UNIVERSITY OF TARTU

Institute of Computer Science

Computer Science Curriculum

Georg Vann

Literature Review: Open Innovation in Software Engineering

Bachelor's Thesis (9 ECTS)

Supervisor(s): Dietmar Pfahl

Tartu 2019

Literature Review: Open Innovation in Software Engineering

Abstract:

This thesis explores how the principles of open innovation have been applied in the software engineering domain. To methodically gather and analyze studies, the systematic literature review approach is adopted. The relative novelty of open innovation means that there is a lack of models and tools that adopt the principles of open innovation according to different activities of software engineering, such as software testing. Software projects that do employ open innovation are mostly fostered by partnerships between large enterprises and higher education institutions, but small-medium enterprises and government institutions are slowly adopting the open innovation approach as well. All in all, it seems there is need of research that would clearly establish the possibilities of adopting open innovation in software engineering projects.

Keywords:

Systematic literature review, open innovation, software engineering

CERCS: P170 Computer science, numerical analysis, systems, control

Süstemaatiline analüüs: avatud innovatsioon tarkvaratehnikas

Lühikokkuvõte:

Antud töö uurib, kuidas on avatud innovatsiooni põhimõtteid rakendatud tarkvaratehnika valdkonnas. Selleks, et asjakohast teaduskirjandust struktureeritult otsida ja töödelda, lähtutakse süstemaatilise analüüsi meetodist. Kuna avatud innovatsioon on üsna uudne nähtus, siis on praegu puudus mudelitest ja tööriistadest, mis aitaksid avatud innovatsiooni põhimõtteid rakendada sellistes spetsiifilistes tarkvaratehnika protsessides, nagu seda on näiteks tarkvara testimine. Need tarkvaraprojektid, kus avatud innovatsiooni põhimõtteid on rakendatud, on põhiliselt koostööd suurettevõtete ja kõrgkoolide vahel; aga ka väike- ja keskmise suurusega ettevõtted ja riigiasutused kohandavad vähehaaval avatud innovatsiooni põhimõtteid. Üldiselt on täheldada puudust teadustööst, mis annaks selgesõnalise ülevaate avatud innovatsiooni rakendamise võimalustest tarkvaraprojektides.

Võtmesõnad:

Süstemaatiline analüüs, avatud innovatsioon, tarkvaratehnika

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Table of Contents

1 Introduction

Ever since the concept of open innovation was formulated by Henry Chesbrough in 2003, it has gained ever more momentum in both research (Chesbrough, Vanhaverbeke, & West, 2014) and application (Cricelli, Greco, & Grimaldi, 2015). When Munir and others conducted their systematic mapping study in 2016, they found that while open innovation is present in the software engineering domain, the studies up to that point had only covered certain aspects of the phenomenon (Munir, Wnuk, & Runeson, 2016).

The aim of this thesis is to apply the principles of a systematic literature review to answer specific research questions regarding the application of the principles of open innovation in software engineering projects. Thus, relevant research is acquired and analyzed in a way that would allow generalizations and lead to wider conclusions.

Chesbrough first defined open innovation as "a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology" (Chesbrough H. W., 2003, p. xxiv). After the concept had seen some three years of existence, Chesbrough updated the definition as "the use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation, respectively" (Chesbrough H. , 2006, p. 1). The latest iteration of the definition takes into account the possibility of non-pecuniary information circulation as such: "We define open innovation as a distributed innovation process based on purposively managed knowledge flows across organizational boundaries, using pecuniary and non-pecuniary mechanisms in line with the organization's business model" (Chesbrough, Vanhaverbeke, & West, 2014, p. 27). Two features stand out in all these definitions:

- the existence of several parties (i.e. organizations);
- the voluntary exchange of knowledge and ideas (i.e. intellectual capital) in pursuit of innovation.

These are the defining characteristics that are being considered when judging if the principles of open innovation have been applied.

Munir, Wnuk, & Runeson (2016) suggested in their mapping study that the roles of testing and requirements engineering have been largely unstudied in the context of open innovation software engineering – this led to the more general RQ1.

West, Salter, Vanhaverbeke & Chesbrough (2014) reviewed the evolution of open innovation during the decade following Chesbrough's (2003) coining of the term and hinted at the possibility that the profile of those organizations adopting open innovation has broadened, e.g. nonprofits and smaller enterprises. This suggestion led to RQ2.

The fact that open innovation in software engineering is quite a new phenomenon led to RQ3 to see where and by whom has the model sparked more interest and where to perhaps expect future research.

The specific research questions are as follows:

- RQ1: In which activities of software engineering have the principles of open innovation been applied (e.g. software requirements, software testing etc.)?
- RQ2: What type of partnerships are behind the software engineering projects (e.g. SME-government institution, large enterprise-large enterprise etc.)?
- RQ3: Which institutions and scholars have published case studies involving open innovation? Do any institutions or individuals stand out?

The methodology of the thesis is explained in Chapter 2. Chapters 3-6 represent the necessary steps to conduct a systematic literature review: searching for studies, selecting relevant studies, extracting relevant information from given studies and presenting found information, respectively. Chapter 7 summarizes the whole thesis. The detailed list of analyzed studies is included as a separate Excel file called "dataExtraction.xlsx". The summarized list of the studies included with their respective study identifiers is presented in Appendix I.

2 Methodology

This thesis follows the guidelines formulated by Barbara Kitchenham (2007) for conducting a systematic literature review in a way which would take into account the characteristics of software engineering as a distinct domain. In large, a literature review consists of five steps which are now briefly explained, all based on the aforementioned guidelines (Kitchenham, 2007).

2.1 Search Procedure

This first stage is about deciding on the fitting research sources and clarifying the search strategy. A systematic literature review requires definite documenting of conducted work. The reader must be able to make sure where and how the articles have been searched. In addition to this, the reader should be able to reproduce the whole search process.

2.2 Study Selection

After obtaining the initial pool of studies, it is necessary to single out those, which are relevant in the light of the review questions. The selection is carried out by defining explicit sets of selection criteria:

- inclusion criteria ensure that desired themes and properties are present;
- exclusion criteria, conversely, filter out studies with unwanted themes and properties.

The studies that make it through this step are the ones that are actually analyzed.

2.2.1 Study Quality Assessment

The guidelines also emphasize the need to assess the "quality" of the acquired studies as a separate step. The recommendation is to go through extensive quality checklists to check the attributes of each individual study. However, in this case, the author has opted to define a separate exclusion criterion which eliminates all studies coming from sources which may provide subpar studies. Thus, the quality assessment is carried out as a part of the study selection procedure.

2.3 Data Extraction

This is the stage where the information, that is needed to address the review questions, is obtained. This is done with the help of a data extraction form – a single form covering all the relevant attributes regarding the research questions. The form is then compared against each individual study.

2.4 Data Synthesis

The final stage is where the extracted information is summarized and presented by both descriptive and quantitative means. On this summary, generalizations and final conclusions can be drawn.

2.5 Methodological Idiosyncrasies and Limitations

Although these stages represent actual steps taken in that very order, it is worthwhile to note that when conducting the research, the whole process is iterative. When working on later sections, it might have an impact on small details of the previous ones and thus provoke small adjustments.

A very important fact to consider is that in the context of this thesis, only research of academic nature is considered. Needless to say, the majority of software engineering projects do not make it into academic research papers. This approach strongly narrows the potential scope of the research but guarantees that the material collected can by analyzed according to the guidelines and that by referring to the sources, it is possible to uncover the objectives, methodology and origins of the material and the bias that may accompany those factors.

3 Search Procedure

Since software engineering is a relatively new field and open innovation as a scientific term is even newer, it is wise to focus on searching by using the digital means, as opposed to fields that maybe have a longer history. Besides, using digital databases simplifies the processes of defining search criteria and reproducing the search procedure. Furthermore, using the digital means is a much faster approach and allows much more content to be covered.

3.1 Research Sources

Brereton et al. (Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007) have identified eight digital databases as credible sources of scientific and academic research. After getting familiar with the possibilities of the search engines of the different databases, two of them were selected. In addition to those, the SpingerLink database has been used, as it provides access to journals like *Empirical Software Engineering* and *Springer Conference Proceedings* (Kitchenham, 2007). Thus, content is acquired from three sources:

- IEEExplore[1];
- ACM Digital Library[2];
- SpringerLink[3].

It is crucial to point out that access to the databases has been gained through the service provided by the University of Tartu Library[4], which involves certain rights and certain restrictions. Should the reader try and replicate the search procedure with different access rights, the results might differ because of that very reason.

3.2 Search Strategy

All those databases have an *advanced search* option of different kinds, which allow for more specific search configurations – these are crucial to comprehensively shape the search strategy. The search configurations allowed by the individual search engines differ from on another: ACM Digital Library and IEEExplore provide extensive options, whereas SpringerLink's possibilities are somewhat restricted. All three, however, allow the following:

- searching for keywords (including exact phrases) in the body and in the title;

[1] https://ieeexplore.ieee.org/Xplore/home.jsp
[2] https://dl.acm.org
[3] https://link.springer.com
[4] https://utlib.ut.ee/en/databases%20

- exclude the presence of keywords from the body;
- specify the author of the study;
- specify the time frame of the studies on a yearly basis.

The keywords, which are being used, fall into two categories: the first set of keywords assures that the content would be related to software engineering and the second set assures that the content would also involve the principles of open innovation.

According to the SWEBOK guide, software engineering as a domain encompasses 15 different stages and processes ((Eds.) Borque & Fairley, 2014). Considering this, seven keywords (or rather, phrases) have been selected that seem the most relevant considering the possibilities of open innovation: *software engineering*, *software requirements*, *software design*, *software testing*, *software maintenance*, *software quality*, *software development*.

Keywords, that *could* indicate the ideas of open innovation, are very numerous – *collaboration, co-operation,* etc. It is essential to consider, that the keywords being used would reflect the co-operation between organizations or companies, rather than smaller units, such as development teams or even individuals. Bearing this in mind, the following are suitable keywords describing open innovation: *open innovation*, *inter-organizational*, *interorganizational*, *cross-organizational*, *cross-company*, *partnership*.

The term *open innovation* became relevant only after the pioneer of the discipline, Henry W. Chesbrough, presented the concept in 2003 (Chesbrough H. W., 2003). To take this into account, the time frame of the studies is restricted as 2003-2018.

Hereinafter, the specific search criteria for all three databases are defined.

3.2.1 IEEExplore Command Search

In the case of the IEEExplore Command Search[5], the *Metadata Only* search scope is used. The string for the query is as follows:

("open innovation" OR inter-organizational OR interorganizational OR cross-organizational OR cross-company OR partnership) AND ("software engineering" OR "software requirements" OR "software design" OR "software testing" OR "software maintenance" OR "software quality" OR "software development")

After the search query two more filters are applied, as seen in Table 1.

[5] https://ieeexplore.ieee.org/search/advsearch.jsp?expression-builder

Table 1. IEEExplore filters applied after command search.

Filter	Value 1	Value 2
Content Type	Conferences	Journals & Magazines
Year (Range)	2003	2018

This search configuration yielded 258 results as of 27.02.2019.

3.2.2 ACM Digital Library Advanced Search

In the case of ACM Digital Library Advanced Search[6], the first thing to pay attention to is that search is being conducted using the scope provided by *The ACM Guide to Computing Literature* option. In addition to this, three conditions are provided, that are presented in Table 2.

Table 2. ACM Digital Library advanced search configuration.

Field	Criterion	Values
Any field	matches any	"open innovation" "inter-organizational" "interorganizational" "cross-organizational" "cross-company" "partnership"
Abstract	matches any	"software engineering" "software requirements" "software design" "software testing" "software maintenance" "software quality" "software development"
Publication Year	is in the range	2003 <…> 2018

This search configuration yielded 194 results as of 07.03.2019.

[6] https://dl.acm.org/advsearch.cfm?coll=DL&dl=ACM

3.2.3 SpringerLink Advanced Search

Since SpringerLink Advanced Search[7] yields very unexpected results when using Boolean operators, a much simpler strategy is applied, as seen in Table 3.

Table 3. SpringerLink advanced search configuration.

Field	**Value**
with the exact phrase	"open innovation"
where the title contains	software
Show documents published between	2003 <and> 2018
Include Preview-Only content	<unchecked>

This search configuration yielded 72 results as of 24.03.2019.

3.3 Search Results

With all three databases combined, the whole search procedure yielded 472 unique results. The overlap (i.e. number of duplicates) between the results of the three databases is provided in Table 4.

Table 4. Overlap of search results.

	IEEExplore	ACM Digital Library	SpringerLink
IEEExplore		50	0
ACM Digital Library	50		2
SpringerLink	0	2	

The results of ACM Digital Library and IEEExplore had a significant overlap which was expected, considering the similarity of the queries. On the other hand, out of the 72 results

[7] https://link.springer.com/advanced-search

provided by SpringerLink, there were only two duplicates with ACM Digital Library. This is due to two reasons: the different search strategy imposed by SpringerLink's limitations and SpringerLink's unique access to the specific aforementioned journals.

4 Study Selection

Out of the 472 unique results yielded by the search procedure, most are expected to be irrelevant in the context of this review. To identify the studies that do provide insight, study selection criteria are applied to every one of the 472 studies.

4.1 Inclusion Criteria

The inclusion criteria are applied in logical conjunction, which means that all need to apply for the study to be included in the final selection.

The inclusion criteria are as follows:

1. the study is or involves a (multiple) case study;
2. the principles of open innovation have been followed, that is
    a. the existence of several parties (i.e. organizations),
    b. the voluntary exchange of knowledge and ideas (i.e. intellectual capital) in pursuit of innovation;
3. the study involves software engineering activites as defined by the SWEBOK guide ((Eds.) Borque & Fairley, 2014).

The second and third criteria are self-explanatory considering the research questions formulated in Chapter 1. The first criterion is necessary because case studies

- are "aimed at investigating contemporary phenomena in their context" (Runeson & Höst, 2009, p. 134);
- "gather information from on or a few entities (people, groups or organizations)" (Benbasat, Goldstein, & Mead, 1987, p. 370).

These are important factors because they ensure that the study is specific, and the processes are described thoroughly enough so that definitive conclusions can be made regarding the research questions. Since Runeson & Höst (2009) argued that different taxonomies are used to categorize studies, then the term "case study" might be too restricting to have been used as a keyword in the search strategy.

4.2 Exclusion Criteria

The exclusion criteria are applied in logical disjunction, which means that if any single one of the criteria applies, then the study is discarded.

The exclusion criteria are as follows:

1. the study is a secondary study;
2. notice of extraction;
3. not in English;
4. overview of workshop or conference;
5. books;
6. full study not available;
7. the affiliated organizations or their relationship has not been clearly defined.

4.2.1 Quality Assessment Criterion

As mentioned in Section 2.2.1, an additional exclusion criterion is defined to make certain that the studies that make it to the final selection are of considerable quality. This criterion is applied in logical disjunction with the other exclusion criteria.

The quality assessment exclusion criterion is as follows:

- the source of the study has a CORE Rankings Portal[8] category C.

The CORE Rankings portal[8] is a generally accepted ranking system and provides a non-subjective way of quality assessment.

4.3 Study Selection Results

Having applied the selection criteria on the initial pool of 472 studies, 27 studies were found to comply with the criteria and thus make it to the final selection. The 27 studies featured 32 cases of open innovation in software engineering.

[8] http://www.core.edu.au/conference-portal

5 Data Extraction

In this stage, the 27 studies are evaluated one at a time. Each study is reviewed with the help of the data extraction form devised in the next section. The aim of the data extraction form is to make certain that each and every study is assessed on the same terms – this offers a systematic approach where subjectivity on a study-to-study basis is minimized.

An important thing to consider is the possibility of having two or more studies that refer to the same subject case. Should this occur, the subject case itself is only considered once to avoid skewing the quantitative results.

Hereinafter the data extraction form is presented in Table 5. The data items in the form serve the purpose of:

- identifying the study at hand;
- providing data to be later quantified in Chapter 6.

Table 5. Data extraction form with possible values and respective research questions.

Data Item	**Possible values**	**Respective research question**
Study identifier	S1, S2…	-
Case identifier	C1, C2…	-
Name of study	-	-
Year of publication	2003-2018	-
Authors	-	RQ3
Universities involved	-	RQ3
Conference	-	-
Publication	-	-

Activities of software engineering that are performed	Software requirements, software testing, software maintenance, software engineering management, software quality, software development generally	RQ1
Nature of partnership	Any combination of following types: large enterprise, small-medium enterprise, government institution, higher education institution, open source community	RQ2

As mentioned before, the data extraction form is applied to every single study. The detailed results of data extraction are provided as a separate Excel file called "dataExtraction.xlsx". The summarized list of the studies included with their respective study identifiers (S1, S2…) is presented in Appendix I. Hereinafter, the studies are referred to with their study identifiers.

6 Data Synthesis

All the data provided in this chapter is based on the data extraction results which have been analyzed in the Excel environment. Each one of the three following sections answers to one of the three research questions formulated in Chapter 1.

6.1 Open Innovation in Different Activities of Software Engineering

When judging the application of open innovation principles in the different software engineering activities defined by the SWEBOK guide ((Eds.) Borque & Fairley, 2014), it becomes clear that most case studies reviewed do not concentrate on specific activities but rather the overall process of software development. The occurrences are listed in Table 6. Some of the 32 cases reviewed featured several different activities.

Table 6. Occurrences of different software engineering activities.

Software engineering activity	**Occurrences**
Software development generally	26
Software maintenance	1
Software testing	5
Software engineering management	4
Software requirements	1

The reasoning behind such distribution can very well be the novelty of open innovation as a possible method of producing innovation. When Fernandez & Svensson (2017, p. 310) studied the use of open innovation in the requirements engineering process, they found that 87% of the respondents reported to have “received, somehow, ideas from external sources”; however, many participants also claimed to having difficulties understanding the concept of open innovation. There has also been previous discussion on the lack of specific tools and methods for requirements engineering that have been adapted to the context of open innovation (Wnuk, Pfahl, Callele, & Karlsson, 2012). Out of the 10 studies that featured the 11 cases of activity-specific approach, only three (S8, S11, S13) described their innovation processes using a somewhat generalized model or tool that could be applied in a different case – all others were purely case-specific. This supports the conjecture that there is a lack of such models, tools and frameworks in the academic literature that explicitly adapt the principles open innovation.

6.2 Partnerships That Pursue Open Innovation in Software Engineering

There are five different types of parties differentiated in the case projects:

- large enterprise (i.e. LE);
- small or medium-sized enterprise (i.e. SME);
- government institution (i.e. GOV);
- higher education institution (i.e. HEI);
- open-source software (i.e. OSS) community.

The boundary between large enterprises and SMEs is at 250 employees, as defined by the European Commission (Eurostat STRUCTURAL BUSINESS STATISTICS & GLOBAL BUSINESS ACTIVITIES, 2019).

Open-source software has in general been defined as software with public source code, that can be freely modified and used, given that the work of the original author is acknowledged (The Open Source Definition (Annotated): Open Source Initiative, 2019). Open-source software with its contributing communities are a good example of a certain kind of open innovation practice (West & Gallagher, 2006).

The occurrences of relationships between different types of parties are listed in Table 7.

Table 7. Number occurrences of relationships between different types of parties.

	LE	SME	GOV	HEI	OSS
LE	17	6	1	27	10
SME	6	1	0	7	6
GOV	1	0	0	3	1
HEI	27	7	3	7	3

OSS	10	6	1	3	0

Figure 1 better visualizes the relationships between different types of parties – a thicker chord between two different types of parties represent a bigger number of such relationships.

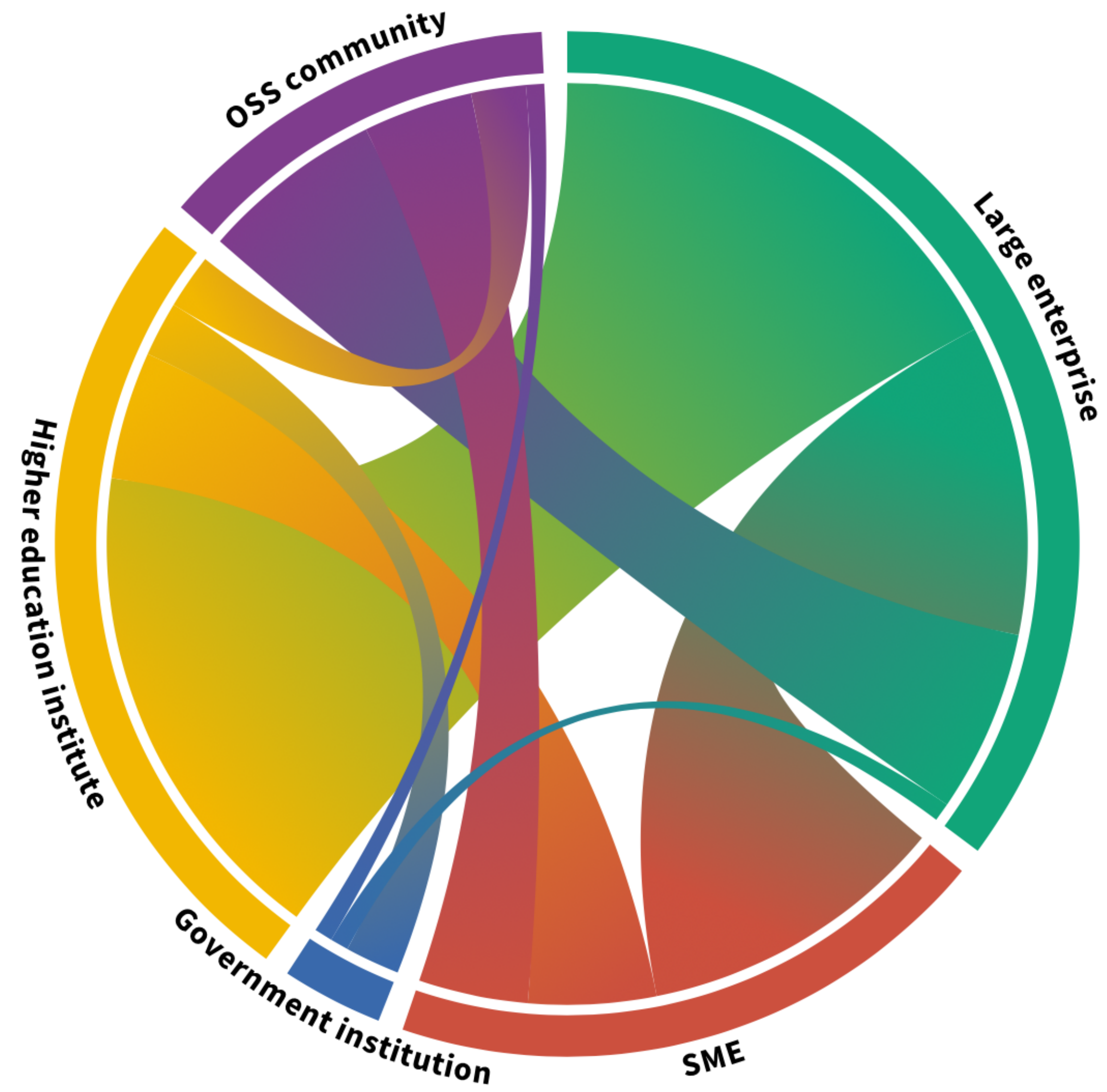

Figure 1. Chord diagram of the relationships between different types of parties.

Collaboration between large enterprises and HEIs are dominating the scene with a significant amount SMEs and open-source communities involved as well. A few government institutions are present as well.

The large number of LEs practicing open innovation comes as no surprise, since their involvement has been studied and confirmed since the emergence of the open innovation paradigm (van de Vrande, de Jon, Vanhaverbeke, & de Rochemont, 2009). The large number of HEIs involved can be considered somewhat of a surprise. While certain previous trends have suggested an increase in collaboration between academia and industry (Perkmann & Walsh, 2007), the broad quantitative study of Cricelli, Greco, & Grimaldi (2015) pointed out the relative scarcity of having universities and research institutions as open innovation partners when studying the adoption of open innovation by European firms.

The apperance of a notable amount of SMEs confirms previous research, which has stated their increasingly extensive involvement in open innovation activities (van de Vrande, de Jon, Vanhaverbeke, & de Rochemont, 2009). A relatively large proportion of SME partnerships is with OSS communities. This can be contributed to the fact that OSS principles have been covered more widely in the software engieering literature - as opposed to the more general paradigm of open innovation - and the notion that SMEs relatively have the most to gain from OSS practices (Munir, Wnuk, & Runeson, 2016).

The small number of government institutions developing their frameworks (S4, S20, S22/S23) can be contributed to the novelty of the Government 2.0 movement (Chun, Shulman, Almazan, & Hovy, 2010). There rest of the studies featuring government involvement are of a specific military and/or tactical nature (S15, S25, S26). However, these studies featuring government institutions do confirm the notion of West, Salter, Vanhaverbeke, & Chesbrough (2014) that there has been an increase in nonpecuniary motivations – in this case developing software for government use – for adopting open innovation.

6.3 Origins of Selected Studies

The analysis of the origins of the final pool of the studies suggests no inclinations toward any universities or scholars that prevail in the case studies. All institutions and scholars occur only once in the selected pool. However, three scholars do stand out by having studied the phenomenon of open innovation in software engineering on a individual case level (S13), as well as, with a wider scope (Munir, Wnuk, & Runeson, 2016; Wnuk, Pfahl, Callele, & Karlsson, 2012) and – H. Munir and P. Runeson once and K. Wnuk twice. Wnuk, especially, seems to concentrate his efforts on the possibilites of applying open-source software solutions to achieve innovation. Such inexistent levels of consolidation of research, can, once again, be attributed to the novelty of the phenomen of open innovation in software engineering – practitioners of software engineering are slowly becoming more familiar with such opportunities. Higher volumes of case studies will probably become available in the coming years.

7 Summary

The goal of this thesis was to investigate the application of open innovation principles in the software engineering domain. Three distinct aspects were explored: the different activities of software engineering that could benefit from open innovation (RQ1); the different types of partnerships that foster open innovation in software engineering projects (RQ2); the origin of relevant studies to find out about current trends in this research field (RQ3).

To find relevant studies, the systematic literature review approach was adopted. This method enables a systematic search and selection of studies followed by a systematic retrieval of data from the selected studies.

Regarding the different activities of software engineering, it became apparent that most case studies focused on software development generally, rather than more specific activities, such as software requirements or software testing. Some cases of software maintenance, software testing, software engineering management and software requirements were present, however. This relative absence of specialization respective to the different activities was attributed to the lack of theoretical models and tools that would support such specific approach.

The partnerships that were most prevalent were between large enterprises and higher education institutions, followed by different combinations where small-medium enterprises, open-source software communities and even a few government institutions were present as well. As principles of open innovation become more widely known, the number of small-medium enterprises, government institutions and perhaps even civic nonprofits is expected to rise.

No universities and only a few scholars stood out when examining the origins of selected studies. H. Munir, P. Runeson and especially K. Wnuk were singled out as those with a more serious degree of research regarding this matter.

There is one overarching feature that characterizes all three aspects and open innovation in software engineering in general, and that is one of novelty. While the paradigm of open innovation itself is gaining ground, its possibilities in software engineering specifically are still relatively little-explored and thus cases of software engineering that employ open innovation are few and far between.

8 References

(Eds.) Borque, P., & Fairley, R. (2014). Guide to the Software Engineering Body of Knowledge, Version 3.0. xxxii. IEEE Computer Society.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly, 11*, 369-386.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 571-583.

Chesbrough, H. (2006). Open Innovation: A New Paradigm for Understanding Industrial. *Open Innovation: Researching a New Paradigm*, 1.

Chesbrough, H. W. (2003). *Open Innovation: The New Imperative for Creating and Profiting from Technology.* Boston: Harvard Business School Press.

Chesbrough, H., Vanhaverbeke, W., & West, J. (2014). *New Frontiers in Open Innovation.* Oxford: Oxford University Press.

Chun, S., Shulman, S. W., Almazan, R. S., & Hovy, E. (2010). Government 2.0: Making Connections Between Citizens, Data and Government. *Information Polity*, 1-9.

Cricelli, L., Greco, M., & Grimaldi, M. (2015, October). Assessing the open innovation trends by means of the eurostat community innovation survey. *International Journal of Innovation Management.*

Eurostat STRUCTURAL BUSINESS STATISTICS & GLOBAL BUSINESS ACTIVITIES. (2019, April 30). Retrieved from European Commission Web Site: https://ec.europa.eu/eurostat/web/structural-business-statistics/structural-business-statistics/sme

Fernandez, S., & Svensson, R. B. (2017). A Survey of Practitioners Use of Open Innovation. *Euromicro Conference on Software Engineering and Advanced Applications*, (pp. 305-312).

Kitchenham, B. A. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Keele: Keele University.

Munir, H., Wnuk, K., & Runeson, P. (2016, April). Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, pp. 684-723.

Perkmann, M., & Walsh, K. (2007). University–industry relationships and open innovation: Towards a research agenda. *International Journal of Management Reviews, 9*(4), 259-280.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 131-164.

The Open Source Definition (Annotated): Open Source Initiative. (2019, May 2). Retrieved from Open Source Initiative web site: https://opensource.org/docs/definition.php

van de Vrande, V., de Jon, J. P., Vanhaverbeke, W., & de Rochemont, M. (2009). Open innovation in SMEs: Trends, motives and management challenges. *Technovation*, 423-437.

West, J., & Gallagher, S. (2006). Challenges of open innovation: the paradox of firm investment in open-source software. *R&D Management*, 319-331.

West, J., Salter, A., Vanhaverbeke, W., & Chesbrough, H. (April 2014. a.). Open innovation: The next decade. *Research Policy*, 805-811.

Wnuk, K., Pfahl, D., Callele, D., & Karlsson, E.-A. (2012). How can Open Source Software Development Help Requirements Management Gain the Potential of Open Innovation: An Exploratory Study. *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, (pp. 271-279). Lund.

Appendices

I. List of Selected Studies

Study identifier	Study
SpringerLink studies	
S1	Richomme, M., Suddul, G., Basgeet, R., & Soobul, A. (2010). Emerginov: How Free Software Can Boost Local Innovation, a Win-Win Deal between Operator and Local Innovation Partners. *AFRICOMM 2010: E-Infrastuctures and E-Services for Developing Countries* (pp. 132-140). Berlin: Springer.
S2	Eklund, U., & Bosch, J. (2012). Introducing Software Ecosystems for Mass-Produced Embedded Systems. *ICSOB 2012: Software Business* (pp. 248-254). Berlin: Springer.
S3	Kautz, K., Bunker, D., Rab, S. M., & Sinnet, M. (2011). Investigating Open Innovation and Interorganizational Networks in the IT Industry: The Case of Standard Software Customization. *Researching the Future in Information Systems* (pp. 231-246). Berlin: Springer.
S4	Choi, Y., Lee, Y.-G., & Ra, J. (2012). A Case of Standard Develop Framework Based on Open-Source Software in Korea Public Sector. *Computer Applications for Graphics, Grid Computing, and Industrial Environment* (pp. 210-214). Berlin: Springer.
S5	Henttonen, K. (2011). Libre Software as an Innovation Enabler in India Experiences of a Bangalorian Software SME. *Open Source Systems: Grounding Research* (pp. 220-232). Berlin: Springer.
S6	Mahajan, A., & Clarysse, B. (2013). Technological Innovation and Resource Bricolage in Firms: The Role of Open Source Software. *Open Source Software: Quality Verification* (pp. 1-17). Berlin: Springer.
S7	Mian, S. Q., Teixeira, J., & Koskivaara, E. (2011). Open-Source Software Implications in the Competitive Mobile Platforms Market. *Building the e-World Ecosystem* (pp. 110-128). Berlin: Springer.
S8	van der Linden, F. (2013). Open Source Practices in Software Product Line Engineering. *Software Engineering* (pp. 216-235). Berlin: Springer.
S9	Lindman, J., Rajala, R., & Rossi, M. (2010). FLOSS-Induced Changes in the Software Business: Insights from the Pioneers. *Software Business* (pp. 199-204). Berlin: Springer.
ACM Digital Library studies	
S10	Faria, K. A., de Andrade Freitas, E. N., & Vincenzi, A. M. (2017). Collaborative Economy for Testing Cost Reduction on Android Ecosystem. *Proceedings of the 8th ACM SIGSOFT International Workshop on Automated Software Testing* (pp. 11-18). New York: ACM.

S11	Enoiu, E. P., & Čaušević, A. (2014). Enablers and Impediments for Collaborative Research in Software Testing: An Empirical Exploration. *Proceedings of the 2014 international workshop on Long-term industrial collaboration on software engineering* (pp. 49-54). New York: ACM.
S12	Carroll, E. R. (2005). Estimating Software Based on Use Case Points. *OOPSLA '05 Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 257-265). New York: ACM.
S13	Munir, H., Linåker, J., Wnuk, K., Runeson, P., & Regnell, B. (2018). Open innovation using open source tools: a case study at Sony Mobile. *Empirical Software Engineering, 23*(1), 186-223.
S14	He, Y., Foley, T., Hofstee, T., Long, H., & Fatahalian, K. (2017, July). Shader Components: Modular and High Performance Shader Development. *ACM Transactions on Graphics (TOG), 36*(4).
IEEExplore studies	
S15	Higley, H. H., & Harder, R. J. (2003). Tailoring an Electronic Meeting System to Military Planning and Operations, a Case Study. *Proceedings of the 36th Hawaii International Conference on System Sciences.* IEEE.
S16	Dadeau, F., Castillos, K. C., Ledru, Y., Triki, T., Vega, G., Botella, J., & Taha, S. (2013). Test Generation and Evaluation from High-Level Properties for Common Criteria Evaluations - The TASCCC Testing Tool. *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation* (pp. 431-438). IEEE.
S17	Carrozza, G., Faella, M., Fucci, F., Pietrantuono, & Russo, S. (2012). Integrating MDT in an Industrial Process in the Air Traffic Control Domain. *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops* (pp. 225-230). IEEE.
S18	van der Valk, R., Pelliccione, P., Lago, P., Heldal, R., Knauss, E., & Juul, J. (2018). Transparency and Contracts: Continuous Integration and Delivery in the Automotive Ecosystem. *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP* (pp. 23-32). IEEE.
S19	Girard, A., Spry, S., & Hedrick, J. (2005, March). Intelligent cruise control applications: real-time embedded hybrid control software. *IEEE Robotics & Automation Magazine, 12*(1), pp. 22-28.
S20	Monteiro, R. S., Pinel, R. E., Zimbrão, G., & de Souza, J. M. (n.d.). The MDArte experience: OrgAnizational aspects acquired from a successful partnership between government and academia using model-driven development. *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)* (pp. 575-586). IEEE.
S21	Abdul, A., Bass, J. M., Ghavimi, H., & Adam, P. (2017). Product Innovation with Scrum: A Longitudinal Case Study. *2017 International Conference on Information Society (i-Society)* (pp. 22-27). IEEE.

S22	Kang, G.-I., Kwon, Y.-I., & Kim, E.-J. (2013). An Analysis of e-Government Standard Framework (eGovFrame) and Its Effects. *2013 15th International Conference on Advanced Communications Technology (ICACT)* (pp. 860-868). IEEE.
S23	Kang, G.-I., Mun, S.-J., & Kwon, Y.-I. (2012). Development of e-Government Standard Framework through Open Innovation Strategy. *2012 14th International Conference on Advanced Communication Technology (ICACT)* (pp. 1117-1122). IEEE.
S24	Aoyama, M., Yabuta, K., Kamimura, T., Inomata, S., Chiba, T., Niwa, T., & Sakata, K. (2013). PROMIS: A Management Platform for Software Supply Networks Based on the Linked Data and OSLC. *2013 IEEE 37th Annual Computer Software and Applications Conference* (pp. 214-219). IEEE.
S25	Santiago, C., Rusu, A., Falconi, L., Petzinger, B., & Crowell, A. (2009). Overview and flight-by-flight analysis of trajectory prediction systems using a 2D Galaxy Visualization. *2009 IEEE/AIAA 28th Digital Avionics Systems Conference* (pp. 3.B.3-1 - 3.B.3-8). IEEE.
S26	Alwardt, A. L. (2012). Leveraging the Cloud to create a Network Centric Support Environment for Support Equipment. *2012 IEEE AUTOTESTCON Proceedings* (pp. 347-351). IEEE.
S27	Zyda, M., Thukral, D., Jakatdar, S., Engelsma, J., Ferrans, J., Hans, M., . . . Vasudevan, V. (2007). Educating the Next Generation of Mobile Game Developers. *IEEE Computer Graphics and Applications, 27*(2), 96-95.

II. License