One of the technological motivation for an inner source approach is to address the issues related to platform-based engineering (PS17) or product line engineering (PS25, PS26). Product line engineering enables organisations to reduce time-to-market and cost, but with traditional platform development, a platform has a potential to become a bottleneck when multiple teams use

the same platform (Oor and Krikhaar, 2008). Based on case study research, Riehle et al. (PS17) identified three main issues related to product line engineering: (i) lack of resources, (ii) product unit power play, and (iii) insufficient collaboration. While the study also found that the inner source approach helps organisation to address such issues, it is often difficult to establish inner source projects for three reasons: organisational changes, psychological challenges and process breakdown (PS17, PS29).

Our primary studies also show that organisations decide to adopt inner source to increase software reuse (PS2, PS3, PS19, PS28). In a large organisation, where divisions or business units are typically working in a silo, they may be unaware that some of them are developing similar software modules, or have finished them. Promotion of reusable modules or components is built on creating software repositories, which give developers access to locate and assess their functionalities.

The common business drivers for inner source are to reduce software development cost, effort and risk (PS6, PS7, PS10, PS17, PS20). In a project-based inner source model, the development cost is shared among the number of divisions using the common asset. Each division contributes existing tools, support staff, processes, assistance and leadership to project. Research by Gurbani et al. (PS6) shows that by selecting a lead group with an existing infrastructure for the software development and a willingness to participate will require a small investment to establish the common assets.

Another motivation to transition to an inner source environment is to facilitate the contribution and sharing of knowledge (PS8, PS29). In the case of Philips Healthcare (PS29), the domain engineering group who provided the platform based on a common architecture did not possess the knowledge to develop every shared component. Moreover, the requirements from the application engineering group competed for priority. In this case, moving to an inner source approach allowed for the optimal sharing of the assets and gave more control to application engineering over platform development.

Linden et al. (PS26) argued that the complexity of software development has changed. Software is no longer produced by a single group but in close collaboration across an organisation and beyond the organisation's borders. For most products and product lines, only 5-10% of the software makes a difference from competitors' products. Thus, organisations must identify a suitable time to change how they collaborate. As an intermediate step toward integrating open source into product line, organisations can adopt inner source software.

Gaughan et al. (PS3) characterised an inner source approach and identified a list of motivating factors for organisations to implement an inner source approach. Their study found that an inner source approach is mainly implemented for supporting global software development or promoting software reuse. Inner source enables flexibility in initiating, terminating, and changing collaborations, in timing of and setting priorities for development teams across organisational and geographical boundaries (PS27).

5.2.3. When and where (Environment)

Large organisations are typically organised into several business units or divisions. In each division or units, there may be smaller units or groups that are working on a specific task. In software development, one key problem with these silos is a duplicative development, where multiple teams work on similar functionalities (PS33). Silo-oriented organisations also limit their engineers to collaborate with others from different divisions. Thus, opening up collaboration via inner source empower engineers for better synergy and innovation (PS33).

Unlike in OSS, research by Stol et al. (PS23) shows that inner source is characterised by time-based releases. Typically the products developed in an inner source environment consist of millions lines of code. With a reliable release schedule, developers are not rushing to include their work into the next release, but rather have more time for integration testing and resolving issues from users, which will build the users' confidence in the project quality.

Gaughan et al. (PS3) conducted a case study on the use of inner source software process in a global SE context. Geographically, inner source is characterised by massive distribution, with teams, community, and peer-groups defined by both virtual and physical boundaries. They found that a common area for choosing inner source is that of research or emerging technologies.

5.2.4. How

Gurbani et al. (PS6) described two models to manage ISSD: infrastructure-based and project-specific. In an infrastructure-based ISSD model, an organisation provides the critical infrastructure (e.g. web servers, mailing lists, code archives, wiki-tools, etc. – called software forges (PS18)) to allow developers to host individual software projects. Several studies have reported the implementation of this model in various organisations e.g. HP (PS2), US DoD (PS5), Nokia (PS9), SAP (PS18), and InstaDef Ltd (PS19). In a project-specific ISSD model, the critical resources or assets are maintained by one division within the organisation, called the core team. The team has the responsibility for sharing the assets across the organisation. Several studies have investigated this model in Alcatel-Lucent Technologies (PS6) and Philips Healthcare (PS29).

Research by Capraro and Riehle (2017) found the infrastructure-based model can be differentiated into a selective and universal model. In the universal model, all software artefacts are published as inner source assets. This model creates a significant impact on the organisation since every component is made available across the organisation. In the selective model, only selected artefacts are visible. To assist organisations evaluating their efforts in implementing inner source, the research by Eckert et al. (PS31) proposed an inner source maturity model based on CMMI-DEV. The model has four capability levels (incomplete, performed, managed, and defined) and three dimensions (people, procedures and methods, and tools and equipment).

In an organisation where each development project is treated as a separate entity, technology selection will remain a project silo and the experience of using these technologies will be embedded in the developers (PS19). In such conditions, the concept of software forges helps organisations to promote software reuse across teams and projects. Ripatti et al. (PS19) developed an internal marketplace to increase the amount of reuse within the organisation. The marketplace is inspired by Google Play and Apple's App Store and used to market components developed in-house and COTS across the projects. However, there is a need to govern the forge otherwise it will be cluttered with components that no one will use (PS8).

The ability of organisations to move to an inner source approach depends on several factors: the ability of management and workers to understand the OSS philosophy, the development of mutual trust between management and workers, the workers' perception of being involved in a challenging and innovative project and the motivation of workers to participate in such projects (PS20). Stol and Fitzgerald (PS23) recommended nine key factors for organisations to consider when adopting inner source: seed product, the stakeholders, modularity, development practices, quality assurance, tools, coordination and leadership, transparency, management support and motivation.

Bank et al. (PS33) described five patterns for collaboration in inner source organisations: 30 day warranty, dedicated community manager, review committee, common requirements, and improve find-ability. A 30 day period is established to decide whether to accept the contributed code goes to production or to reject. A

Table 13
Benefits associated with inner source.

High quality software product	Authors' reputation increased as they share high quality code (P2, P3, P17)
	The code is visible to the community, thus allows sharing community debugging (P2, PS33)
	Community provides feedback to developers (PS7)
Effective and efficient software development process	Reduced development time and time-to-market (PS2, P17, PS20, PS21, PS31)
	Reduced maintenance cost and effort (PS6, PS7, PS10, PS17, PS20, PS35, PS36)
	Increased number of parallel development processes (P12)
	Increased innovativeness (PS13, PS18)
	Avoids duplication of work and promotes the reuse of software (PS2, PS3, P10, PS12, PS13, PS17, PS19, PS28, PS34)
	Better requirement comprehension and prioritisation (P17)
	Rapid re-deployment of key developers (PS2, PS12, PS21)
Enhanced knowledge management	Fosters knowledge sharing and transfer across community (P12)
	Increased information availability and visibility (PS25, PS33)
Improved employees motivation	Defines an entry path for newcomers (PS25)
	Opportunity to develop professionally valuable technical skill (PS5)
	Higher job satisfaction through improved relationship with colleagues (PS17)
	More meaningful work (PS17)
	Improves developers' reputation (PS17)

dedicated community manager spends 100% of this time to nurture and grow the community. A review committee consists of senior managers of all business units which participate in the inner source initiative. They are responsible for managing inner source projects across organisations. Common requirements aim to align the requirements so that the code will satisfy the need for many projects. The code could be refactored into smaller pieces and the projects that have similar requirements might agree on it. Improving find-ability concerns on providing guidelines and conventions for how to store the codes in the repository.

5.2.5. Who (Client, Actor and Owner)

In a project-specific inner source initiative, there are several roles involved: benevolent dictator and release advocate (PS6) and trusted lieutenant (PS23). Benevolent dictators are typically the author of the original code, who control the code base to ensure the incoming contributions and proposed features match with the architecture principles. Trusted lieutenants are the ones that assist benevolent dictators. Release advocates ensure that the code changes for all features are submitted on time and keep track of all business division-specific impacts for particular releases (PS6). In an infrastructure-based inner source model, software projects are owned and maintained by the individual project's creator (PS22). As the number of contributors grow, maintenance is therefore performed by the community.

Unlike a closed approach, inner source projects involve flexible coordination and leadership (PS23). One of the key tenets is meritocracy, thus any developers who give significant contributions and have deep expertise can become trusted lieutenants or benevolent dictators'. However, there is no rule of thumb for which roles are needed. It depends on the organisation's context and needs (PS23).

At an organisational level, inner source has been adopted mainly by established and large organisations with OSS experience. These organisations develop a range of software products to large customers. At a project or product level, there are four requirements for a project or product to gain benefits from using ISSD (PS6): (i) a technology is needed by several product groups, (ii) the technology is immature, so that there is an opportunity to evolve continuously, (iii) the product groups have different needs and specific expertise in customising the technology for their needs, and (iv) the initial product has a sound, modular architecture. The importance of seed product is also recognised by Stol et al. (PS21). The availability of basic architecture and running version can attract users who later become the contributors.

5.3. RQ3 – What are the Reported Benefits and Challenges Associated with ISSD?

In this section, we present the reported benefits and challenges associated with ISSD from our primary studies. Tables 13 and 14 list the summary of the reported benefits and challenges and their reference sources.

5.3.1. Benefits associated with ISSD

The most common benefits with ISSD reported across the literature include reduced costs and efforts required in development and maintenance, and the promotion of software component reuse. Indeed, in many cases the potential benefits of ISSD are often similar to that which organisations reaped rewards from the adoption of OSS (Hauge et al., 2010; Höst and Oručević-Alagić, 2011).

As discussed in Section 5.2.5, there are four requirements to enjoy the benefits of ISSD (PS6). For example, a product is immature, even though it is used by different users. Moreover, the product has modular architecture, and the expertise to customise the product is available in-house. If an organisation's environment identifies the four requirements, under the right circumstances, organisations can benefit from ISSD approaches. We can further categorise the benefits of ISSD into (i) high quality software products, (ii) effective and efficient software, (iii) enhanced knowledge management, and (iv) improved employees motivation. For example, we have identified that for organisations to attain the benefit of a high quality software product, there is a much wider impact on increasing the author's reputation by sharing high quality code. Dinkelacker et al. (PS2) describes the benefits of "openness" to developers within the corporate environment to sustain the community of developers. ISSD also enjoys benefits of improved code quality, building capabilities for community debugging software reuse, faster deployments, wider personnel awareness/visibility of codebase, and higher job satisfaction (PS2, PS3, PS18).

The sense of community across an inner source environment plays a critical role in providing feedback to developers and synchronising activities (PS7) and ultimately reduce development time and time-to-market (PS2, PS18, PS20, PS21). By reducing time and improving communication and collaboration, ISSD also enjoys the benefits of lower maintenance costs and efforts (PS6, PS7, PS10, PS17, PS35, PS36). Therefore, ISSD can bring about additional benefits in delivering an effective and efficient software development process (PS12) and increased innovativeness across teams (PS13, PS18). From an organisational perspective, capturing the collaborative interactions between teams also encourages managers to enhance knowledge management capabilities. This entices develop-

Table 14
Challenges associated with inner source software development.

Integration and architecture	Interfaces of components in the shared assets are not well specified and documented (PS22)
	Missing functionality in the newly delivered components (PS22)
	Balancing refactoring and requirements (PS5, PS22)
	Merging independent changes done across two development lines (PS5)
Knowledge management	Lack of documentation and specialised domain knowledge (PS22)
	Maintaining coding standard and training new developers to maximise utility of ISSD (PS2, PS5)
	Risk of losing intellectual property rights (IPR) or other strategic product information (PS26)
Migration and Usage	Migration from existing tools and infrastructure (PS2, PS5, PS22)
	Appropriate IT support to maintain software forge (PS2)
	Difficulty in using shared assets, configuring is complex (PS22)
	A proper search and navigation infrastructure (PS2)
Management	Managing the appropriate skill set at the corporate level (PS2)
	Encouraging commitment, transparency and knowledge exchange (PS13)
	Lack of guidance for implementing inner source (PS31)
	Lack of measurement for evaluating inner source initiative (PS32)
	Managing and balancing the constructive and deconstructive tensions (PS30)
Culture and community	Cultural resistance to change (PS11, PS14, PS30)
	Getting people to invest time or effort in sharing and building skills and knowledge outside their own domain (PS13)
	Lack of a clear unifying vision so people do not drift in different directions and send out mixed messages to the rest of the teams (PS13)
	Reluctant to accept contribution, to contribute and treat core team as traditional supplier (PS22)
Security and transparency	Requires appropriate authentication, authorisation and audit mechanism to properly control access to source code (PS2, PS3)
	Fear to use digital media to record idea generation and brainstorming in a meeting (PS3)
	Job security due to sharing or showing a substandard code or system (PS3)

ers to engage in knowledge sharing and transfer across the inner source community (PS12, PS25, PS33). It is also widely reported that ISSD can improve employee motivation in terms of improved guidance (PS25), opportunities to up-skill in new technical areas (PS6) and lead to overall higher levels of job satisfaction through inner source development communities (PS17).

5.3.2. Challenges associated with ISSD

ISSD also suffers from a number of key challenges in organisations. These are typically related to integration and architecture, as well as building the right culture and effective inner source community. For example, in terms of the challenges in integrating OSS in the product development, Stol et al., (PS22) identifies product selection, documentation, community, support and maintenance, integration and architecture, migration and usage, and legal and business factors. They compare these to challenges of inner source to include documentation and knowledge, community, support and maintenance, integration and architecture, migration and usage. In general, these challenges are similar to those of OSS and mapped onto inner source except for legal and business challenges. In addition, Gurbani et al. (PS6) describe the challenge associated with software architecture. For example, there are challenges in merging independent changes carried out across two development lines since each line has features and bug fixes that another team may need. In terms of tool compatibility, Gurbani et al. (PS6) also described how challenges typically emerge in keeping different repositories synchronised between individuals and teams.

Knowledge management plays a vital role in sustaining an inner source community. However, knowledge management also presents several key challenges. Specifically, the lack of documentation and specialised domain knowledge (PS22) to provide inner source community members with the "how-to" knowledge and design documentation to use the software in a meaningful way. In addition, maintaining coding standards and training new developers to maximise utility of inner source becomes a challenge if organisations fail to implement a sufficient knowledge management resources (PS2, PS5). This can have negative consequences for organisation if interacting with external parties as part of this inner source strategy. For example, van der Linden et al. (PS27) describes how sharing creates the risk of losing intellectual property rights (IPR) or other strategic product information. Van der Linden et al. (PS27) caution managers to be aware of what knowledge is made in open source and now to hamper internal strategic innovation efforts. Research also indicates that it can be difficult to use the shared asset which ought to be usable by different business divisions in different specialty domains (PS22). This generates many new challenges for the migration process from existing tools and infrastructure (PS2, PS5, PS22).

Wesselius (PS29) and Riehle et al. (PS17) describes how it is often difficult to establish inner source projects for three reasons relating to organisational changes, psychological challenges and process breakdown. Indeed, from a management perspective, ISSD brings about new challenges, for example, in terms of recruiting the appropriate skill set at the corporate level (PS2) and identifying ways to encourage team commitment, transparency and knowledge exchange (PS13). More importantly from a management perspective, managers often lack guidance for implementing inner source (PS31) and measurement capabilities for evaluating inner source initiatives (PS32) to communicate its benefits to other stakeholders. Managing teams often requires strong leadership to leverage constructive tensions and mitigate deconstructive tensions that may exist in the developing or sustaining of an inner source community (PS30). Regardless of the management approach, managing the overall inner source culture and community is challenging. For example, change is often met with resistance (PS11, PS14, PS30). Therefore, to address this challenge and encourage individuals to invest their time and effort in an inner source initiative, it often requires that managers successfully promote a clear vision with a single message on inner source across various teams (PS13, PS22).

As with any change to business and IT operations, ISSD also presents new challenges around security and transparency. For example, given the openness of source code and the collaborative nature of inner source teams, appropriate authentication, authorisation and audit mechanisms are required to properly control access to source code (PS2, PS3). In addition, identifying new approaches to capture innovation is considered challenging as there can be a fear of using digital media to record idea generation and brainstorming in meetings (PS3). With such openness and sharing

Table 15
Example future research question for examining inner source.

Research Cluster	Example of Research Questions
Inner Source Adoption	How to architect software in ways appropriate for different styles and organisational settings? How to improve interaction and contributions of each stakeholder in the community? How do inner source communities evolve over time? What are the existing strategies for transitioning to an ISSD and how effective are they? What are the factors that dictate why and how organisations select specific projects for inner source approach? What strategies can be employed to overcome competing tensions of openness in closed software development environment?
Supporting Inner Source Activities	Are there any specific OSS-based practices that can be used in an inner source environment? What are any community-based practices that can be used in an inner source environment? How can an inner source community effectively communicate with their members? What are the existing process/strategies/methods to reduce risks associated with inner source?
Applying Inner Source Approach in Various Context	Are there different types of inner source implementation, e.g. differentiated by size, technology sectors, country economy or other factors? How can inner source be tailored to suit environments it was not originally designed to support, e.g. startups, safety critical domains, regulated environments?
Management and Governance of Inner Source Approach	Is there a common cultural/organisational/team characteristic among inner source communities? What metrics needs to be collected to better understand and manage inner source? How is value created and captured within an inner source environment?
Methodologies and Theories for Inner Source Research	What theories, framework, metrics and other instruments from existing related bodies of knowledge can be applied to inner source research? How can these be effectively applied to improve the implementation of inner source in practice and the study and improvement of inner source research?

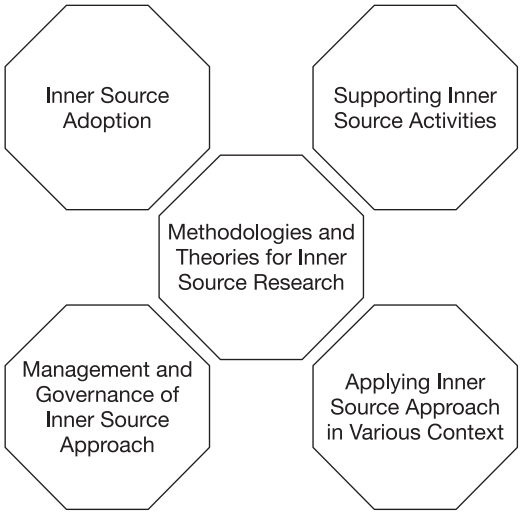

Fig. 5. Overview of the inner source software development research agenda.

of code and expertise, there can be a growing sense of insecurity among developers. Therefore, ISSD can present new challenges around job security due to the sharing or showing of a substandard code or system (PS3) which also needs to be managed within an inner source environment.

5.4. RQ4 – What are the Research Gaps in the Current Research on Inner Source?

The previous sections have shown that as more and more organisations have adopted inner source to improve their internal development processes, there is a strong need for further research into ISSD. Therefore, we establish a research agenda, which is grouped into five clusters, shown in Fig. 5. This research agenda draws on the findings, the analysis, and limitations of the primary studies. While this grouping is one of the several possible ways to create the clusters, it served the purpose of easing the presentation and discussion of the research agenda.

Inner Source Adoption (Section 5.4.1) covers research tracks that investigate factors related to the adoption of an inner source approach in an organisation. Supporting Inner Source Activities (Section 5.4.2) encompasses research foci that address specific SE challenges encountered by organisations over time. Applying Inner Source Approach in Various Context (Section 5.4.3) seeks to strengthen innovation by extracting successful inner source practices and integrating them in various environments. Management and Governance of Inner Source Approach (Section 5.4.4) covers research tracks that investigate management practices to govern inner source. Finally, all of these research areas are connected by research tracks that develop methodologies and theories for inner source research (Section 5.4.5). The example research questions are provided in Table 15.

5.4.1. Inner source adoption

As in automated factories, people in large organisations are trained to do prescribed and specific tasks reliably. Hence, resistance will emerge from any endeavour to change the status-quo. This is also the case for inner source adoption (PS1, PS2). Introducing the concept of openness in an established corporate culture and closed software development environment can be viewed as a risk thus should be implemented with care. The visibility brought about by inner source is perceived as an opportunity by some developers but also a risk for others (PS6). As shown in Table 14, there are a number of challenges associated with an inner source adoption in organisations. For example, the adoption of inner source in organisations suffers from the leadership, management, culture, and community related issues. Adopting inner source requires commitment from all stakeholders, not only from developers but also leaders and managers since it will affect both roles (PS1). In addition, it would be more valuable to learn from mistakes or failures during the adoption or transition to inner source. For practitioners, it will be an alert to be more cautious while implementing inner source initiatives.

The transition from close to open collaboration raised tensions among the stakeholders in organisations (PS30). Such organisations are increasingly under pressure to embrace open and collaborative principles, while at the same, to time keep operating in a tight and controlled manner. The constructive and destructive role that competing tensions play in hampering and driving inner source practices. Hence, it is important to find strategies to manage and balance such tensions, for example to reduce deconstructive tensions and regard constructive tensions.

As more projects and developers are using software, each one wants to tailor it to its own process and specific needs (PS5). What results is a process diversity, with a kaleidoscope of processes

within an organisation to support every individual and teams, including dozens or even hundreds of methods, method variants, sub-routines and variants or routines, deviations and improvisations, all of which are simultaneously alive and interacting with each other. Thus, it would be valuable to improve our understanding of how to allow such tailoring and customisation to support an open source development style, while still preserving the core architecture.

In a project-based model, business divisions have access to the shared assets and modify them accordingly. This could be very helpful when the core team lacks domain knowledge about the requirements that are owned by a business division. However, the research by Stol et al. (PS22) found that this is not the case. The core team who maintains the assets architecture suffers from the "not developed here" syndrome. The contribution might not fit the architectural principle and lead to a change on the original architecture. When this happens, it will create a tension between fulfilling business division needs and performing architectural refactoring (PS5, PS23, PS29). Thus, it would be interesting for research to find ways to improve interactions and contributions of each stakeholders in the community.

5.4.2. Supporting inner source activities

The research track in this cluster shares the theme of studying, identifying, transferring, and evaluating frameworks/methods, tools and models aimed at supporting ISSD activities. Our findings show that while there is a great concentration of empirical research on the study of how organisations adopt an inner source approach into their internal software development processes, other research areas receive much less attention. One of the implications of these findings for research is the need for more empirical studies on engineering practices to support an inner source approach.

The study by Capraro and Riehle (2017) identified four inner source development practices: participatory reuse, self-selection tasks, volunteering, and collaborative development projects. However, our SLR results show that only few studies provide detailed insights of practices that support inner source (PS22). An answer to this issue could help practitioners to develop inner source context specific engineering practices. Specifically, while the inner source approach is highly influenced by the open source approach, there is a need to translate OSS-based practices that suit an organisational context to achieve the many benefits associated with open source (PS10).

Future research can also investigate how emerging tools influence the developers' performance in achieving their goals and communication. Communication and coordination mechanisms are important for monitoring work processes and thus sustain inner source communities across the organisation (PS28). As the popularity of microblogging tools such as Twitter has attracted developers, research in this area would help develop an understanding of how these tools contribute to establishing and improving inner source communities (PS22).

The concept of software forge has been suggested as a way to promote software reuse internally (PS19, PS25, PS28). Software forge is a web-based collaborative software platform for sharing software artefacts. In the case where various contractors or vendors are employed within different areas of the organisation, giving them access to code for them could increase a security risk as the organisation does not have control over what the vendors are using. Solving this issue will help the organisation to mitigate and minimise risks associated with inner source while still encouraging collaboration with organisational partners (PS3, PS14).

5.4.3. Applying inner source approach in various context

The research track in this section proposes to apply inner source concepts in various contexts. The idea of extracting a concept from one context and applying it in another has proven successful, such as in SLRs and open source principles (Unterkalmsteiner et al., 2016). Our primary studies indicate that the majority of studies on the inner source approach are conducted in the context of large organisations, which have a long history with OSS projects. However, the adoption of inner source approach in large organisations requires a significant effort in changing the culture and transforming the way of working. Therefore, applying an inner source approach in various context e.g.SMEs, startups, or other non-software contexts seems a promising avenue to improve the development activities.

The existing inner source frameworks/methods, models and tools as shown in Table 12 are developed and tested in a specific environment. For example, the corporate open source model (PS5) was developed in a telecom company to implement IETF session initiation protocol (SIP). When they started the work, the protocol was already at a mature standard. Scholars have called for more studies to validate the results of existing research in diverse contexts to improve their generalisability (PS5).

5.4.4. Management and governance of inner source approach

The research track in this cluster shares the theme of studying, identifying, transferring and evaluating frameworks/methods, tools and models aimed at managing and governing ISSD activities. The issue of community is still an important area in inner source. Future research direction could examine the concept of community (PS3). In general, the community consists of all developers and staff who interact with code. It is still unclear whether the community should have certain traits to be able to support inner source. Answering this question is necessary to understand how to effectively manage an inner source community across an organisation.

Another direction for future research is related to the metrics for managing inner source projects (PS29). Our literature review findings show that the emerging literature has focused on how ISSD enables organisations to improve software development efficiencies through a set of shared assets (PS29), but it is still unclear as to how organisation assess or measure such improvement. Different constructs have been proposed by Vitharana et al. (PS28) to measure the impact of inner source on software reuse. However, more studies are called for to precisely define the metrics and validate them in a real-life setting. In particular, how an inner source approach creates business value for the organisation is still unanswered.

5.4.5. Methodologies and theories for inner source research

The track in this cluster directs their research towards identifying the means to better study and understand ISSD. Theories are important in any scientific field, as they present a systematic way of understanding a contemporary phenomenon. Inner source research does not operate in a vacuum, but rather can borrow theories from areas such as SE and IS, business and management, as well as from the fields of organisational and social science.

Our literature review has identified a number of theories that have been applied in the context of inner source research, including OSS adoption theory, institutional entrepreneurship, open innovation, ANT, knowledge management, and grounded theory. Theorising the inner source approach is important since there is lack of cohesion, cumulative tradition and clarity. Theoretical advancements need to be achieved so that researchers can make better sense of diverse contexts and situations. This leads to the questions such as 'what theories or framework from an existing body of knowledge can be applied to inner source research?' More importantly, 'how can these theories be effectively applied to improve the implementation of inner source in research and practice?' By answering these questions, this research track would position inner source at the core of academic research, whilst also providing

practice with empirical evidence on the utility of inner source in a diverse environment.

6. Discussion

In the past decade, several publications have discussed the use of the inner source approach for internal software development process improvement in organisations. We identified 37 primary studies presenting 25 empirical research (67%) on inner source approach in various contexts, including SMEs and multinational organisations. In general, our primary studies were of good quality, in terms of scientific rigour and industry relevance (see Section 4.2). We also observed how inner source research has evolved and possibly matured in some SE KAs. This makes it possible to identify changes in the inner source research direction for the last 20 years.

The need for an effective and efficient software development process has been raised since the 'software crisis' in 1968. While various generic and specific processes have been proposed to improve development processes, these processes are constantly adapted and extended to meet the changing needs. Requirements for process and methods vary between individual projects, for example development context, delivery, project team, deadline, etc. Thus, organisations need to tailor various methods by selecting and integrating different method fragments into congruent and consistent methods relevant for the situation at hand (Brinkkemper, 1996; Lindvall and Rus, 2000). This is also the case for inner source.

Various models, frameworks, and tools for implementing inner source, for example Corporate Open Source or Progressive Open Source, have been proposed in the literature. The successful implementation of these frameworks within the origin organisations has gained interest from other organisations to adopt and replicate them. This could be the reason why most research on inner source belongs to the SE process (Table 9). This is the only matured KAs, as it covers all five topics. Specifically, all studies under the software life cycle topic address the adoption and adaption of inner source in various contexts.

Within the SE professional practice KA, group dynamic and psychology is the second topic that received most interest from research (7 studies). It is understandable, as implementing inner source can be a quite challenging process. As each organisation is unique, what defines success factors and challenges might be different. Our primary studies show that a large number of challenges are related to the dynamics and psychology within the inner source community and the organisation itself. It is important to provide recommendations for this challenges, because at the end the success of software development activities is determined by humans.

Our SLR results allow us to analyse the characteristics of the inner source approach. We adapted the framework by Feller and Fitzgerald (2000) to analyse the inner source phenomenon. We observed two important aspects of inner source highlighted in the literature: the adoption of open source practices (e.g. access to repository, fishbowl development environment, peer-review, etc) for internal development and collaboration among developers across business units or divisions. Different types of collaboration patterns have been proposed in the literature and seems proven in some organisations (PS33). Because each organisation is unique in terms of the culture, the product architecture, and the history of software development processes, these patterns need to be tailored prior to an adoption in a different context.

The research by Capraro and Riehle (2017) found 43 publications in inner source from 2001 to 2014, which 21 of them were peer-reviewed papers. Building on previous efforts, our study found 37 primary studies, which were peer-reviewed papers and published in conference and journals. We also found 9 studies on inner source published after 2014. This means that our systematic review process updated their findings by identifying 16 additional peer-reviewed papers.

In terms of benefits and challenges of inner source, our SLR results are aligned with the review by Capraro and Riehle (2017). The findings have shown that the adoption of ISSD helps organisations to improve better quality, time-to-market and innovativeness. The transition to inner source also has challenges related to the culture, security and transparency, and management. 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 (Brown et al., 1989). Thus, the set of benefits and challenges of ISSD are becoming an early warning system for organisations so that they are aware of its potential benefits and impediments.

6.1. Summary of contributions

Our contributions are four-fold. First, the review incorporates both SE and IS literature to provide an in-depth overview of inner source research reported in the literature to date. Specifically, our review shows how inner source research changes over time by mapping the current research themes to KAs in SWEBOK (Table 9). Our review shows that even though inner source is not a well-defined methodology, the majority of research centred on its adoption and adaptation on various contexts and the dynamics of the inner source community. Only a small number of papers address the core of SE activities in inner source such as software requirements, software testing, and configuration management. One could argue that some of KAs e.g. software design KA or software testing KA may be of less interest for inner source or some categories could be more relevant than others. Thus, we call for more studies to address this gap.

Secondly, our study presents a methodological contribution by applying the OSS framework (Feller and Fitzgerald, 2000) as the lens to analyse the inner source phenomenon. The framework assisted us in maintaining a balanced examination of inner source approaches from five different perspectives, i.e. 'what', 'why', 'when and where', 'how', and 'who'. The study by Capraro and Riehle (2017) seemed to be geared toward the 'what' and 'how' of OSS framework's concepts (Feller and Fitzgerald, 2000). However, since inner source is the limited application of OSS within organisational boundaries, some elements of the framework are not relevant to inner source. For example, in the category Who (Client, Actor, Owner), rather than asking about organisations distributing OSS products, we asked about the characteristics of organisations adopting inner source. Our results show that organisations will gain more benefit with inner source when they are familiar and have been involved with OSS projects. Moreover, we also observed that the framework could be extended by asking about the inner source community, which is the key element of an inner source project. However, as described in Section 5.4.4, there is a lack of research in this area. Thus, we need more study to shed light on how to manage an effective inner source community in an organisation.

Third, our review updates the challenges from management perspective identified in the study by Capraro and Riehle (2017). First, there is lack of guidance to implement inner source, as the existing framework/method and models are specific to a particular context. Secondly, there is lack of measurement to evaluate the inner source initiatives. This makes organisations measure too little, measure wrong things or not measure inner source initiatives at all. Third, as organisations are moving from closed to an open software development, unforeseen organisational tensions may arise and ultimately prevents them from continuously improving software practices.

The final contribution is the mapping of a research agenda to further knowledge on the inner source approach. We identified 5 research themes based on the findings of our SLR study, as discussed in Section 5.4. Our review also identifies the existing theories used and its application to study the inner source phenomenon (Table 10), the contribution of existing research (Fig. 4) and the existing frameworks/methods, models and tools for inner source (Table 11). In addition, we also map the current research to the KAs in SWEBOK (Table 9). Researchers in the inner source or open source area could use these all together to position and align their own work in this area.

6.2. 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 the validity of this study are identified and discussed.

6.2.1. Publication bias

Publication bias is a problem that positive research outcomes are more likely to get published than negative ones (Kitchenham and Charters, 2007). It can be more of a problem when methods or techniques are promoted and sponsored by influential and reputable organisations. In such cases, organisations tend to publish positive results that support the new method or technique (Kitchenham and Charters, 2007). To address this threat, we did not restrict the sources of primary studies to a certain publisher, journal or conference. Thus, it can be assumed that the breadth of the field is covered sufficiently. However, we had to considered the trade-off of considering as much literature as possible, while at the same time accumulating and extracting reliable information. Therefore, we decided not to include grey literature, e.g. work-in-progress, books, technical reports, blogs, unpublished or not-peer review articles.

6.2.2. Threats to identification of primary studies

The strategy to build the search string aimed to retrieve as many studies as possible related to inner source. Therefore, the main metric to decide about the quality of our search string is the recall of the search results. Recall is referred to as the ratio of the retrieved relevant items and all existing relevant items (Saracevic, 1995). The recall of the search string was estimated by conducting a pilot search as described in Section 3.1. This showed an initial recall of 75 %, and after a refinement of the search string, a recall of 100 %. Nevertheless, the threat of missing relevant articles still exists. The use of different combination of terminology with respect to the exercised search string may have biased the identification of primary studies (Wohlin, 2014).

To further decrease the probability of missing relevant studies, we did not restrict the review to SE and computer science related areas only, but also included literature from other related domains, e.g. management, business and economics. However, we were aware that our search string would yield a large number of potentially irrelevant studies, and lead to a low precision.

We did not attempt to optimise the search string for precision. Precision refers to the ratio of retrieved relevant items and all retrieved items (Saracevic, 1995). As shown in the final results, our precision is 0.26 %, considering 14,293 unique papers (without duplicates) and 37 primary studies. This is an expected result, as retrieving more relevant papers (high recall) usually correlates with an increase of irrelevant items (low precision) (Raghavan et al., 1989). The low precision causes more effort to select the primary studies. Our strategy to address this threat is discussed in Section 6.2.3.

Table 16
Cohen's Kappa results of selection process.

Reviewers	Cohen's Kappa Coefficient		Agreement Level	
	Pilot	Actual	Pilot	Actual
R1 & R2	0.14	0.39	Slight	Fair
R1 & R3	0.25	0.56	Fair	Moderate
R1 & R4	0.30	0.79	Fair	Substantial

6.2.3. Threats to Selection of Primary Studies

The purpose of defining a review protocol is to reduce reviewer bias (Kitchenham and Charters, 2007). Review protocol describes selection (inclusion and exclusion) strategy. A well-defined review protocol improves consistency in the selection and extraction when the review is performed by more than one reviewer. To limit subjective bias for an individual reviewer, each paper was reviewed by two reviewers when applying inclusion/exclusion criteria. Before the actual selection was undertaken, a pilot with 50 papers was performed to see the agreement level among the reviewers. Cohen's Kappa statistics was used to assess the strength of agreement among the reviewers as suggested by Kitchenham and Charters (2007). Table 16 shows the kappa coefficients during the pilot and the actual selection, among the first reviewer (R1), second reviewer (R2), third reviewer (R3) and fourth reviewer (R4). Therefore, during the pilot phase, the Kappa coefficient among all reviewers were low. Any dissimilarity in assessment between reviewers was discussed in the presence of all reviewers. As the results show, the inter-rater agreement level among the reviewers were improved during the actual selection.

7. Conclusion and future work

Influenced by the success of OSS development, ISSD is gaining more attention from both academics and practitioners. This approach allows organisations to deliver high quality software products in a shorter timeframe and reduce costs and time to market. This paper attempts to aggregate and summarise an in-depth review of ISSD in both SE and IS literature.

The review establishes the state of the research on ISSD in terms of focused knowledge areas and contributions. Majority of research centred on the adoption and adaptation of inner source in various context, while other KAs in SWEBOK receive less attention. Thus our review calls for more studies on these areas to advance our knowledge on the inner source phenomenon. Furthermore, to advance our understanding of inner source, 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 inner source. With the help of the OSS framework, our study also has identified the characteristics of the inner source phenomenon, including its types of projects or products, the motivation for adopting inner source, its environment, inner source processes and tools and the actors involved in inner source. We also highlight the challenges as well as the benefits for organisations that are interested in inner source. For research, we identified a research agenda to further advanced our knowledge in the inner source area.

As part of our future research, we plan to conduct a comprehensive survey of practitioners to identify the key challenges in implementing inner source and propose resolution strategies to overcome. The survey could be also performed to validate the significance of our research agenda. In addition, it is possible that adding non-software and information system related databases may yield similar or different findings. The comparison of findings from different databases and the findings presented in this study can potentially be considered as future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Henry Edison: Methodology, Investigation, Data curation, Writing - original draft. **Noel Carroll:** Investigation, Writing - review & editing. **Lorraine Morgan:** Conceptualization, Investigation, Writing - review & editing. **Kieran Conboy:** Investigation, Supervision, Writing - review & editing.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754489 and with the financial support of the Science Foundation Ireland grant 13/RC/2094.

References

Aksulu, A., Wade, M., 2010. A comprehensive review and synthesis of open source research. J. Assoc. Inf. Syst. 11, 576–656.
Alexy, O., Henkel, J., Wallin, M.W., 2013. From closed to open: job role changes, individual predispositions, and the adoption of commercial open source software development. Res. Policy 42 (8), 1325–1340.
Avison, D., Fitzgerald, G., 1995. Information Systems Development: Methodologies, Techniques and Tools. McGraw-Hill Education.
Bank, E., Grütter, G., Hanmer, R., Stol, K.-J., Sudarsan, P., Williams, C., Yao, T., Yates, N., 2017. InnerSource patterns for collaboration. In: Proceedings of 24thConference on Pattern Languages of Programs.
Berg, V., Birkeland, J., Nguyen-Duc, A., Pappas, I.O., Jaccheri, L., 2018. Software startup engineering: a systematic mapping study. J. Syst. Softw. 144, 255–274.
Bourque, P., Fairley, R.E. (Eds.), 2014. SWEBOK V3.0 Guide to the Software Engineering Body of Knowledge. IEEE.
Brereton, P., Kitchenham, B., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80 (4), 571–583.
Brinkkemper, S., 1996. Method engineering: engineering of information systems development methods and tools. Inf. Softw. Technol. 38 (4), 275–280.
Brown, J.S., Collins, A., Duguid, P., 1989. Situated cognition and the culture of learning. Educ. Res. 18 (1), 32–42.
Capraro, M., Dorner, M., Riehle, D., 2018. The patch-flow method for measuring inner source collaboration. In: Proceedings of 15thInternational Conference on Mining Software Repositories, pp. 515–525.
Capraro, M., Riehle, D., 2017. Inner source definition, benefits, and challenges. ACM Comput. Surv. 49 (4).
Carroll, N., Morgan, L., Conboy, K., 2018. Examining the impact of adopting inner source software practices. In: Proceedings of the 14thInternational Conference on Open Collaboration.
Crowston, K., Wei, K., Howison, J., Wiggins, A., 2012. Free/libre open-source software development: what we know and what we do not know. ACM Comput. Surv. 44 (2), 7:1–7:35.
Dinkelacker, J., Garg, P.K., Miller, R., Nelson, D., 2002. Progressive open source. In: Proceedings of 24th International Conference on Software Engineering, pp. 177–184.
Dybå, T., Dingsoyr, T., 2008. Strength of evidence in systematic reviews in software engineering. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, pp. 178–187.
Dybå, T., yr, T.D., Hanssen, G.K., 2007. Applying systematic reviews to diverse study types: an experience report. In: Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement, pp. 225–234.
Eckert, R., Meyer, S.K., Stuermer, M., 2017. How are open source practices possible within a medical diagnostics company? developing and testing a maturity model of inner source implementation. In: Proceedings of 13th International Symposium on Open Collaboration.
Edison, H., Carroll, N., Conboy, K., Morgan, L., 2018. An investigation into inner source software development: preliminary findings from a systematic literature review. In: Proceedings of 14th International Symposium on Open Collaboration.
Falagas, M.E., Pitsouni, E.I., Maletzis, G.A., Pappas, G., 2008. Comparison of PubMed, Scopus, web of science, and google scholar: strengths and weaknesses. FASEB J. 22 (2), 338–342.
Feller, J., Fitzgerald, B., 2000. A framework analysis of the open source software development paradigm. In: Proceedings of European Conference on Information Systems.
Fitzgerald, B., 2006. The transformation of open source software. MIS Q. 30 (3), 587–598.
Gassmann, O., Enkel, E., 2004. Towards a theory of open innovation: three core process archetypes. In: Proceedings of R&D Management Conference.
Gaughan, G., Fitzgerald, B., Shaikh, M., 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 Software Engineering and Advanced Application, pp. 20–27.
Ghalib, A.K., 2004. Systemic knowledge management: developing a model for managing organisational assets for strategic and sustainable competitive advantage. J. Knowl. Manage. Pract. 5.
Gorschek, T., Garre, P., Larsson, S., Wohlin, C., 2006. A model for technology transfer in practices. IEEE Softw. 23 (6), 88–95.
Grottke, M., Karg, L.M., Beckhaus, A., 2010. Team factors and failure processing efficiency: an exploratory study of closed and open source software development. In: Proceedings of 34th IEEE Annual Computer Software and Applications Conference, pp. 188–197.
Gurbani, V.K., Garvert, A., Herbsleb, J.D., 2006. A case study of a corporate open source development model. In: Proceedings of 28thInternational Conference on Software Engineering, pp. 472–481.
Gurbani, V.K., Garvert, A., Herbsleb, J.D., 2010. Managing a corporate open source software asset. Commun. ACM 53 (2), 155–159.
Hauge, Ø., Ayala, C., Conradi, R., 2010. Adoption of open source in software-intensive organizations - a systematic literature review. Inf. Softw. Technol. 52, 1133–1154.
Hauge, Ø., Sørensen, C.-F., Røsdal, A., 2007. Surveying industrial roles in open source software development. Open Source Development, Adoption and Innovation, 234. Springer, Boston, MA.
Höst, M., Oručević-Alagić, A., 2011. A systematic review of research on open source software in commercial software product development. Inf. Softw. Technol. 53, 616–624.
Huxham, C., Beech, N., 2003. Contrary prescriptions: recognizing good practice tensions in management. Organ. Stud. 24 (1), 69–93.
Ivarsson, M., Gorschek, T., 2011. A method for evaluating rigor and industrial relevance of technology evaluations. Empir. Softw. Eng. 16 (3), 365–395.
Kenny, R., Fallon, E., Fallon, S., Jacob, P., Usher, D., 2017. CALAIS – a component analysis learning algorithm for inner source development. In: Proceedings of 19th International Conference on Computer Modelling and Simulation, pp. 3–10.
Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering - a systematic literature review. Inf. Softw. Technol. 51, 7–15.
Kitchenham, B., Charters, S., 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report. Keele University and Durham University.
Kitchenham, B.A., Dybå, T., Jorgensen, M., 2004. Evidence-based software engineering. In: Proceedings of 26th International Conference on Software Engineering, pp. 273–281.
Klotins, E., Unterkalmsteiner, M., Gorschek, T., 2015. Software engineering knowledge areas in startups companies: a mapping study. In: Lecture Notes in Business Information Processing, 210, pp. 245–257.
Latour, B., 1986. 'The Power of Association' Power, Action and Belief. A New Sociology of Knowledge?. Routledge & Kegan Paul.
van der Linden, F., 2009. Applying open source software principles in product lines. UPGRADE 10 (3), 32–40.
van der Linden, F., Lundell, B., Marttiin, P., 2009. Commodication of industrial software: a case for open source. IEEE Softw. 26 (4).
Lindman, J., Rossi, M., Marttiin, P., 2008. Applying open source development practices inside a company. Open Source Development, Communities and Quality, 275. Springer, Boston, MA.
Lindman, J., Rossi, M., Marttiin, P., 2010. Open source technology changes intra-organizational system development - a tale of two companies. In: Proceedings of International Conference on Information Systems.
Lindvall, M., Rus, I., 2000. Process diversity in software development. IEEE Softw. 17 (4), 14–18.
Linåker, J., Krantz, M., Höst, M., 2014. On infrastructure for facilitation of inner source in small development teams. In: Proceedings of the 15th International Conference on Product-Focused Software Process Improvement, pp. 149–163.
Maier, R., Hädrich, T., Peinl, P., 2005. Enterprise Knowledge Infrastructures. Springer, Berlin.
Martin, G., Lippold, A., 2011. Forge.mil: a case study for utilizing open source methodologies inside of government. In: IFIP Advances in Information and Communication Technology, 365, pp. 334–337.
Martin, K., Hoffman, B., 2007. An open source approach to developing software in a small organization. IEEE Softw. 46–53.
Melian, C., Mähring, M., 2008. Lost and gained in translation: adoption of open source software development at Hewlett-Packard. In: Open Source Development, Communities and Quality, 275. Springer, Boston, MA, pp. 93–104.
Morgan, L., Feller, J., Finnegan, P., 2011. Exploring inner source as a form of intra-organisational open innovation. In: Proceedings of European Conference on Information Systems.
Neus, A., Scherf, P., 2005. Opening minds: cultural change with the introduction of open-source collaboration methods. IBM Syst. J. 44 (2), 215–225.
Oor, P., Krikhaar, R., 2008. Balancing technology, organization, and process in inner source. In: Dagstuhl Seminar Proceedings 08142. Available at http://drops.dagstuhl.de/opus/volltexte/2008/1548.
O'Reilly, T., 2000. Open Source and OpenGL. http://archive.oreilly.com/pub/a/oreilly/ask_tim/2000/opengl_1200.html.
Oručević-Alagić, A., Höst, M., 2016. A two phase case study on implementation of open source development practices within a company setting. In: Proceedings

of International Conference on Software Engineering and Knowledge Engineering, pp. 63–70.
Pulkkinen, M., Mazhelis, O., Marttiin, P., Merliuoto, J., 2007. Support for knowledge and innovations in software development – community within company: inner source environment. In: Proceedings of the 3rd International Conference on Web Information Systems and Technologies, pp. 141–150.
Raghavan, V., Bollmann, P., Jung, G.S., 1989. A critical investigation of recall and precision as measures of retrieval system performance. ACM Trans. Inf. Syst. 7 (3), 205–229.
Riehle, D., Capraro, M., Kips, D., Horn, L., 2016. Inner source in platform-based product engineering. IEEE Trans. Softw. Eng. 42 (12), 1162–1177.
Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B., Odenwald, T., 2009. Open collaboration within corporations using software forges. IEEE Softw. 26 (2), 52–58.
Ripatti, M., Kilamo, T., Salli, K.-T., Mikkonen, T., 2015. Internal marketplace as a mechanism for promoting software reuse. In: Proceedings of 14th Symposium on Programming Language and Software Tools, pp. 119–133.
Robson, C., 2011. Real World Research. John Wiley & Sons.
Saracevic, T., 1995. Evaluation of evaluation in information retrieval. In: Proceedings of 18th International Conference on Research and Development in Information Retrieval, pp. 138–146.
Schreiber, A., Galoppini, R., Meinel, M., Schlauch, T., 2014. An open source software directory for aeronautics and space. In: Proceedings of 10th International Symposium on Open Collaboration.
Scott, W.R., 2001. Institutions and Organizations. Thousand Oaks, CA.
SEI, 2010. CMMI for Development v1.3. Technical Report. Software Engineering Institute, Carnegie Mellon University.
Sharma, S., Sugumaran, V., Rajagopalan, B., 2002. A framework for creating hybrid-open source software communities. Inf. Syst. J. 12 (1), 7–25.
Smith, P., Garber-Brown, C., 2007. Traveling the open road: using open source practices to transform our organization. In: Proceedings of Agile.
Smith, W.K., Lewis, M.W., 2011. Toward a theory of paradox: a dynamic equilibrium model of organizing. Acad. Manage. Rev. 36 (2), 381–403.
Stol, K.-J., Avgeriou, P., Babar, M.A., Lucas, Y., Fitzgerald, B., 2014. Key factors for adopting inner source. ACM Trans. Softw. Eng. Methodol. 23 (2).
Stol, K.-J., Babar, M.A., Avgeriou, P., Fitzgerald, B., 2011. A comparative study of challenges in integrating open source software and inner source software. Inf. Softw. Technol. 53 (12), 1319–1336.
Stol, K.-J., Fitzgerald, B., 2015. Inner source–adopting open source development practices in organizations. IEEE Softw. 32 (4), 60–67.
Theunissen, M., Kourie, D., Boake, A., 2008. Corporate-, agile-, and open source software development: a Witch's Brew or an elixir of life. Balancing Agility Formalism Softw. Eng. Lect. Not. Comput. Sci. 5082 (84–95).
Torkar, R., Minoves, P., Garrigós, J., 2011. Adopting free/libre/open source software practices, techniques and methods for industrial use. J. Assoc. Inf. Syst. 12 (1).
Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., Morgan, L., Oivo, M., Selig, C., Seppänen, P., Sweetman, R., Tyrväinen, P., Ungerer, C., Yagüe, A., 2016. Software Startups: a research agenda. e-Informatica Softw. Eng. J. 10 (1), 89–123.
Vitharana, P., King, J., Chapman, H.S., 2010. Impact of internal open source development on reuse: participatory reuse in action. J. Manage. Inf. Syst. 27 (2), 277–304.
Wesselius, J., 2008. The bazaar inside the cathedral: business model for internal markets. IEEE Softw. 25 (3), 60–66.
Wohlin, C., 2014. Writing for synthesis of evidence in empirical software engineering. In: Proceedings of 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1–4.

Henry Edison is a Marie Curie Fellow with Lero - the Irish Software Research Centre at NUI Galway, Ireland. He received a Ph.D. in Computer Science from Free University of Bozen-Bolzano, Italy. His research interests include empirical software engineering in the area of software product innovation, software startup, Lean startup, inner source, and large-scale agile. He has published in leading journals and conferences in his field.

Noel Carroll is a lecturer in Business Information Systems and a Research Fellow with Lero the Irish Software Research Centre at NUI Galway. He previously worked with the University of Limerick, Dublin City University, and Technische Universiteit Delft. Noel has published, chairs and reviews for leading international journals and conferences in his field. His research examines socio-technical factors in empirical software engineering including topics such as software innovation, agile transformation, and data analytics.

Lorraine Morgan is a Lecturer in Business Information Systems and Senior Researcher with Lero - the Irish Software Engineering Research Centre at NUI Galway. Her principle research interests include open innovation, open source software, inner source and crowdfunding. She has also been involved in a number of collaborative research projects involving international and national based industry partners, attracting over 20m in funding. Her research has also been published in leading journals including the Journal of Strategic Information Systems, Information Systems Journal, European Journal of Information Systems, Database for Advances in Information Systems and Information and Software Technology.

Kieran Conboy is a Professor in Business Information Systems and leads the Lero research group at NUI Galway. He previously worked for Accenture Consulting and the University of New South Wales in Australia. Kieran has published over 100 articles in leading international journals and conferences including Information Systems Research, the European Journal of Information Systems, and the Journal of the AIS. His research examines contemporary technology management and design including concepts such as temporality, flow, open innovation and agility. He is an editor of the European Journal of Information Systems and has chaired many international conferences in his field.