

A Literature Review of User-Led Open Source Consortia

MASTER THESIS

Markus Büttner

Submitted on 7 January 2022



Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik
Professur für Open-Source-Software

Supervisors:

Elçin Yenişen Yavuz, M.Sc.
Prof. Dr. Dirk Riehle, M.B.A.



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 7 January 2022

License

This work is licensed under the Creative Commons Attribution 4.0 International license (CC BY 4.0), see <https://creativecommons.org/licenses/by/4.0/>

Erlangen, 7 January 2022

Abstract

The development of increasingly complex software systems is difficult to handle, especially for smaller entities. In addition to the use of commercial software, open source development models are therefore becoming increasingly popular. The formation of "User-Led Open Source Consortia" is one of these models. Here, several institutions join forces by pooling their resources to develop software for their own use. The resulting software is made available to the public under open source licenses.

The amount of publications on the subject is small, but has been growing steadily over the last years, with the phenomenon being named and defined in slightly different ways.

This thesis presents a qualitative literature review that examines this body of research. In the process, the various characteristics of this phenomenon will be highlighted.

Contents

1	Introduction	1
2	Related Work	3
3	Research Design	7
3.1	Research Questions	7
3.2	Literature Selection	8
3.2.1	Literature Database	8
3.2.2	Literature Search	8
3.2.3	Literature Filtering and Preparation	11
3.3	Qualitative Data Analysis	11
3.3.1	Tools	12
3.3.2	Coding Process	12
4	Results and Discussion	15
4.1	Resulting Literature	15
4.2	Properties of User-led Open Source Consortia	17
4.2.1	Community	17
4.2.2	Product	18
4.2.3	Organization	19
4.2.4	Evolution	21
4.2.5	Goals	22
4.3	Classification of User-led Open Source Consortia	24
4.3.1	Competition between Members	24
5	Conclusion & Limitations	25
6	Future Research	27
7	Acknowledgements	29
	Appendices	31

A	Pre-identified Publications	33
B	Snowball Search Results	34
C	Keyword Search Results	35
D	Code System	36
E	Literature Overview	37
References		39

List of Figures

3.1	Snowball literature search process as described by Wohlin (2014).	9
3.2	Coding process as suggested by Thomas (2006).	12
3.3	Early results of the first coding step.	13
3.4	Overview of topics added before the second coding step.	13
4.1	Publications related to ULCs over the years.	16
4.2	Number of publications per researched organization.	16
4.3	Kuali organizational chart (Liu, Wang et al., 2012)	20
4.4	Comparison of different methods of software acquisition (Wheeler & Hilton, 2012)	23

List of Tables

2.1	Part of the comparison between community source, regular open source and commercial software by Liu, Wang et al. (2012).	4
-----	--	---

Acronyms

CD-ERP Collaboratively-Developed Enterprise Resource Planning

COSS consortium-based open source software

CS community source

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

HE higher education

OS operating system

OSS open source software

PDF Portable Document Format

QDA qualitative data analysis

RQ research question

SOA service-oriented architecture

SSH Secure Shell

ULC User-led Open Source Consortium

1 Introduction

In today's world, almost any electronic device we use contains some kind of processor. Many of those devices even surpass high-end computing equipment of the last century in regard to computing power and features. The emergence of embedded devices and phenomena like the Internet of Things further amplifies this. Ranging from inconspicuous devices like pocket calculators or kitchen appliances to self-driving cars or smartphones and tablets, all of those devices are computers. As a consequence, there is a rising need for software to make use of those devices. Be it the operating system (OS), server-side software or actual end-user applications, it all has to be implemented and therefore facilitates some kind of software development process. The use of open source development models is not uncommon for the development of small projects such as programs and libraries by hobby programmers. In the field of commercial software, closed source development models have long been commonplace.

With open source development tools and libraries already being an important part of the closed source development process, the rise of popular open source software (OSS) like the Linux Kernel or the Android Open Source Project – just to name a few – changed the attitude towards OSS in commercial environments even further. Companies such as Oracle or IBM recording significant revenues from service offerings in the context of self-developed and third-party OSS (IBM, 2021; Oracle Corporation, 2021) prove the long-held prejudice that commercial endeavors with OSS are not possible to be unjustified.

The growing adaption of OSS has inevitably led to the birth of new development models. In particular, the development of large software systems to be used by thousands of users requires a development model that ensures the ability to quickly adapt to changing requirements and the longevity of the system as a whole. Development and ongoing maintenance of such systems were the domain of commercial software vendors for a long time. Largely due to the vast amount of funding, knowledge and workforce required to implement them. The resulting high cost and possible vendor lock-in¹ – among other drawbacks – encouraged the inception of a new open source development model that will be examined in

¹Customers dependence on products and services of a single provider that prevents switching to another provider because of the high technical and/or financial efforts involved

depth by this thesis.

For the remainder of the thesis we call this model *User-led Open Source Consortium (ULC)*. ULC represents a hybrid model of software development. It combines elements of closed source software development and OSS development. The result is a community that develops software for its own use by pooling resources. If it is not outsourced to paid external developers, the implementation of the software is done by the community members' own in-house developers. Through the use of appropriate licenses, the end result is available to external parties as OSS.

There is a small body of research surrounding ULCs² that has been slowly but steadily growing over the last years. It describes different software development models that ultimately resemble ULCs with varying characteristics. There have been publications that deal with various aspects of specific projects and organizations, but an effort to derive a general description or provide means to distinguish such endeavors is still missing. This thesis fills that gap by conducting a systematic literature review on said body of research.

The main contribution of this thesis is a literature review that provides an overview of existing literature on ULCs and further uses that to derive properties of such organizations.

The rest of the thesis is structured as described hereafter: First, chapter 2 provides an outline of related publications, organizations, and other phenomena. Chapter 3 describes the methodology used, with section 3.1 first explaining the research questions (RQs). Sections 3.2 and 3.3 discuss the literature acquisition process and the subsequent analysis and evaluation of the literature found. Chapter 4 begins by providing an overview of the literature evaluated and ends with presenting and discussing the research findings. The final conclusion is found in chapter 5. Chapter 6 addresses future work.

²Even though it deviates from the correct plural form, ULCs will be used to refer to multiple User-led Open Source Consortia in order to improve reading flow and reduce potential confusion caused by identical acronyms

2 Related Work

The conducted literature review mainly involves working with literature in the context of ULC. Since the lack of a similar publication is the main reason for the existence of this thesis, this section will highlight literature that provides additional insight into the examined and related topics. Due to the small size of the body of research, some publications mentioned in the following sections will inevitably be part of the literature review.

The term ULCs describes a software development model that has yet to receive a widely accepted name throughout the software development and research community. Apart from the small size of this area of research, the lack of such an official description may be one reason why the search for publications yields only a few results. Nonetheless, there are several publications naming and describing the phenomena:

If there is a designation that is nearing official status through widespread use, it probably is "*community source*" (CS). As one of the oldest terms for the phenomenon, it was most likely coined by Brad Wheeler, one of the central figures in the Kuali Foundation¹ (Hanganu, 2008). At this point, it should be mentioned that Gartner, a market research company, also claims to have coined the term (Taft, 2009).

Community source is also used by Liu et al. throughout most of the research groups publications (Liu, Hansen et al., 2012, 2014, 2020; Liu et al., 2013; Liu & Tu, 2011; Liu et al., 2007; Liu, Wang et al., 2012; Liu, Wheeler et al., 2008; Liu et al., 2010; Liu, Zeng et al., 2008; Liu & Zhao, 2007, 2008). The research group around Liu provides the major part of publications on the topic and therefore has the most detailed description of their version of ULCs.

Other terms that essentially describe CS are "*directed open source*" (Mackie, 2008) and "*collaborative community-source*" (Wheeler & Hilton, 2012).

With "*Collaboratively-Developed Enterprise Resource Planning*" (CD-ERP), Almigheerbi et al. (2020a) describe a variant of *community source* aimed particu-

¹<https://kuali.org/>

2. Related Work

	Community source	Ordinary open source	Commercial software
Definition	The application development requires initial investments which are shared by the partners; the result is open source.	The software is free for adopters; the developers are volunteers.	The users need to buy software; dedicated investment is required to develop the software.
Builders	Organized teams of formal employees from multiple partner institutions	Loosely connected volunteer developers	System development departments in software companies
Users	Development partners, deployment members, and others	Organizations or individuals who use the software for free	Organizations or individuals who pay for the software
Customization	The system is designed to meet the requirements of development partners	The system is designed to meet some generic requirements	Standard package

Table 2.1: Part of the comparison between community source, regular open source and commercial software by Liu, Wang et al. (2012).

larly at use in Libyan universities, where the Libyan Ministry of Higher Education in particular is to be part of the consortium. In the course of their research, a literature review on this model has additionally been conducted (Almigheerbi et al., 2020b).

Germonprez et al. (2013) describe a concept called "*community of competitors*". The concept is similar to CS, but emphasizes that members cooperate in the context of the consortium, while otherwise continuing to compete in the same market.

Produsage is another term encountered in the context of user-led creation of digital products. Although this concept may also be expanded to encompass the creation of software, it mainly describes the creation of digital media for the Web 2.0 (Bruns, 2007a, 2007b).

Hanganu (2008) mentions an alternative interpretation of CS that is quite the opposite of the models presented above. The source code is licensed to a closed community and changes to the source code may also only be distributed within this community.

Of all the terms discovered during the thesis, *community source* is the only one which Wikipedia has a definition for (Wikipedia, 2021). The definition is based on the description published by Wheeler (2007) and coincides with that of Liu's research group.

Table 2.1 gives an overview of the definition of *CS* by Liu's research group. It is described as a model combining the advantages of regular commercial software development and open source software development: A shared investment of resources from participating partners results in the production of OSS, whereas the development is done by the partners employees. Since the partners are also the users of the system, it is tailored to meet their specific needs (Liu, Wang

et al., 2012).

In their most recent publication, Liu’s research group uses the term "*consortium-based open source software*" (COSS). The underlying meaning is still the same as that of *CS*, but COSS intends to emphasize the aspect of coordination between independent entities and also counteracts the ambiguity of the term by underlining that it is a consortium rather than an unspecified type of community (Liu et al., 2021).

With the term "*Open Source User Foundations*", Riehle (2016) provides a name that, even without knowing the definition, suggests the existence of elementary differences to commonly developed OSS:

An open source user consortium is a consortium of companies who sponsor, steer, and possibly also develop open source software for their own use rather than as part of software products they sell (Riehle, 2016)

Additionally taking into account the refining to User-led Open Source Consortia (Riehle, 2021), this will be the definition we follow in this thesis as closely as possible.

2. Related Work

3 Research Design

This chapter covers the planning and execution of the systematic literature review that represents the core contribution of this thesis. The general approach is adapted from the ‘Procedures for performing systematic reviews’ (Kitchenham, 2004). The overall review process as intended by Kitchenham is divided into the following three main phases and their respective sub-phases:

1. Planning the review
 - (a) Identifying need for a review
 - (b) Developing the review protocol
2. Conducting the Review
 - (a) Identifying the body of research
 - (b) Selecting studies
 - (c) Assessing study quality
 - (d) Extracting Data
 - (e) Synthesizing Data
3. Reporting the Review

The review protocol resulting from phase one is used to lay down the actual RQs, the literature search and selection process and also the data extraction and synthesis. Its contents are outlined in this section. The subsections contain appropriate explanations in case additional guidelines were used or deviations from or additions to Kitchenham’s guidelines were deemed necessary.

3.1 Research Questions

As already mentioned the main goal of the thesis is to elaborate the properties of ULCs as presented by the existing body of research on the topic. To achieve this goal, the conducted research focuses on the following RQs.

RQ1: What are the defining properties of ULCs?

Open source communities are often composed of a diverse set of people and other actors who work together to create, improve, and distribute software for their own and others' use. In order to describe ULCs and compare them with other forms of open source software development, we need to identify the defining properties of such projects.

RQ2: How can different ULCs be classified based on their properties?

Even if various ULCs have similar philosophies and approaches at their core, it must also be assumed that the exact implementation of this type of software development differs from project to project or consortium to consortium. It is therefore necessary to create means that allow comparison and classification.

3.2 Literature Selection

Regarding the literature selection process, it should be noted that the initial scope of the literature review was limited to a predetermined selection of publications. A list of publications directly or indirectly relevant to the topic of ULCs was provided at the beginning of the thesis. After a thorough study of this list and after excluding publications that do not match the filter criteria established in section 3.2.3, only 12 publications on the topic of ULCs remain. Appendix A contains a list of these publications. The following sections describe the methods used to find and filter more literature.

3.2.1 Literature Database

For all activities involving search in online databases, Google Scholar¹ was used. The search results were limited by setting additional constraints like publication types, year of publication or the general field of research, but depending on the used keywords, we still received thousands of unrelated results. As the effort required to check every returned result would pose an unmanageable problem for time-limited work like this thesis, we decided to counter this by limiting each search to the first 40 returned results.

3.2.2 Literature Search

As already mentioned we do have a small selection of literature that is already deemed relevant to our topic and contains some of the most popular publications

¹<https://scholar.google.com/>

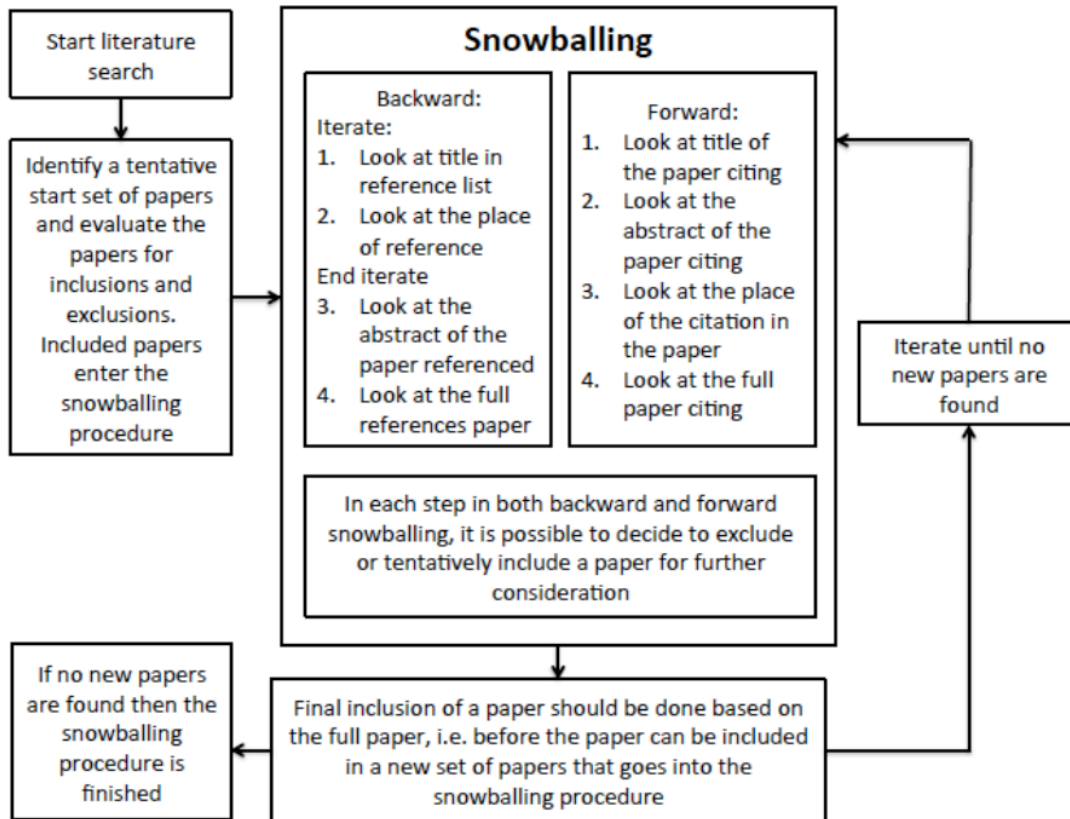


Figure 3.1: Snowball literature search process as described by Wohlin (2014).

on ULC. Looking at said selection, two facts immediately stand out without requiring much analysis efforts:

- The number of different authors is comparatively small
- Most publications are related by referencing each other

Kitchenham (2004) advises to employ a keyword search to find relevant literature. Although ULCs are a niche topic, the used search engines still yield a lot of results during the keyword search. The number of relevant results is extremely low compared to the total number of results. Because of that and also in the light of the two facts listed above, we employ the snowball search approach described by Wohlin (2014) as our main search strategy. This allows us to make use of the tightly-knit net of publications on ULCs while also being able to account for any related topics that may only be explored by specific keyword searches.

Figure 3.1 shows an overview of the snowball search process. We already have had a set of publications on ULCs that was compiled from the original existing literature according to our filtering criteria. We use this as the starter set for the snowball search. At the core of the snowball search there are two search

steps: backward and forward search. We executed these steps iteratively for each publication that is included in the currently observed set of publications.

During the *backward search* the references of the publication are checked for relevance to our topic. This involves looking at the title of the reference and which context it is referenced in. Should a reference present itself as possibly relevant, its abstract and full text need to be evaluated in order to add it to the set of observed publications.

The *forward search* allows for determining more recent publications by checking where the currently observed publication itself is being cited. Google Scholar provides a "Cited By"-functionality which is used during this stage of search. The limit of only looking at the first 40 results that is introduced earlier is not applied in this case. On the one hand, the number of citations is usually below this limit, and on the other hand, it is assumed at this point that a citation indicates a connection to our topic that cannot be ignored. Apart from that the forward search follows the same principles as the backward search: After looking at the title and abstract of the citing publication, a final decision on its inclusion in the observed set of publications is made based on its full text.

Utilizing each publication found during any of both stages, the process is repeated until no additional publications are found. Appendix B lists all publications found during each iteration of the snowball search.

To complement the snowball search we still employed a round of keyword search as originally suggested. This allows us to gather publications that do not have any connections to the already found publications on ULCs. In addition to keywords derived from the already identified literature, the keyword search makes use of an initial set containing the following keywords and -phrases:

- user-controlled, user-created, user-developed, user-led, user-managed
- collaborative open source, community source, software development consortium, software development foundation, software produsage

Different combinations are used to create search phrases that yield results as closely related to ULCs as possible. An overview of keyword combinations yielding publications which were later used in the literature review is shown in Appendix C. Publications already discovered during the snowball search were ignored. Starting with evaluating title and abstract and eventually the publications' full text, a decision on inclusion in the review is made based on the filter criteria presented in the following section.

3.2.3 Literature Filtering and Preparation

A decision on inclusion in the review and further preparations has to be made for each individual publication prior to conducting the actual review.

The decision on inclusion is based on a set of general filter-criteria established prior to the thesis:

- Area of research
 - Computer Science
 - Management Information Systems
- Language
 - English
 - German

Further, publications have to cover concepts, projects or communities related to ULCs. Publications that only address the technical or architectural aspects of the products created by a ULC and not the actual ULC behind it must not be considered. Considering these constraints, the expected body of research is already very small. In order to be able to include interesting gray literature, no further restrictions are made regarding the types of publications or the study design. Only dissertations and theses are excluded.

In conjunction with the filtering, a set of complementary preparations has to be carried out. The final decision on inclusion and the literature review itself rely on the full text of the publication. Since the literature review is carried out using software tools, a digital copy has to be acquired. If no version is publicly available at one of the literature databases, we try to obtain it using the FAU's licenses by connecting to the university network via SSH and trying to access the publication that way. In case this also fails, we directly ask one of the authors for a copy. The last step of the preparations is to collect basic information about each publication. This information includes the year and type of publication, the authors, and the ULCs covered.

3.3 Qualitative Data Analysis

The data extraction and synthesis during the literature review is done by conducting a qualitative data analysis (QDA). Since there is no predetermined framework for analyzing data on ULCs, the inductive approach presented by Thomas (2006) is used. In the following sections the involved tools and the process itself is laid out.

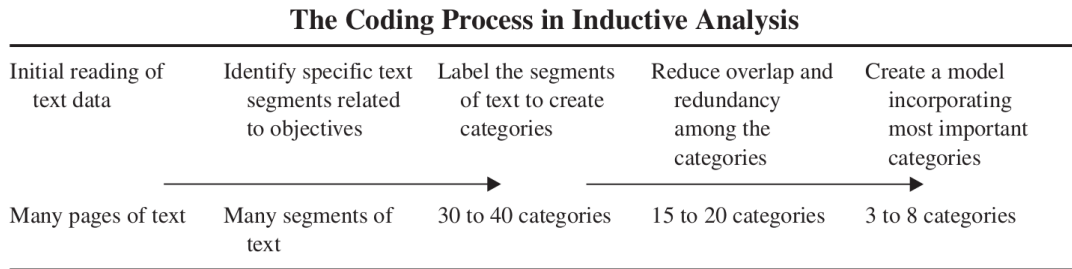


Figure 3.2: Coding process as suggested by Thomas (2006).

3.3.1 Tools

Multiple tools are used to collect, organize and process all publications that are part of the QDA conducted in this thesis. The most important tools are listed and briefly explained in this section.

The collection and organization of publications is aided by the open source software *Zotero*² and its accompanying browser add-on. Apart from organizing publications of any type in so-called "Collections" it features the automatic import of publications and their respective metadata from many websites and literature databases. The possibility of tagging publications with custom keywords and creating links between publications is used to document the search process. The statistical overviews presented in section 4.1 are based on exports of the literature collection.

The literature analysis began with the chairs own online QDA-tool *QDACity*³ and later migrated to the de facto standard tool for QDA *MAXQDA*⁴. It supports various types of qualitative and mixed methods data analysis including Inductive Coding as employed in our QDA.

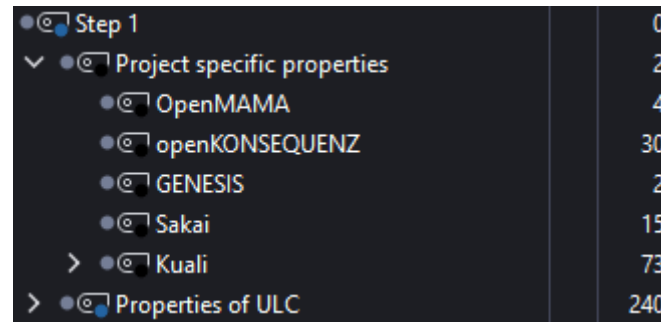
3.3.2 Coding Process

The process of *Inductive Coding* suggested by Thomas (2006) is an approach for extracting information relevant to the particular research objectives from an existing body of research. An overview of the process can be seen in Figure 3.2. The initial reading of the text data was already done while filtering relevant publications during the literature search. Starting from there, each step further condenses the relevant information into categories, until only a few remain. The resulting set of categories is called *code system*. The categories are derived from the RQs presented in section 3.1.

²<https://www.zotero.org/>

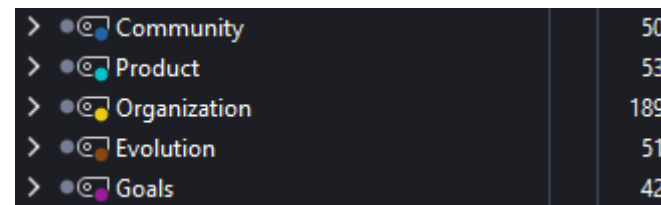
³<https://qdacity.com/>

⁴<https://www.maxqda.com/>



● Step 1	0
▼ ● Project specific properties	2
● OpenMAMA	4
● openKONSEQUENZ	30
● GENESIS	2
● Sakai	15
> ● Kual	73
> ● Properties of ULC	240

Figure 3.3: Early results of the first coding step.



> ● Community	50
> ● Product	53
> ● Organization	189
> ● Evolution	51
> ● Goals	42

Figure 3.4: Overview of topics added before the second coding step.

The first actual step of coding is the identification of relevant text segments. Since all of our RQs involve finding properties of ULCs, said segments are collected under the category "*Properties of ULC*". Many of the identified segments cover general properties of ULC, but some also describe a specific ULC. For a better overview during the coding process those segments are labeled with a category specific to the respective ULC. The resulting code system can be seen in Figure 3.3.

The second step consists of creating categories and labeling the text segments found in step one accordingly. This step produced a lot more categories than the upper limit of 40 that was suggested by Thomas. Since many of the categories are only loosely related, we assigned them to a parent-category which we call "topic". Inside those topics the suggested number of categories could mostly be upheld. This also improves the general overview of the process and makes the following steps more manageable. The topics as shown in Figure 3.4 are used:

Community – Who participates in the ULC, how are they related to each other and what are the circumstances while joining or establishing the ULC.

Product – What are the products developed by the ULC, what technologies are used and which other properties do they have?

Organization – How is the ULC organized and how similar is it compared to OSS development? How do the involved processes look like?

Evolution – What changes are happening in and around the consortium? How

3. Research Design

does the consortium react to those changes?

Goals – Which goals do the members pursue when joining or establishing a ULC? Which goal does the ULC itself pursue?

In the third step, more generalized categories are created to represent existing similar categories. All categories that are duplicates of each other are merged under one of the duplicates. In the later case the duplicate assigned to the most text segments is kept, while all other duplicates are deleted.

During the fourth and last step a model is created from the most important categories. To achieve this additional sub-categories are created. These categories resemble sub-topics that are used to link conflicting properties and highlight important insights. After this step the QDA is finished and ready to be reported. The final code system can be seen in Appendix D.

4 Results and Discussion

The results of the literature review are reported in this chapter. Starting with an overview of the collected literature we then proceed to examining the discoveries made during the QDA.

4.1 Resulting Literature

Following the methods for literature acquisition described in section 3.2, we found a total of 27 publications related to ULCs including 12 of the pre-identified publications. Of those publications the majority are either conference papers (8) or journal papers (15). 4 were online resources, including websites (2), a blog entry (1) and a briefing note (1). A table containing an overview of all publications is available in Appendix E. Figure 4.1 visualizes the number of publications published from 2004 to 2021. Over the entire period observed, there is not a single year without not at least one publication. Apart from the years 2007 and 2008, which can be traced back to the beginning of increased publications on the Kualu Foundation, there is still a significant number of publications in the context of ULCs over the last ten years. This shows that it is still an active part of the ongoing open source research.

Although there is an active community of researchers and users involved with ULCs, the total number of researched organizations is small. As shown in Figure 4.2, only five different ULCs are covered in the reviewed literature and nearly two thirds of it discusses the Kualu Foundation alone (11) or in conjunction with the Sakai Project (7). Together with the publications specifically discussing the Sakai Project (1), more than two thirds of the publications discuss a ULC from higher education (HE). A small fraction of the publications covers openKONSEQUENZ (2), GENESIS (1) and OpenMAMA (1), while the remaining publications (5) do not focus on a specific ULC.

In conjunction with the small number of different project, there is also a very limited selection of authors and research groups. The most prominent fact is the share of publications made by Liu’s research group. Considering the co-authors

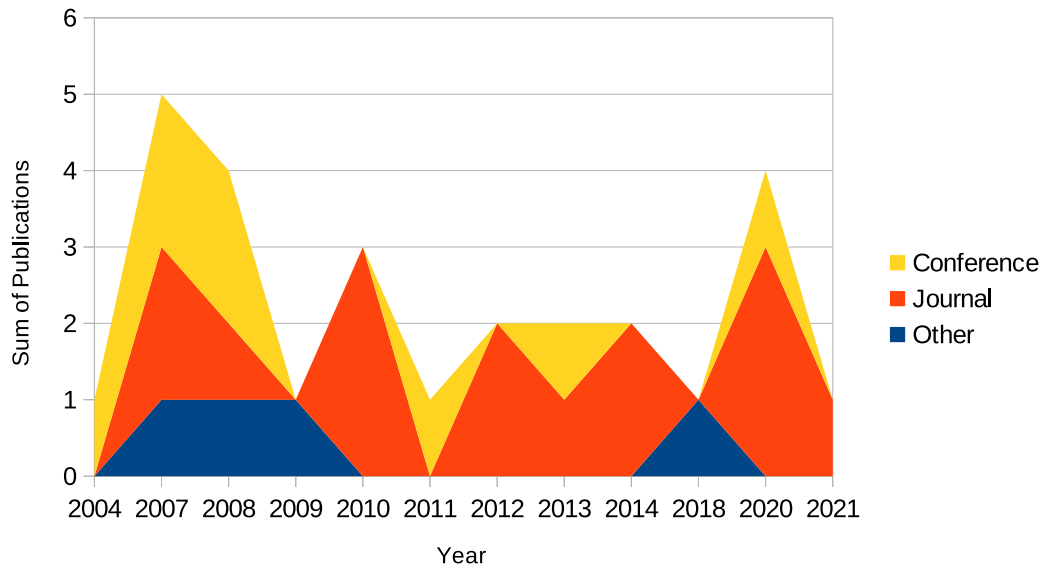


Figure 4.1: Publications related to ULCs over the years.

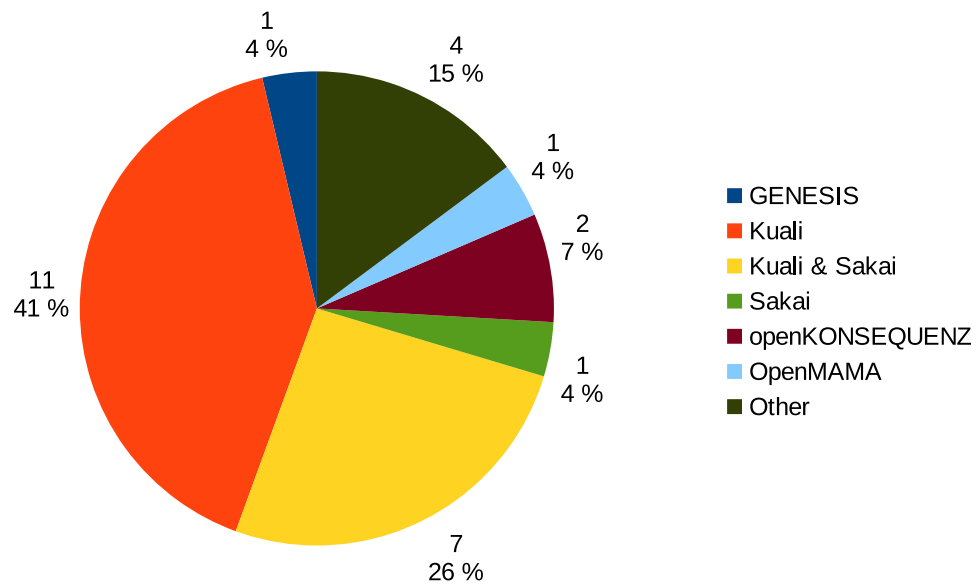


Figure 4.2: Number of publications per researched organization.

on those publications there is some more diversity, but the limited number of completely different

4.2 Properties of User-led Open Source Consortia

The definition of ULCs in chapter 2 gives a hint on the properties one could expect a ULC to have. This chapter highlights those and additional properties attributed to ULCs by the different authors in their respective publications. The properties are arranged in five topics with each of them covering an aspect of ULCs.

4.2.1 Community

The topic "Community" represents the members and their respective roles along with the date of joining in relation to the formation of the ULC and what may have contributed to joining at that time. Four categories were drawn from the literature examined: *Members*, *Roles*, *Adoption time* and *Circumstances*.

Institutions with backgrounds in HE are particularly often involved in ULCs (Liu et al., 2014; Wheeler, 2007). The striking frequency with which institutions with similar influence and status come together in this regard is referred to by Liu et al. (2013) as "actor similarity" and is taken as a sign of potential for good collaboration. The potentially associated "culture fit", i.e. the congruence of attitude and norms of the members, could also contribute positively here (Liu, Wheeler et al., 2008).

The participation of industrial and commercial companies, or the mixing of several backgrounds, is also possible, as Boldyreff et al. (2004) and Schwab et al. (2020) demonstrate on the basis of GENESIS and openKONSEQUENZ. This may even go to the extent of actively competing outside the ULC. In this case the ULC serves as a "*neutral home*" for the members (Germonprez et al., 2013).

The members can take different roles within the ULC. Liu, Zeng et al. (2008) show this using the case of Kuali, which distinguishes between "*development partners*" and "*deployment partners*", in other words, partners involved in the development and partners who primarily want to use the finished software. Even the latter can still benefit the respective ULC, since successful missions serve as advertising to attract new members (Panettieri, 2007). Success in general is a way to promote and validate the ULC model (Severance, 2007).

The point in the "life" of a ULC at which a member joins is crucial to the course of the collaboration. The opportunity to participate in the forming of the ULC at an early stage and to represent one's own interests is particularly advantageous for early adopters, although it is precisely this early phase that

demands a considerable amount of work due to the build-up process (Liu et al., 2013; Wheeler, 2007). If a member is particularly capable of learning, they can benefit especially as a late adopter, since the time the other members spent on the learning process can be made up quickly (Liu et al., 2013).

In addition to the goals pursued by joining a ULC, other *circumstances* are also crucial. Trust in the success of the ULC and the other members is one of them. If this is not present, it is difficult to justify the associated risks and costs (Liu et al., 2013).

The size and capabilities of a member also have an influence. For small members it is interesting as soon as it enables the development of otherwise too costly projects (Liu, Zeng et al., 2008; Wang et al., 2010).

For larger members, this factor is less decisive, since the costs have proportionally less influence and joining is bound to become profitable more quickly. This effect is amplified for members with existing development skills, as easier deployment and faster adaptation of the software to their own needs is expected (Liu, Zeng et al., 2008; Liu & Zhao, 2007).

4.2.2 Product

The "Product" topic provides an overview of what the developed software comprises and how it is structured and implemented. The three categories *Common base*, *New technologies* and *(Technology) flexibility* were considered.

Common Base: Underlying is a common base that is reusable and is designed to meet the general requirements of the ULC. Liu, Wang et al. (2012) see link this to increase sustainability. Germonprez et al. (2013) refer to this part of the software as "the basics", which in the further course ensure that there is more time for adding differentiating features.

Those differentiating features are not to be influenced by the common base in a way that profits the members in an unequal way. This is of special importance in the case of members competing with each other outside the ULC (Germonprez et al., 2013; Heinritz et al., 2014).

To make the common base usable for all members, it must be extensible. Successful ULCs such as the ones involved with Kuali or Sakai place particular emphasis on the modularized architecture of their software. This means that new features can be easily added to existing systems in the form of modules. These can easily be used by other members, or also be integrated in the common base, provided there is a general benefit. Liu, Wheeler et al. (2008) refer to this architectural concept as 'develop once and use anywhere on any platform'.

Software is seldom used on its own and must therefore support integration with other software. The reviewed literature presents two distinct cases: The gradual integration of parts of existing software in the course of a migration (Heinritz et al., 2014) and the permanent cooperation with tools used in parallel (Collins,

2010).

New Technologies: In the first case mentioned above, the process is supported by the use of new technologies – specifically service-oriented architecture (SOA) (Collins, 2010). In the literature, other new technologies such as web services and workflow automation are also recognized as useful in software customization (Liu & Tu, 2011; Liu, Wang et al., 2012). SOA remains the most influential, being the one which significantly supports the properties of the common base and its extension (Heinritz et al., 2014; Liu et al., 2007).

(Technology) Flexibility: On the basis of Kuali, Liu, Wang et al. (2012) show that flexibility, or even more specifically technology flexibility¹, is essential for satisfying the specific requirements of individual members. Likewise, frequent adaptation to volatile requirements within the ULC and the effort involved in deploying software to new members' systems is greatly reduced. (Liu & Tu, 2011; Liu, Wang et al., 2012). Liu, Wheeler et al. (2008) even go as far as to propose that technology flexibility is more important for ULC than in-house development, because this, way not only one member creates customized software for himself, but the customization of the software becomes easier for everyone.

4.2.3 Organization

In this section we present the findings from the literature that describe the organization and processes of ULCs. Remarks on similarities with the development of OSS also get reviewed. For this purpose, the topic "Organization" was examined in relation to five aspects: *Collaboration*, *Rules & Commitments*, *Resource Pooling*, *Service & Support*, and *Similarities to OSS*.

Collaboration: The *management* of a ULC is a process handled by the member community. Liu et al. (2007) refer to structures formed in this context as 'virtual organizations'. This does not involve a fixed, physical place for cooperation, but rather a geographical and temporal distribution of the members and their collaborative work (Liu et al., 2010; Panettieri, 2007). As exemplified by the organizational chart of the Kuali board in fig. 4.3, decisions involve a hierarchy of functional and technical boards consisting of employees of the members (Almigheerbi et al., 2020a; Collins, 2010; Liu et al., 2007).

Influence: Although members in a ULC limit their individual influence in favor of shared goals, additional influence can be exerted both as a result of meritocratic structures in the ULC or based on the size of a member's resource investment (Wheeler & Hilton, 2012).

According to Wheeler (2007), enlightened self-interest and associated *mutual ben-*

¹'The ability to adapt to both incremental and revolutionary change in the business or business process with minimal penalty to current time, effort, cost, or performance.' (Nelson et al., 1997)

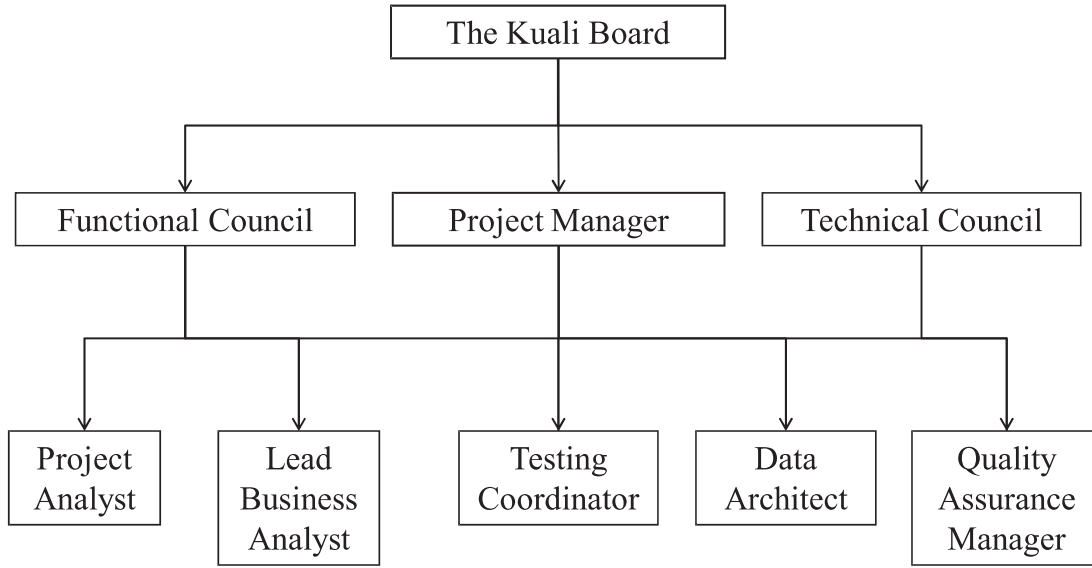


Figure 4.3: Kuali organizational chart (Liu, Wang et al., 2012)

effit holds the ULC together by ensuring that all members benefit equally from the collaboration.

Outsourcing is another aspect of collaboration in and outside a ULC. Liu et al. (2010) and Liu and Zhao (2008) show in the case of Kuali that outsourcing is already a central part of the development process. If the development work is externalized, the number of necessary in-house developers decreases and the design of the software can be put in the foreground (Liu & Zhao, 2008). In particular, employed OSS practices allow external partners to build up extensive expertise in dealing with the software and, if necessary, to use this expertise for other services (Heinritz et al., 2014).

The *legal representation* is also part of the ULC. This requires all members to surrender their rights to the developed software, but at the same time makes it easier for the ULC to manage and represent its intellectual property holistically (Heinritz et al., 2014; Wheeler, 2007).

Rules & Commitments: ULC are subject to fixed rules and commitments that go beyond what is usual for OSS (Liu et al., 2007; Panettieri, 2007). This is necessary considering ULCs are not about individual developers loosely working together, but about employees of companies or institutions working in formally organized teams to achieve common goals (Liu et al., 2020). These roles and responsibilities are documented accordingly (Wheeler & Hilton, 2012). In addition, there are the shared values of the ULC, which contain specifications such as the licenses used or technical aspects like modularity or reusability. They are another factor that is meant to hold the ULC together, since the members would not be able to uphold these specifications on their own (Liu et al., 2020, 2021).

Resource Pooling: In general, financial resources as well as personnel and infrastructure are provided by the members or external supporters Liu et al. (2007) and Riehle (2018). The *finances* can come both from the members themselves and from external sources. Sufficient financial resources are required in particular for the establishment of a ULC, which makes additional external funding at the start an important factor Liu et al. (2013).

Joining a ULC requires a significant investment. For the Kuali project, this amounts from half a million to one million US dollars for a development partner. This ensures that the member collaborates closely to contribute to the success of the project. In the case of Kuali, 75% of this sum is paid in personnel costs Liu et al. (2013), Liu et al. (2007).

The infrastructure used by the members may also be pooled. Reducing effort and energy spent on maintaining and running many individual systems, should be seen as a contribution to the protection of our environment (Almigheerbi, 2020).

Service & Support: One effect of the OSS model is that the support is completely unbundled from the software (Wheeler, 2007). It can therefore be provided by any entity. Members of the ULC can make use of the network of the community as for know-how transfer (Liu et al., 2013) and thus make the process more efficient (Wheeler, 2007). Commercial support is also possible and offers smaller service providers the opportunity to exploit a niche in the market for themselves through specialization (Schwab et al., 2020).

Similarities to OSS: The efficient support is based on the ULCs similarities to OSS. It may adapt parts of corporate software development, but with core features like open communication (Wheeler, 2007) and development (Heinritz et al., 2014) or the public availability of all artifacts

It may adapt parts of enterprise software development, but with core features like open communication (Wheeler, 2007) and development (Heinritz et al., 2014) or public availability of all artifacts, it still offers many features of OSS. One of the most important decisions in this context is the selection of a license that allows further use and reuse as OSS (Taft, 2009), but is also compatible with the particular circumstances of ULC (Heinritz et al., 2014; Liu et al., 2007).

4.2.4 Evolution

This section covers the literature segments dealing with changes happening in and around the ULC and whether it can take an active or passive role in managing the change. The two categories *Members* and *Model* were examined.

Members: Being the "moving part" in each organization, members resemble a constant source of change. Joining the ULC with varying or conflicting requirements (Liu et al., 2007) is why technology flexibility is such an important aspect

(Liu, Wheeler et al., 2008).

Although rising member counts are a positive thing, they also complicate the ULC's management. Refining the management structures as Liu et al. (2021) suggests is an active measure the ULC can take to mitigate this.

Member turnover is another issue related to members joining and leaving the ULC. The resulting changes to the development teams may complicate the management (Liu et al., 2010), but the influx of new members also possibly increases the ULC's diversity and can bring technical insights (Liu et al., 2021).

Model: The ULC model may also be part of the evolutionary process. Hanganu (2008) views the governance model as something that can be changed in later stages. Riehle (2018) writes about a similar change, but instead of changing the development model, they suggest to look at (umbrella) foundations and check whether associating with one would be of mutual benefit.

4.2.5 Goals

This section creates an overview of prevalent member's individual and common organizational goals presented in the reviewed literature. The categories *Cost Reduction*, *Resource Sharing*, *Software Self-Usage* were isolated in reference to the definition of ULC. The choice fell on important categories, which had aspects that received little visibility in the previous Topics due to the reductions during the coding process.

Cost Reduction: As an individual goal this is one of the main reasons for the existence of ULCs. It is shown as directly related to the reduction of financial risk achieved by collaboration and resource pooling on the organizational level (Germonprez et al., 2013; Liu et al., 2007; Wheeler, 2007).

Resource Sharing: Similar to cost reduction this goal is present throughout most of the literature and also strongly related to collaboration and resource pooling (Liu et al., 2007; Wheeler, 2007).

Software Self-Usage: The collaborative production of software for the own use is part of the definition of ULC (Riehle, 2018). In addition to helping avoid vendor lock-in, it is also a factor contributing to cost reduction by providing an alternative to the "buy or build" decision (Liu et al., 2007; Panettieri, 2007; Schwab et al., 2020). Figure 4.4 shows the models position at the opposite of the spectrum.

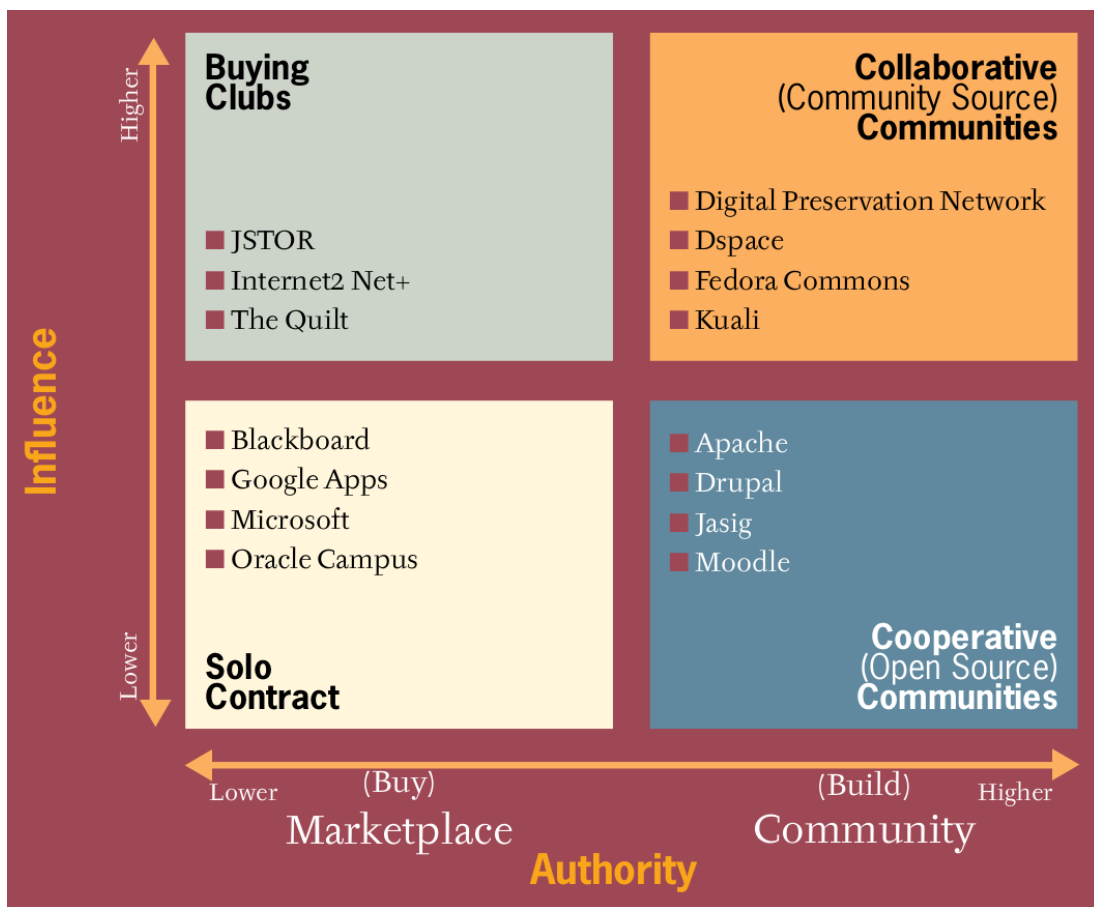


Figure 4.4: Comparison of different methods of software acquisition (Wheeler & Hilton, 2012)

4.3 Classification of User-led Open Source Consortia

The initial intention for answering RQ2 was to create a classification framework based on the properties identified to answer RQ1. Based on the small number of different consortia covered in the reviewed literature, the validity of a universal classification framework built on this basis is questionable. Considering this, we do not attempt to create such a classification framework. Nevertheless, in order to at least provide a partial answer to RQ2, we hereafter present the identified property that is most likely to be usable to classify different ULCs.

4.3.1 Competition between Members

The relationships between members outside the ULC are a distinctive Property. Looking at the “Community of Competitors” presented by Germonprez et al. (2013) and the actors involved in openKONSEQUENZ (Schwab et al., 2020), three different classes can be derived:

- No competition
- Competition possible
- Competing actors

5 Conclusion & Limitations

While some features and concepts are limited to individual authors and projects, a look at the results reveals that there is a common notion of ULC. Some details differ, but at least the basic ideas as formulated in the definition of ULC we used are recognizable in each project and author. However, what could already be seen when conducting the literature search is that while recent publications exist in this topic area, the concept as such has not undergone any serious changes or innovations for quite some time.

This mentioned static of the subject complex strongly points to one of the limitations this work is subject to. Already with the decision to base the literature search mainly on the snowball principle and also not to broaden the initial literature as recommended by the guidelines, the possible range and diversity of the later examined literature was strongly limited. This effect was exacerbated by the method of analysis, which, although designed to be as objective as possible, was nonetheless – especially in the case of inexperienced authors – not completely objectively tenable.

5. Conclusion & Limitations

6 Future Research

Expanding on the issues mentioned in the previous chapter, repeating the process applied in this thesis on an expanded set of data could be a first step. Especially additional organizations outside the higher education sector should be considered. Looking at younger and active projects like iPlant and openMAMA could be help in finding more recent research and insights on the topic.

7 Acknowledgements

At first, I want to thank Prof. Dr. Riehle and the Open Source Research Group for their ongoing dedication to open source. Without them, I would not have had the chance to gain insight into this subject area from an academic perspective.

Secondly, I would like to thank Elçin Yenişen Yavuz for the mentoring, advice, and most importantly patience during the work on my master thesis.

7. Acknowledgements

Appendices

A Pre-identified Publications

- ‘Communication and conflict issues in collaborative software research projects’ - Boldyreff et al., 2004
- ‘Open source communities of competitors’ - Germonprez et al., 2013
- ‘Konsortiale Open-Source-Softwareentwicklung im Energiesektor’ - Heinritz et al., 2014
- ‘Achieving flexibility via service-centric community source’ - Liu et al., 2007
- ‘A Cooperative Analysis Framework for Investment Decisions in Community Source Partnerships’ - Liu, Zeng et al., 2008
- ‘Outsourcing of Community Source’ - Liu et al., 2010
- ‘Technology flexibility as enabler of robust application development in community source’ - Liu, Wang et al., 2012
- ‘Antecedents of Community Source Network Formation’ - Liu et al., 2013
- ‘The community source approach to software development and the Quali experience’ - Liu et al., 2014
- ‘How to Capture Open Source User Consortia 4/4 - Software Research and the Industry’ - Riehle, 2018
- ‘The Ecosystem of openKONSEQUENZ, A User-Led Open Source Foundation’ - Schwab et al., 2020
- ‘Open Source 2010’ - Wheeler, 2007

B Snowball Search Results

- Iteration 1
 - ‘The community source development model’ - Hanganu, 2008
 - ‘Real Options Analysis of the Community Source Approach’ - Liu and Zhao, 2007
 - ‘On Assessment of Project Success in Community Source Development’ - Liu, Wheeler et al., 2008
 - ‘COMMUNITY-BASED OPEN SOURCE - The Phenomenon and Research Opportunities’ - Liu and Tu, 2011
 - ‘Keeping the family together’ - Liu et al., 2020
 - ‘Community Source Software in Higher Education’ - Wang et al., 2010
 - ‘The Marketecture of Community’ - Wheeler and Hilton, 2012
- Iteration 2
 - ‘A collaboratively-developed enterprise resource planning (CD-ERP) Approach in Libyan higher education’ - Almigheerbi et al., 2020a
 - ‘An Empirical Analysis of Critical Success Factors for CD-ERP Model.’ - Almigheerbi et al., 2020b
 - ‘Practices of the circular economy in Community Source projects’ - Almigheerbi, 2020
 - ‘On Outsourcing and Offshoring in Community Source’ - Liu and Zhao, 2008

C Keyword Search Results

The following list shows the keyword combinations that yielded publications that were used in the literature review. This list does not include all tried keyword combinations. Combinations that did not yield new results or combinations covered by the snowball search are not listed.

- collaborative software development consortium
 - ‘Sustaining collaborative software development through strategic consortium’ - Liu et al., 2021
- kuali community source
 - ‘Partnering for innovation’ - Collins, 2010
 - ‘Pushing the Boundaries of Innovation through Community Source’ - Severance, 2007

D Code System

The code system consists of six topics selected to represent the different aspects of ULC. Each topic (1st column) is associated with several properties or categories (2nd column). Categories in turn consist of several properties (3rd column).

Topic	Category/Property	Property
Community	Members	Background Actor similarity
	Roles	—
	Adoption time	Early adopters Late adopters
	Circumstances	Trust Small partners Large partners Capabilities
Product	Common base	Common requirements Non-differentiating Extensibility Integration
	New technologies (Technology) flexibility	— —
Organization	Collaboration	Management Influence Mutual Benefit Outsourcing
	Rules & Commitments Resource Pooling	— Finances Personnel Infrastructure
	Service & Support	Unbundling Support through Mem- bers Commercial Support
	Similarities to OSS	—
Evolution	Members	—
	Model	—
Goals	Cost reduction	
	Resource sharing	—
	Software Self-Usage	—

E Literature Overview

Reference	Year	Oranization(s)	Publication Type
Almigheerbi, 2020	2020		Journal Article
Almigheerbi et al., 2020a	2020		Journal Article
Boldyreff et al., 2004	2004	GENESIS	Conference Paper
Collins, 2010	2010	Kuali	Journal Article
Germonprez et al., 2013	2013	OpenMAMA	Journal Article
Hanganu, 2008	2008	Kuali, Sakai	Briefing Note (Online)
Heinritz et al., 2014	2014	openKONSEQUENZ	Journal Article
Liu et al., 2020	2020	Kuali, Sakai	Journal Article
Liu et al., 2021	2021	Kuali	Journal Article
Liu et al., 2014	2014	Kuali	Journal Article
Liu et al., 2013	2013	Kuali	Conference Paper
Liu and Tu, 2011	2011	Kuali	Conference Paper
Liu et al., 2007	2007	Kuali	Conference Paper
Liu, Wang et al., 2012	2012	Kuali, Sakai	Journal Article
Liu, Wheeler et al., 2008	2008	Kuali	Conference Paper
Liu et al., 2010	2010	Kuali	Journal Article
Liu, Zeng et al., 2008	2008	Kuali	Journal Article
Liu and Zhao, 2008	2008	Kuali	Conference Paper
Liu and Zhao, 2007	2007	Kuali	Conference Paper
Panettieri, 2007	2007		Website (Online)
Riehle, 2018	2018		Blog Post (Online)
Schwab et al., 2020	2020	openKONSEQUENZ	Conference Paper
Severance, 2007	2007	Kuali, Sakai	Journal Article
Taft, 2009	2009		Website (Online)
Wang et al., 2010	2010	Sakai	Journal Article
Wheeler and Hilton, 2012	2007	Kuali, Sakai	Journal Article
Wheeler, 2007	2007	Kuali, Sakai	Journal Article

References

- Almigheerbi, T. S. (2020). Practices of the circular economy in community source projects: A preliminary study [Publisher: Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu]. *Business Informatics. Informatyka Ekonomiczna*, (4), 9–12. Retrieved October 7, 2021, from <http://cejsh.icm.edu.pl/cejsh/element/bwmeta1.element.desklight-466ce43a-bd83-48c4-bfdb-c410a46b53a2>
- Almigheerbi, T. S., Ramsey, D. & Lamek, A. (2020a). A collaboratively-developed enterprise resource planning (CD-ERP) approach in libyan higher education. *International Journal of Information and Education Technology*, 10(4), 284–298.
- Almigheerbi, T. S., Ramsey, D. & Lamek, A. (2020b). An empirical analysis of critical success factors for CD-ERP model. *J. Comput.*, 15(2), 37–47.
- Boldyreff, C., Nutter, D. & Rank, S. (2004). Communication and conflict issues in collaborative software research projects. *"Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering" W8S Workshop - 26th International Conference on Software Engineering, 2004*, 14–17. <https://doi.org/10.1049/ic:20040258>
- Bruns, A. (2007a). The future is user-led: The path towards widespread produsage. *Proceedings of perthDAC 2007: The 7th International Digital Arts and Culture Conference*, 68–77.
- Bruns, A. (2007b). Produsage. *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*, 99–106.
- Collins, M. (2010). Partnering for innovation: Interviews with OCLC and kuali OLE [Publisher: Taylor & Francis]. *Serials Review*, 36(2), 93–101.
- Germonprez, M., Allen, J. P., Warner, B., Hill, J. & McClements, G. (2013). Open source communities of competitors. *Interactions*, 20(6), 54–59. <https://doi.org/10.1145/2527191>
- Hanganu, G. (2008). *The community source development model* [OSSWATCH]. Retrieved October 4, 2021, from <http://oss-watch.ac.uk/resources/communitysource>
- Heinritz, C., Herdt, P., Janeck, S., Regenbogen, G., Riehle, D., Rose, F., Roth, M., Thoma, D. & Tuchs, M. (2014). Konsortiale open-source-softwareentwicklung

- im energiesektor. *Zeitschrift für Energiewirtschaft*, 83–86. <http://dirkriehle.com/wp-content/uploads/2014/06/konsortiale.pdf>
- IBM. (2021). *IBM reports 2021 third-quarter results* [IBM newsroom]. Retrieved January 4, 2022, from <https://newsroom.ibm.com/2021-10-20-IBM-Reports-2021-Third-Quarter-Results>
- Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004), 1–26.
- Liu, M., Hansen, S. & Tu, Q. (2012). Organizational control in community source projects. *SIGBPS Workshop on Business Processes and Services (BPS'12)*, 65.
- Liu, M., Hansen, S. & Tu, Q. (2014). The community source approach to software development and the kualu experience. *Communications of the ACM*, 57(5), 88–96. <https://doi.org/10.1145/2593687>
- Liu, M., Hansen, S. & Tu, Q. (2020). Keeping the family together: Sustainability and modularity in community source development. *Information and Organization*, 30(1). <https://doi.org/10.1016/j.infoandorg.2019.100274>
- Liu, M., Hansen, S. & Tu, Q. (2021). Sustaining collaborative software development through strategic consortium. *The Journal of Strategic Information Systems*, 30(3), 101671. <https://doi.org/10.1016/j.jsis.2021.101671>
- Liu, M., Hull, C. E. & Hung, Y.-T. C. (2013). Antecedents of community source network formation: The case of kualu [ISSN: 1530-1605]. *2013 46th Hawaii International Conference on System Sciences*, 4267–4276. <https://doi.org/10.1109/HICSS.2013.95>
- Liu, M. & Tu, Q. (2011). COMMUNITY-BASED OPEN SOURCE - the phenomenon and research opportunities: *Proceedings of the 13th International Conference on Enterprise Information Systems*, 419–424. <https://doi.org/10.5220/0003491104190424>
- Liu, M., Wang, H. J. & Zhao, J. L. (2007). Achieving flexibility via service-centric community source: The case of kualu. *Association for Information Systems - 13th Americas Conference on Information Systems, AM-CIS 2007: Reaching New Heights*, 1522–1532. Retrieved September 16, 2021, from [https://scholars.cityu.edu.hk/en/publications/achieving-flexibility-via-servicecentric-community-source\(d6c33cb5-dade-44a2-a759-9641c0f913bb\).html](https://scholars.cityu.edu.hk/en/publications/achieving-flexibility-via-servicecentric-community-source(d6c33cb5-dade-44a2-a759-9641c0f913bb).html)
- Liu, M., Wang, H. J. & Zhao, J. L. (2012). Technology flexibility as enabler of robust application development in community source: The case of kualu and sakai. *Journal of Systems and Software*, 85(12), 2921–2928. <https://doi.org/10.1016/j.jss.2012.06.026>
- Liu, M., Wheeler, B. C. & Zhao, J. L. (2008). On assessment of project success in community source development. *ICIS 2008 Proceedings*, 11.
- Liu, M., Wu, X., Zhao, J. L. & Zhu, L. (2010). Outsourcing of community source: Identifying motivations and benefits. *Journal of Global Information Management*, 18(4), 36–52. <https://doi.org/10.4018/jgim.2010100103>

-
- Liu, M., Zeng, D. & Zhao, J. L. (2008). A cooperative analysis framework for investment decisions in community source partnerships. *AMCIS 2008 Proceedings*, 12. <https://aisel.aisnet.org/amcis2008/278>
- Liu, M. & Zhao, J. L. (2007). Real options analysis of the community source approach: Why should institutions pay for open source? *Proceedings of the First China Summer Workshop on Information Management*. Retrieved October 7, 2021, from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=IqYuFeAAAAAJ&citation_for_view=IqYuFeAAAAAJ:ihjKul7GT4YC
- Liu, M. & Zhao, J. L. (2008). On outsourcing and offshoring in community source. *2008 IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE)*, 1–3. <https://doi.org/10.1109/AMIGE.2008.ECP.82>
- Mackie, C. J. (2008). Open source in open education: Promises and challenges [Publisher: MIT Press Cambridge, MA]. *Opening up education: The collective advancement of education through open technology, open content, and open knowledge*, 119–131.
- Nelson, K., Nelson, H. & Ghods, M. (1997). Technology flexibility: Conceptualization, validation, and measurement [ISSN: 1060-3425]. *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, 3, 76–87 vol.3. <https://doi.org/10.1109/HICSS.1997.661572>
- Oracle Corporation. (2021). *Oracle announces fiscal 2022 first quarter financial results*. Retrieved January 4, 2022, from <https://investor.oracle.com/investor-news/news-details/2021/Oracle-Announces-Fiscal-2022-First-Quarter-Financial-Results/default.aspx>
- Panettieri, B. J. C. (2007, April 1). *Breaking away* [Campus technology]. Retrieved October 11, 2021, from <https://campustechnology.com/articles/2007/04/breaking-away.aspx>
- Riehle, D. (2016). Open source user foundations.
- Riehle, D. (2018). *How to capture open source user consortia 4/4 - software research and the industry* [Software research and the industry] [Section: 1.3 Open Source Foundations]. Retrieved September 16, 2021, from <https://dirkriehle.com/2018/08/12/how-to-capture-open-source-user-consortia/>
- Riehle, D. (2021, November 8). *User-led open source consortia, defined* [Software research and the industry]. Retrieved January 7, 2022, from <https://dirkriehle.com/2021/11/08/user-led-open-source-consortia-defined/>
- Schwab, B., Riehle, D., Barcomb, A. & Harutyunyan, N. (2020). The ecosystem of openKONSEQUENZ, a user-led open source foundation [Series Title: IFIP Advances in Information and Communication Technology]. *Open Source Systems*, 582, 1–13. https://doi.org/10.1007/978-3-030-47240-5_1
- Severance, C. (2007). Pushing the boundaries of innovation through community source [Publisher: Citeseer], 11.

References

- Taft, D. K. (2009, February 24). *Community-source development appeals in tough times* [eWEEK]. Retrieved October 8, 2021, from <https://www.eweek.com/development/community-source-development-appeals-in-tough-times/>
- Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data [Publisher: SAGE Publications Inc]. *American Journal of Evaluation*, 27(2), 237–246. <https://doi.org/10.1177/1098214005283748>
- Wang, H., Blue, J. & Plourde, M. (2010). Community source software in higher education [Conference Name: IT Professional]. *IT Professional*, 12(6), 31–37. <https://doi.org/10.1109/MITP.2010.120>
- Wheeler, B. (2007). Open source 2010: Reflections on 2007. *EDUCAUSE Review*, 42(1), 49–52. Retrieved September 16, 2021, from <https://www.learntechlib.org/p/100351/>
- Wheeler, B. & Hilton, J. L. (2012). The marketecture of community, 18.
- Wikipedia. (2021, September 28). Community source [Page Version ID: 1047039920]. *Wikipedia*. Retrieved December 30, 2021, from https://en.wikipedia.org/w/index.php?title=Community_source&oldid=1047039920
- Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 1–10. <https://doi.org/10.1145/2601248.2601268>