Taxonomies of Software Ecosystem Health Metrics and Practices: A Systematic Literature Review

by

Arman Yousef Zadeh Shooshtari
B.Sc., Shahid Beheshti University, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science

Taxonomies of Software Ecosystem Health Metrics and Practices:
A Systematic Literature Review

by

Arman Yousef Zadeh Shooshtari
B.Sc., Shahid Beheshti University, 2018

Supervisory Committee

Dr. Margaret-Anne Storey, Supervisor
(Department of Computer Science)

Dr. Daniel M. German, Departmental Member
(Department of Computer Science)

ABSTRACT

Context: Since the beginnings of software engineering, metrics (such as SLOCs) and practices have been used in an attempt to measure and improve the features of software development projects, their process, or their contributors. Measuring and enhancing software ecosystem features brings a new complexity level because a software ecosystem comprises several interrelated software projects. Over the past two decades, software ecosystems have gained considerable attention, and researchers have proposed various metrics and practices to measure and improve software ecosystems' health.

Objective: This thesis presents a systematic literature review that aims to build comprehensive taxonomies for software ecosystem health metrics and practices. These taxonomies synthesize the results of previous categorizations and update them with newer metrics and practices proposed since then. This study also aims to collect and synthesize all the definitions, metrics, and practices proposed to define, measure, and improve software ecosystem health in the literature.

Method: I conducted a systematic literature review and identified 40 primary studies related to defining and measuring software ecosystem health. I extracted the definitions, metrics, and practices for software ecosystem health from the primary studies, and then I categorized the metrics and practices to build the taxonomies.

Results: I identified a total of 7 different definitions for software ecosystem health, 142 different metrics, and 174 various practices for software ecosystem health. Our taxonomies for software ecosystem health metrics and practices have three categories (niche creation, productivity, and robustness). Each of these categories has several sub-categories of metrics and practices.

Conclusion: Software ecosystems have a wide range of stakeholders that have different perspectives regarding software ecosystem health. To satisfy this spectrum, researchers have proposed various metrics and practices to measure and improve software ecosystems' health. To improve unifying contrasting opinions, I conducted this study. The metrics and practices proposed are diverse in both purpose and the data required to compute them. Some metrics are presented along with a method on how to compute them. In contrast, others are defined abstractly without an operational approach to calculate them, and some are mentioned without a clear rationale. Furthermore, the same metric or practice is often proposed in more than one publication using different names. This thesis addresses these alignment problems.

Contents

Supervisory Committee **ii**

Abstract **iii**

Table of Contents **iv**

List of Tables **vii**

List of Figures **ix**

Glossary **x**

Acknowledgements **xi**

Dedication **xii**

1 Introduction **1**

1.1 Problem Domain 2
1.2 Problem Statement 2
1.3 Research Questions 3
1.4 Thesis Contributions 4
1.5 Thesis Overview 5

2 Background and Related Work **7**

2.1 Software Ecosystems and Their Health 7
2.2 Models for Measuring Software Ecosystem Health 8
2.3 Software Ecosystem Health Practices 9
2.4 Gaps and Challenges Discussed in the Literature on Software Ecosystem Health 10

3 Research Method **12**

3.1 Systematic Literature Review 12
3.2 Construction of Taxonomy for Software Ecosystem Health Metrics . . 15
3.3 Construction of Taxonomy for Software Ecosystem Health Practices . 17

4 Findings **18**
4.1 RQ1: How has Software Ecosystem Health been defined in the Literature? 18
4.2 RQ2: What Metrics Have Been Proposed for Evaluating Software Ecosystem Health? 20
4.3 RQ3: What Practices Have Been Proposed for Improving Software Ecosystem Health? 21

5 A Taxonomy of Software Ecosystem Health Metrics **25**
5.1 Niche creation 25
5.2 Productivity 28
5.3 Robustness 33

6 A Taxonomy of Software Ecosystem Health Practices **39**
6.1 Niche creation 39
6.2 Productivity 41
6.3 Robustness 47

7 Discussion **70**
7.1 Metrics 70
7.2 Practices 71
7.3 How My Taxonomies can be Used by Practitioners and Researchers . 73
7.4 Contributions 73
7.5 Implications for Stakeholders 74
7.6 Implications for Researchers 75

8 Limitations **77**

9 Future Work **79**

10 Conclusions **81**

Bibliography **82**

A List of Selected Primary Studies **91**

B List of Included Metrics in My Taxonomy **96**

C List of Excluded Metrics from My Taxonomy **108**

D List of Included Practices in My Taxonomy **118**

E List of Excluded Practices from My Taxonomy **136**

List of Tables

Table 4.1 Definitions of software ecosystem health. 19
Table 4.2 Attributes related to one of the metrics (Geographical members' distribution) 22
Table 4.3 Attributes related to one of the practices (Provide guidelines informing about actions that are allowed and not allowed to keep backward compatibility.) 24

Table 5.1 Some of the metrics for the niche creation category—a single metric is used as an example for each of the sub-categories. 29
Table 5.2 Some of the metrics for the productivity category—a single metric is used as an example for each of the sub-categories. 33
Table 5.3 Some of the metrics for the robustness category—a single metric is used as an example for each of the sub-categories. 38

Table 6.1 Some of the practices as examples for the niche creation category 42
Table 6.2 Some of the practices for the productivity category—a single practice is used as an example for each of the sub-categories. 64
Table 6.3 Some of the practices for the robustness category—a single practice is used as an example for each of the sub-categories. 67

Table A.1 Final set of selected primary studies based on the inclusion and exclusion criteria of the SLR 95

Table B.1 All the 142 metrics I included in my taxonomy (with some of their related attributes) 107

Table C.1 All the 79 metrics I excluded from my taxonomy (with some of their related attributes) 117

Table D.1 All the 174 practices I included in my taxonomy (with some of their related attributes) 135

Table E.1 All the 13 practices I excluded from my taxonomy (with some of their related attributes) 137

List of Figures

Figure 3.1 Number of studies per database. 14
Figure 3.2 Phases of the SLR 15
Figure 3.3 Number of primary studies published per year. We did our search on 09/11/2019, so we have not included the studies after this date. 16

Figure 5.1 Categories and sub-categories of the software ecosystem health taxonomy 26
Figure 5.2 Sub-categories and metrics in the niche creation category within the software ecosystem health taxonomy 30
Figure 5.3 Sub-categories and metrics in the productivity category within the software ecosystem health taxonomy 34
Figure 5.4 Sub-categories and metrics in the robustness category within the software ecosystem health taxonomy 37

Figure 6.1 Categories and sub-categories of the software ecosystem health practices taxonomy 62
Figure 6.2 Some of the practices for the niche creation category—a single practice is used as an example for each of the sub-categories. . . 63
Figure 6.3 Sub-categories and practices in the productivity category within the software ecosystem health practices taxonomy 65
Figure 6.4 Sub-categories and practices in the robustness category within the software ecosystem health practices taxonomy—part 1 . . . 68
Figure 6.5 Sub-categories and practices in the robustness category within the software ecosystem health practices taxonomy—part 2 . . . 69

Glossary

Term	Description
Attribute	Attributes are the information like name, the method to compute, interpretations, etc. I have collected for each metric and practice through my systematic literature review.
Health characteristic	Health characteristics are productivity, robustness, and niche creation introduced by Iansiti et al. [35].
Metric	A metric is a standard for measuring or evaluating something, especially one that uses figures or statistics.
Practice	Practices provide a means to address a particular aspect of a problem systematically and verifiably.
Taxonomy	Taxonomy is the practice and science of classification of things or concepts, including the principles that underlie such classification.
Category	Categories compose the highest level of my hierarchical taxonomies for practices and metrics.
Sub-Category	Sub-Categories compose the middle level of my hierarchical taxonomies for practices and metrics.
Health indicator	Health indicators are categories that our SLR studies categorize the metrics and practices under them.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep and sincere gratitude to my supervisor, Dr. Margaret-Anne (Peggy) Storey, for her mentorship, enthusiasm, and encouragement throughout my study and research stages. Without her invaluable guidance and inspiration, this thesis would not have been possible.

I would also like to thank Dr. Daniel M. German for his support, insightful feedback, and extended discussion, which have contributed to the improvement of this thesis.

To all present and former members of the CHISEL lab, I am grateful for your suggestions on this thesis, our friendship, and all the fun we have had: Soroush Yousefi, Alexey Zagalsky, Andreas Koenzen, Ying Wang, Eirini Kalliamvakou, Jorin Weatherston, Leon Li, Matthieu Foucault, Omar Elazhary, Trishala Bhasin, Neil Ernst, and Cassandra Petrachenko.

A special thanks to Omar for sharing his insights, knowledge, and experience that greatly assisted my research; Jorin for providing helpful advice and generous support for my work; Cassie for her immediate help when I needed it and her thoughtful editing of this thesis as well as other documents.

I was fortunate to collaborate with Soroush in my research. I appreciate his great support, valuable feedback and participation in the evaluation of this research.

Lastly, I would like to thank my parents, wife, and friends for their love, understanding, and support along the way.

Arman Yousef Zadeh Shooshtari

DEDICATION

For everyone is interested in measuring and improving the health of software ecosystems.

Chapter 1

Introduction

Over the past two decades, a great deal of work has been dedicated to studying software ecosystems (SECOs) from a business, social, and technological viewpoint [9,70]. However, the different types of stakeholders like developers, end-users, investors, etc. and researchers still find it challenging to evaluate or improve software ecosystems because they have different perspectives regarding the health of software ecosystems, and many metrics and practices are corresponding to these different viewpoints. Also, there is no comprehensive categorization of metrics and practices that help the stakeholders select the metrics and practices based on their specific perspectives and needs regarding the software ecosystems. In this thesis, we try to fill this gap by building comprehensive taxonomies of metrics and practices which help various stakeholders measure and improve software ecosystems from different viewpoints regarding the software ecosystem.

Although there are different definitions for SECO in the literature [8,39,40,46,49], in this thesis, I use the following definition: *"a collection of software projects, which are developed and co-evolve in the same environment. The environment can be organizational (e.g., a company), social (e.g., an open-source community), or technical (e.g., the Ruby ecosystem)."* [44]

Software ecosystems provide organizations with several advantages: they can speed up innovation, help disperse innovation costs, and reduce software maintenance costs by sharing activities with other members of the ecosystem [6]. As a result, many companies now rely on SECOs to meet their technological or business needs. Some of the most successful companies that take advantage of this approach include Apple, Amazon, and Google.

1.1 Problem Domain

Deciding to rely on, use, or extend an ecosystem has always been a challenge for developers, practitioners, business analysts, architects, and active stakeholders in a software organization. For example, suppose a project relies on an ecosystem that is not secure enough; the software's security in the project can also become vulnerable due to depending on an insecure software ecosystem. So deciding on selecting an ecosystem always introduces some risks that should be evaluated by some metrics. This decision also can affect much of an organization's future development. Organizational decision-makers also face many challenges in choosing which open innovation communities to engage with as an alliance partner [62].

To assist in this crucial and often risky decision-making process, Iansiti and Levien [34] introduced the concept of SECO health, stating that *"a healthy ecosystem provides durably growing opportunities for its members and for those who depend on it. A healthy ecosystem keeps working and growing efficiently, as well as surviving crisis and generating innovation."* Defining health was not enough, later work by Jansen [37] revealed the importance of operationalizing the concept of health so that it can be measured. Over the past few years, significant research effort has been spent on investigating methods to assess software ecosystems. These papers range from proposing ways to measure open source projects, such as [13, 71, 74], to concentrating on the definition and implementation of metrics that evaluate one or more aspects of software ecosystems, such as [43, 51, 69].

Software ecosystems have become a significant contributor to software development. Software ecosystems consist of several software development projects, and the projects can decide on joining a software ecosystem. It is common for a software development project to assess and improve the health of the software ecosystem they join (e.g., reuse a library developed by an ecosystem).

1.2 Problem Statement

This thesis tries to address the following problems. A significant challenge for both practitioners and researchers is the lack of a comprehensive catalog of metrics and practices proposed in the literature. A clear description of which metrics can be used to measure different aspects of ecosystem health is needed because different stakeholders based on their need and perspectives regarding the ecosystem health concept

need to measure ecosystem health, and also a clear description of which practices can be applied to improve various aspects of ecosystem health is necessary because the stakeholders based on their priorities and viewpoints about software ecosystem health want to apply the practices to improve the software ecosystem health. It is not that metrics and practices are not presented in other literature, they are discussed in many papers. But one problem is that the metrics and practices already proposed are very diverse in both their purpose and in terms of the data required to compute them. Some metrics and practices are presented along with a method on how to compute them, while, others are defined abstractly without an operational approach to calculate them, and some are mentioned without a clear rationale. That is making sense of the proposed metrics and practices, and understanding how they relate to each other and how they have been used and validated is hard to fathom. Furthermore, sometimes the same metric or practice is often proposed in more than one publication using different names, while sometimes a similarly named metric or practice is defined in different ways.

Although several researchers have proposed a variety of metrics and practices to evaluate and improve SECO health, no previous work has provided such comprehensive taxonomies that categorize all of the metrics and practices that have been defined in the literature, nor do they summarize which of these metrics and practices have been evaluated and their rationale. Also, comprehensive and fine-grained categorizations of metrics and practices lacked in the literature [27], which is a contribution from this thesis. Using these comprehensive taxonomies, researchers can identify software ecosystem health aspects that there are not enough metrics and practices to evaluate and improve them. Researchers can create new metrics and practices to measure and improve these aspects of software ecosystem health. Besides, practitioners like different stakeholders in software ecosystems can apply our taxonomies' metrics and practices to measure and improve the health of software ecosystems based on their needs, priorities, and goals. Using our taxonomies, the practitioners can save a lot of time and effort by using only the metrics and practices of their favorite and important categories and gaining a quick insight into any software ecosystem's health.

1.3 Research Questions

As mentioned above, my goal in this thesis is to build comprehensive taxonomies for all the metrics and practices proposed in the literature to measure and improve the

software ecosystem's health. The first step in reaching this goal was collecting all the literature's metrics and practices and then categorizing them to build taxonomies. As described above, one of my taxonomies' critical goals is to help researchers define new metrics and practices for the categories of our taxonomy in which there is a lack of metrics and practices. Researchers should know what software ecosystem health means to define new metrics and practices to measure and improve software ecosystem health because they may create some new metrics or practices that measure something else instead of software ecosystem health. So they need a definition to check the relation of their new metric or practice with the concept of software ecosystem health. Defining a concept before creating new metrics for measuring it is not only necessary for the concept of software ecosystem health. For example, for creating new metrics for measuring a person's happiness, we need first to define happiness because if we do not have a clear definition, we may create metrics that measure something else like sadness instead of happiness. As a result, I needed to collect all the existing definitions for software ecosystem health from the literature and synthesize them to define software ecosystem health.

Since I needed to collect all the metrics, practices, and definitions proposed for defining, measuring, and improving software ecosystem health in the literature. I defined the following research questions:

RQ1: How has software ecosystem health been defined in the literature?

RQ2: What metrics have been proposed for evaluating software ecosystem health in the literature?

RQ3: What practices have been proposed for improving software ecosystem health in the literature?

To answer the above research questions, I conducted a systematic literature review because it is a method to identify, evaluate, and interpret the available research relevant to a particular topic, research question, or phenomenon of interest [41].

1.4 Thesis Contributions

1. A systematic literature review (SLR) that results in identifying 40 primary studies that describe practices, theories, approaches, issues, definitions, and/or metrics related to measuring software ecosystems' health.

2. Through my systematic literature review, I collected seven definitions for software ecosystem health for answering my first research question. I also collected 221 metrics with their related details such as name, methods to compute them, their interpretation, etc. for answering my second research question. Besides, I collected 188 practices with their related details, like the study that created them, etc. to answer my third research question.

3. This SLR resulted in creating two hierarchical taxonomies with three top-level categories that organize similar metrics and practices based on their purpose. These top-level categories, which were previously defined by Iansiti et al. [35], are "Niche creation", "Productivity", and "Robustness". These two taxonomies serve several purposes for both practitioners and researchers. First, they can be used to quickly discover which metrics and practices have been created for a specific purpose; second, they help document whether a metric or practice has been implemented and empirically evaluated; and third, they show specific areas in which there is a lack of metrics and practices and evaluation of ecosystem health metrics and practices.

4. The two hierarchical taxonomies that I built based on my SLR provide comprehensive and fine-grained categorizations of metrics and practices lacked in the literature [27]. In addition to three main categories which are "Niche creation", "Productivity", and "Robustness", my taxonomy of metrics has 27 subcategories and my taxonomy of practices has 31 sub-categories which are the most comprehensive categorizations of metrics and practices built so far to the best of our knowledge.

1.5 Thesis Overview

This thesis is structured as follows.

Chapter 2 Background and Related Work reviews the background and related work. This chapter covers the important works in the area of software ecosystem health. I also discuss the research that has tried to categorize software ecosystem health metrics and practices and identify why those need further research.

Chapter 3 Research Method describes my research method for the systematic literature review and the creation of my taxonomies. This chapter describes the SLR

steps in detail and the procedures I followed to build the taxonomies for software ecosystem health metrics and practices.

Chapter 4 Findings presents the findings collected from the SLR. These findings include definitions, metrics, and practices that define, measure and improve software ecosystem health. In this chapter, I also describe the attributes I consider for each of the metrics and taxonomies. I suggest a definition for software ecosystem health that I created by synthesizing the previous definitions proposed in the literature.

Chapter 5 A Taxonomy of Software Ecosystem Health Metrics describes the taxonomy I built based on the metrics from the SLR. I describe all the categories and sub-categories of the taxonomy, and also I present the metrics in each of these categories and sub-categories.

Chapter 6 A Taxonomy of Software Ecosystem Health Practices describes the taxonomy I built based on the practices from the SLR. I describe all the categories and sub-categories of the taxonomy, and also I present the practices in each of these categories and sub-categories.

Chapter 7 Discussion includes the discussion about my findings and taxonomies. This chapter discusses the contributions and importance of my work. It also discusses the implications of my work for various practitioners and researchers.

Chapter 8 Limitations mentions the limitations which I faced in my research. Some of these limitations are in conducting the systematic literature review, and the rest are in building the taxonomies based on the SLR findings.

Chapter 9 Future work presents the future work that can be done based on my work. The information presented in this chapter is an important contribution to my work and gives a clear research path for researchers who want to research software ecosystem health.

Chapter 10 Conclusions concludes the thesis and provides a summary of the work completed.

Chapter 2

Background and Related Work

This chapter covers important work in the field of software ecosystem health. I also address the research that tried to categorize the software ecosystem's health metrics and practices and identify why more research is needed in this area.

2.1 Software Ecosystems and Their Health

Moore [53, 54] was the first to use the term 'business ecosystem' and its derivatives, such as SECO, as critical conceptualizations of today's business networks. Moore [53] defined a business ecosystem as a complex network of organizations and individuals involved in a service or product being produced or distributed.

Following Moore's work, some researchers described SECO as a specific type of business ecosystem. For example, Manikas and Hansen [46] pointed out that SECOs are business ecosystems where actors' interactions are centered on a standard software technology or platform. Similarly, Hyrynsalmi et al. [29] conceptualized that SECOs are business ecosystems where software forms a focal part of the exchange unit. The term SECO has also been used to point to a wide range of software ecosystems, from mobile software ecosystems that produce software or applications for Smartphones (e.g., Apple iOS and Google Android) [22] to open-source ecosystems based on distributed code repositories (e.g., KDE) [26].

Moore [53] stated that *"the survival of an individual actor in a business ecosystem depends on the entire network and the survival of the ecosystem depends on the choices and agency of the individual actors"*. Given the importance of this, many researchers have investigated business and software ecosystems, and looked at ways to measure

their health [3, 21, 32, 35, 37, 45].

Iansiti et al. [35] established three health characteristics for business ecosystems based on biological ecosystems: productivity, robustness, and niche creation. They described these three health characteristics as follows.

Niche creation "refers to the ability to create value by putting new functions into operation to increase meaningful diversity in the ecosystem. Diversity gives an ecosystem potential for productive innovation and indicates its ability to absorb shocks from outside."

Productivity "can be measured as a return on the capital invested or the economic value added from tangible and intangible assets created while producing goods or services. This refers to a biological ecosystem's ability, e.g., create biomass from inputs such as sunlight."

Robustness "is measured in the survival rate of the ecosystem's members, either in relation to other ecosystems or over time. Robustness means that the ecosystem can face and survive changes in the environment."

2.2 Models for Measuring Software Ecosystem Health

After introducing these three health characteristics, several researchers have used or extended them to evaluate the health of SECOs. For instance, den Hartigh et al. [23] added *network health*, and *partner health* components to their model and found the relation between these two components with the three health characteristics introduced by Iansiti et al. [35]. In their model, *partner health* is a long-term, financially-based representation of a partner's strength of management and of its competencies to exploit opportunities that arise within the ecosystem, and *network health* is a representation of how well a partner is embedded in the ecosystem as well as the impact the partner has in its local network. Likewise, Ben Hadj Salem Mhamdia [4] expanded the model of Iansiti et al. [35] and evaluated an ecosystem's health with *robustness, productivity, interoperability, the satisfaction of stakeholders, and creativity.* Also, Carvalho et al. [12] added *sustainability and diversity* to the model of Iansiti et al. [35] for the evaluation of SECOs.

Manikas and Hansen [45] proposed a logical framework to define and measure SECO health, which consisted of the network of actors, the health of each actor, each

software component, platform, software-network, and orchestrator. Their framework added a new viewpoint for the evaluation of ecosystem health. However, in building the framework, they did not consider that a SECO can be based on a common standard rather than just a shared software platform [38].

Jansen [37] acknowledged the absence of ecosystem health operationalization in his review of the literature. To fill this gap, Jansen [37] introduced OSEHO, an open-source ecosystem health model based on the health characteristics identified by Iansiti et al. [35]. Although Jansen's model is comprehensive, the framework only applies to open-source software ecosystems and not SECO types. Another significant contribution of his work was distinguishing between health at the project level and the ecosystem level.

Shaikh and Levina [62] proposed seven characteristics for measuring SECO health. These include *strength of ecosystem partners, level of support by partners, commercial acceptance of the chosen license regime, modularity of the platform, ability to reuse components and complementary products, ecosystem governance structures, and powerful influencers in the ecosystem.* While some of these characteristics were taken from previous works, some were proposed for the first time and added a new view for evaluating SECOs.

2.3 Software Ecosystem Health Practices

Practices provide a means to address a particular aspect of a problem systematically and verifiably. They address a specific part of a problem rather than addressing the whole issue [36]. Practices have a direct effect on the health of software ecosystems. Practices produce results which, whether good or bad, can be expressed by metrics [21].

Da Silva Amorim et al. [18] stated that the software platform should be strong enough to attract developers from third parties to create and maintain applications on the platform. The software architecture in this environment is a crucial point that should support all the community's demands. The architecture of ecosystems enables communication and knowledge management by Sharing the information between outside and internal stakeholders. It fosters alignment between technical problems and business objectives and defines the employment scope for developers [56]. In addition to their peculiarities, software ecosystem architectures face a set of performance-influencing challenges. Because of these challenges, the organizations adopted several

architectural practices to help maintain the products' performance and health [7]. Da Silva Amorim et al. [18] started researching the architectural practices employed by open-source software ecosystems to face architectural challenges and analyze their effect on software ecosystem health.

Da Silva Amorim et al. [16] claimed that the health of a SECO should not be assessed by considering metrics alone. Practices can be useful for understanding a SECO and for evaluating its health. For example, the practice "review all code before accepting into the release" may impact several quality indicators and increase productivity while avoiding rework. They presented the findings of an ethnographic study conducted to examine SECO's practices from three perspectives—business, social and technological—and their effect on SECO health.

Da Silva Amorim et al. [20] elaborated that experienced members of a software ecosystem already have the expertise to create and maintain appropriate practices. Newcomers are nonetheless inexperienced and should be trained in knowing and applying the practices adopted. Based on the training they have received, they will develop their way of working by community rules. Adequate training will contribute to the efficient use of architectural practices that affect the health of the ecosystem. For this reason, they realized the need to investigate architectural practices that are taught in the training of newcomers. They analyzed how these practices influenced health characteristics. In this way, they could set an example scenario describing the education newcomers receive to achieve healthy open-source ecosystems.

2.4 Gaps and Challenges Discussed in the Literature on Software Ecosystem Health

Several systematic literature reviews on SECO health have been published [3, 21, 32, 45]. In the most recent SLR, da Silva Amorim et al. [21] identified six different definitions for SECO health, more than 200 metrics for assessing health, and 19 practices. However, none of these SLRs have created comprehensive taxonomies for the metrics and practices. None of the existing SLRs have synthesized all of these definitions, metrics, and practices and put them in categories based on different perspectives regarding the software ecosystem health concept. Besides, the conducted SLRs have not investigated which of these metrics have been used in practice to measure software ecosystem health. In this thesis, I am trying to take the first steps to fill these gaps.

Several researchers have challenged the existing literature on SECO health. Hyrynsalmi et al. [30] delivered a criticism of ambiguous definitions and the need for a redefinition of the terms. In another paper, Hyrynsalmi et al. [31] also questioned the current literature by pointing out three criticisms: (1) It has yet to be examined whether the existing ecosystem health metrics would function proactively or if the metrics would only be reactive, describing the previous incidents; (2) For most ecosystem health metrics, the natural evolution of the ecosystems [59,66] has not been considered; (3) It is not evident whom the ecosystem health metrics are intended for (for example, ecosystem developers, newcomers or customers).

To sum up, although several researchers have proposed a variety of metrics and practices to evaluate and improve SECO health, no previous work has provided comprehensive taxonomies that categorize all of the metrics and practices that have been defined in the literature, nor do they summarize which of these metrics and practices have been evaluated and their rationale. Also, comprehensive and fine-grained categorizations of metrics and practices lacked in the literature [31]. The purpose of this work is to fill this gap.

Chapter 3

Research Method

My research methodology consists of two main steps. First, I conducted a systematic literature review to collect necessary data to build the taxonomy. Then I followed a mixture of bottom-up and top-down approaches to synthesize the first step's findings and create two taxonomies of ecosystem health metrics and practices to improve ecosystem health. I explain the techniques followed in each step of the SLR throughout the rest of this chapter.

3.1 Systematic Literature Review

To obtain an overview of the research literature on the measurement of SECO health, I performed a systematic literature review, a method to identify, evaluate, and interpret the available research relevant to a particular topic, research question, or phenomenon of interest [41]. By conducting an SLR, I aimed to identify and interpret the available research relevant to the following research questions:

RQ1: How has software ecosystem health been defined in the literature?

RQ2: What metrics have been proposed for evaluating software ecosystem health in the literature?

RQ3: What practices have been proposed for improving software ecosystem health in the literature?

I performed my systematic literature review based on the guidelines described by Kitchenham and Charters [41]. I selected the following four electronic databases

for my search: ACM Digital Library, IEEEXplore, ScienceDirect, and SpringerLink. Appropriate search terms are important to properly and effectively search for relevant studies. In this respect, Kitchenham and Charters [41] propose viewpoints related to population, intervention, comparison, and outcome (PICO), which SLRs have widely utilized [1, 57, 75]. I do not have a clear intervention and comparison in this study; however, the relevant terms for population and the outcome are as follows:

Population: Industry groups and application areas are considered a population. In my research, I chose **software ecosystem** as my population.

Outcome: By providing taxonomies and an overview of software ecosystem health, metrics, and practices adopted in this field, I help practitioners measure how **healthy** these ecosystems are and improve their **health**.

To maintain search consistency among the multiple databases in my study, I constructed the following search string based on the PICO structure:

Search string: *"software ecosystem" AND (health OR healthy)*

Then I defined inclusion and exclusion criteria to be sure that relevant studies would be selected. I applied the following inclusion criteria: (1) The study must describe practices, theories, approaches, issues, definitions, and/or metrics related to measuring the health of SECOs; (2) The study must be unique, i.e., if a study was published in more than one venue, the complete version was used. I used the following exclusion criteria: (1) Studies written in languages other than English; (2) Studies only available as abstracts or PowerPoint presentations.

To conduct my SLR, manage a large number of references, and remove duplicate studies, I used a tool called StArt [1](State of the Art through Systematic Review). I performed the search phase on 09/11/2019, and all the papers published before this date were considered in the search. In the search phase, I searched the electronic databases with my search string, collecting 364 papers. Figure 3.1 shows the number of search results per database. In the selection phase, I applied the inclusion and exclusion criteria, taking into account the abstract, title, and keywords of each study. In the selection phase, 42 studies were accepted for the extraction phase, and the rest of the papers were rejected. In the extraction phase, inclusion and exclusion criteria were applied again, taking into account the full content of the 42 papers. In

[1] `http://lapes.dc.ufscar.br/tools/start_tool`

the extraction phase, 27 papers were considered as primary studies. Additionally, I used a backward and forward snowballing technique to find other relevant papers [73], which added another 13 studies to the list of primary studies, bringing my total to 40 studies. I present the list of primary studies in the appendix. Figure 3.2 shows the phases of my SLR. Figure 3.3 indicates the number of primary studies published per year.

The details of each phase of my SLR, including the papers accepted and rejected in each phase, all the found papers in the search phase from the databases, and duplicated papers, are available in `https://github.com/Armanyousefzade/Systematic-Literature-Review`.

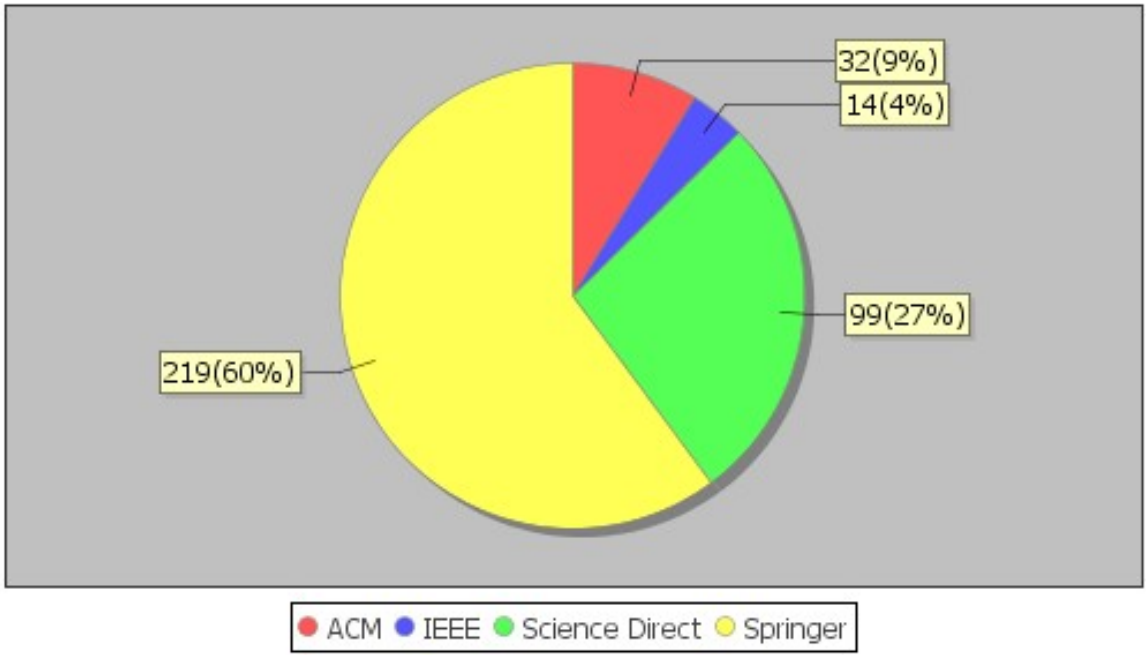

Figure 3.1: Number of studies per database.

I captured the SECO health definitions mentioned in the primary studies selected in the data collection phase. I also collected any metrics and practices used to measure and improve SECO health described in these studies.

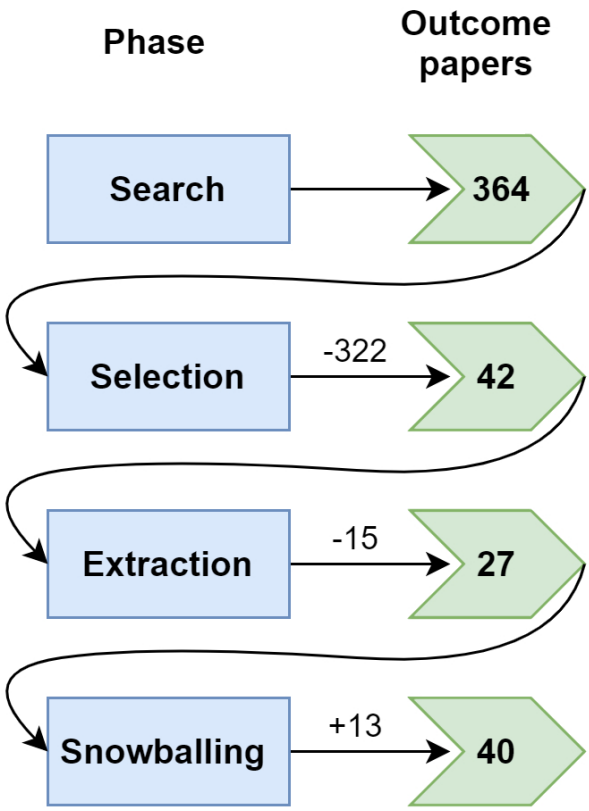

Figure 3.2: Phases of the SLR

3.2 Construction of Taxonomy for Software Ecosystem Health Metrics

I used both bottom-up and top-down approaches to build the taxonomy for the metrics collected during my SLR [60]. I used a top-down approach because most of my primary studies were unanimous on the three high-level categories of metrics: productivity, robustness, and niche creation. Hence, I included these three high-level categories in my taxonomy. I also used a bottom-up approach because comprehensive and fine-grained categorizations of metrics lacked in the literature [31]. As part of this, I grouped the metrics I found into sub-categories and then aligned the sub-categories with the three high-level categories.

I built my taxonomy in two main phases. In the first phase, I followed the bottom-up approach mentioned above. I took the set of metrics collected in the SLR as the starting point and then used a card sorting process [65] to cluster them. After clustering the metrics, I assigned a label to each cluster based on the concept that the

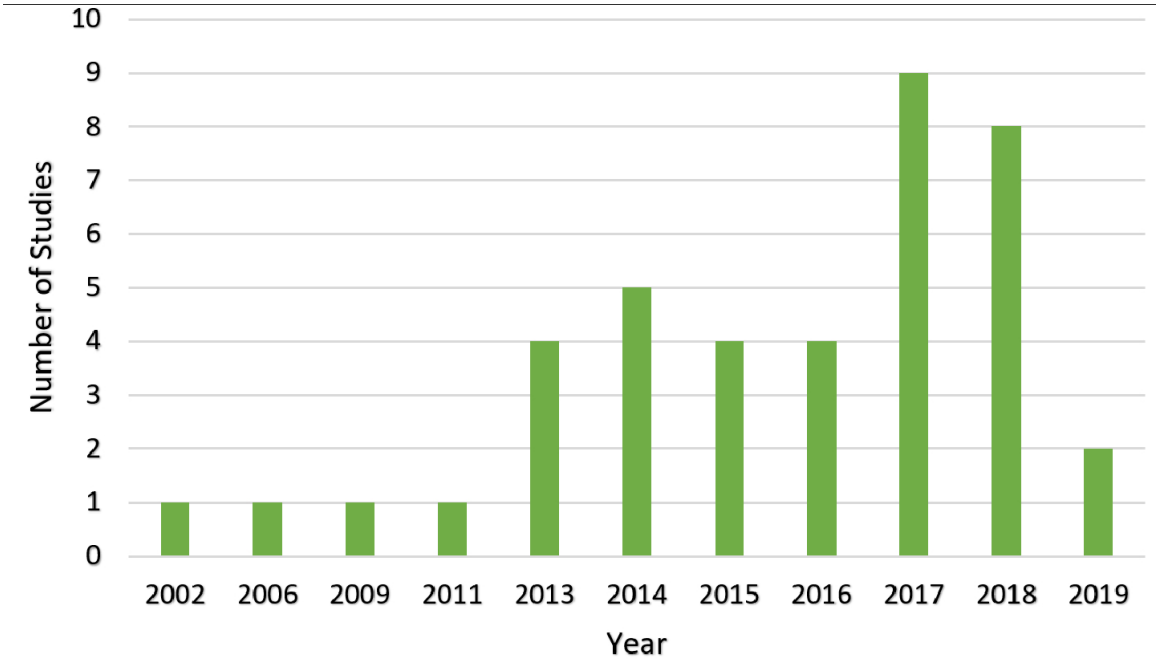

Figure 3.3: Number of primary studies published per year. We did our search on 09/11/2019, so we have not included the studies after this date.

cluster's metrics measured. I refer to these clusters as sub-categories in my taxonomy. In the second phase, I followed a top-down approach. I used the previously defined three categories in the taxonomy and assigned each sub-category to one of these three categories. As mentioned, each top-level category was related to one of the health characteristics introduced by Iansiti et al. [35], and the assignment of a sub-category to a category was based on the health characteristic being measured by the metrics in that sub-category.

It is worthwhile to mention that clustering the metrics was based on their direct and not indirect relations on software ecosystem health. For example, "employee satisfaction rate" is a metric in the "satisfaction" sub-category, although this metric has an indirect relation with "Survival in ecosystem" as another sub-category. It is common in different taxonomies that categories may overlap if we also consider indirect relations. Still, when we only consider more direct relations, the categories are completely independent of each other.

As some bias may have been introduced in clustering, if only one researcher performed it, I recruited a second researcher (a PhD student in my research group) to cluster the metrics independently. We compared our clusters, and when we saw some differences, we discussed the differences until we reached an agreement. There were

only a few cases (about 3%) where we could not reach an agreement, and in these cases, I made the final decision on how to cluster as I had more experience in the topic.

3.3 Construction of Taxonomy for Software Ecosystem Health Practices

After constructing a taxonomy for metrics with a PhD student's assistance in my research group, I had gained enough context and experience to build a taxonomy for practices on my own.

I built the taxonomy for practices similar to how I built taxonomy for metrics. I used both bottom-up and top-down approaches to build the taxonomy for the practices collected during my SLR. I used a top-down approach because most of my primary studies were unanimous on the three high-level categories of practices: productivity, robustness, and niche creation. Hence, I included these three high-level categories in my taxonomy. I also used a bottom-up approach because comprehensive and fine-grained categorizations of practices lacked in the literature. As part of this, I grouped the practices I found into sub-categories and then aligned the sub-categories with the three high-level categories.

I built my taxonomy in two main phases. In the first phase, I followed the bottom-up approach mentioned above. I took the set of practices collected in the SLR as the starting point and then used a card sorting process to cluster them. After clustering the practices, I assigned a label to each cluster based on the concept that the cluster's practices improved. I refer to these clusters as sub-categories in my taxonomy. In the second phase, I followed a top-down approach. I used the previously defined three categories in the taxonomy and assigned each sub-category to one of these three categories. As mentioned, each top-level category was related to one of the health characteristics introduced by Iansiti et al. [35], and the assignment of a sub-category to a category was based on the health characteristic being improved by the practices in that sub-category.

I described my research methodology for the systematic literature review and the creation of my taxonomies for metrics and practices in this chapter. In the next chapter, I present the findings collected from my SLR. These findings include definitions, metrics, and practices that define, measure, and improve software ecosystem health.

Chapter 4

Findings

This chapter summarizes the findings from my systematic literature and the answers to my research questions.

4.1 RQ1: How has Software Ecosystem Health been defined in the Literature?

I collected definitions of SECO health from each of the primary studies found in my SLR. I observed that the studies defined and described SECO health in seven ways. Some of these definitions are partially overlapping. Table 4.1 shows all of the definitions and the primary studies used each definition.

Hyrynsalmi et al. [32] performed an SLR covering various SECO health concepts, and da Silva Amorim et al. [21] conducted an SLR that identified a variety of definitions for SECO health. These studies reported a wide range of descriptions of ecosystem health. Still, I needed to extract the definitions for SECO health from my primary studies to contextualize my findings and boost my comprehension.

Da Silva Amorim et al. [21] collected the definitions for a healthy software ecosystem during the execution of an SLR and synthesized them to create the following definition: *"A healthy software ecosystem has the capacity of keeping their productivity and attractiveness, facing problems, disruptions and junctions. At the same time, they also monitor and implement advances in their strategies to achieve success over time. This success should include all their internal elements considering their interactions and dependencies."*

Some of the primary studies used the SECO health term without defining it, and

Definition	**Studies**
"A healthy ecosystem provides durably growing opportunities for its members and for those who depend on it. A healthy ecosystem keeps working and growing efficiently, as well as surviving crisis and generating innovation."	[S11, S19, S21]
"The healthiness of software is defined as a degree of a healthy software ecosystem, which means that a firm in a healthy software ecosystem can easily reach its financial goal better than other firms in other SECOs."	[S10]
"Health is a term from biology, which refers to a system's status or a specific species. Like with natural ecosystems, a business ecosystem's health tells us something about the system's longevity and propensity for growth."	[S2, S4, S5, S14, S15, S16, S17, S23, S24, S29, S31, S32]
"Health refers to how well the ecosystem is functioning—its ability to endure and remain variable and productive over time."	[S11, S12, S13, S33]
"The well-functioning of a software ecosystem, its strength and longevity are named health."	[S26]
"The software ecosystem's health reflects its capacity to grow and meet the ecosystem community's needs."	[S31, S34]
"A healthy software ecosystem has the capacity of keeping its productivity and attractiveness, facing problems, disruptions, and junctions. At the same time, they also monitor and implement advances in their strategies to achieve success over time. This success should include all their internal elements considering their interactions and dependencies."	[S18]

Table 4.1: Definitions of software ecosystem health.

as I mentioned earlier, several studies provided different definitions of SECO health. This variety of interpretations has created a wide range of metrics to evaluate SECO health. I observed that primary studies that have defined SECO health differently measure SECO health by different metrics and from different perspectives.

I synthesized the definitions across the set of definitions shown in Table 4.1 and suggest the following new definition:

> A software ecosystem is healthy if it: provides durably growing opportunities for its members and for those who depend on it; keeps working and growing efficiently; survives crisis; generates innovation; provides the possibility for firms in it to reach their financial goals better than the firms not in the ecosystem; has a propensity for growth; grows continuously; can endure, remains variable and productive over time, and meets its community's needs.

4.2 RQ2: What Metrics Have Been Proposed for Evaluating Software Ecosystem Health?

Most of the primary studies used metrics or measures to evaluate SECO health. I collected all the metrics related to SECO health from the primary studies in my SLR. Most of the primary studies in my SLR explicitly present the metrics in tables, images, or content of the studies, so identifying and collecting the metrics was straightforward. If any other researchers want to repeat the process of collecting metrics from my primary studies, they will likely identify the same number of metrics that I did. I observed that several attributes constitute a metric, the number of these attributes was different in each study, and each study provided a sub-set of the following eleven attributes for each metric:

Name. Name of the metric as used by the primary studies.

Definition. Question, a stakeholder, may have that use of the metric answers.

Method. How the metric is to be computed.

Procedure. How the data needed for the metric is to be collected.

Interpretation. How the metric should be interpreted.

Source. Data needed to compute the metric.

Data Type. The type of data that the metric can have (e.g., Boolean, numerical, etc.).

Study-Proposed. Publications that proposed the metric.
Study-Used. Studies that applied the metric to measure the health of a SECO.
Category. The categories the metric falls under in the primary studies.
Type of Ecosystem. The type of ecosystem where the metric may be used.

I collected all the above eleven attributes for each metric. When a study had not provided one of these attributes, I mentioned the value of that attribute with *Not provided* word.

I observed that some of the primary studies proposed the same metrics but with different names. I also found that some of the metrics had the same name in various primary studies, but they measured different attributes. I did my best to align duplicated metrics. Also, since my goal is to build a taxonomy of metrics that can be used to measure the health of SECOs, I excluded metrics that did not have a clear implementation. For example, Van Lingen et al. [67] proposed *perceived ecosystem health* as a metric but did not provide any methods to measure and interpret it. As it does not have a clear implementation, I excluded it from my taxonomy.

I collected 221 metrics but excluded 79, which were duplicates or did not have a clear implementation. I placed all the metrics I collected (with their related attributes) in two lists: one list contains the 142 metrics included in my taxonomy. The other list includes the 79 excluded metrics. Both of these lists of metrics and their detailed attributes are available at `https://github.com/Armanyousefzade/Software-ecosystem-health`. These lists are also available in the Appendix chapter. I show an example of the attributes for one metric in Table 4.2. Chapter 5 describes the taxonomy I built based on the metrics from the SLR. I describe all the categories and sub-categories of the taxonomy, and also I present the metrics in each of these categories and sub-categories.

4.3 RQ3: What Practices Have Been Proposed for Improving Software Ecosystem Health?

Some of the primary studies suggested practices to improve SECO health. I collected all the practices related to SECO health from the primary studies in my SLR. For each practice, I gathered the following five attributes:
Name. Name of the practice as used by the primary studies.
Key areas. The area in which the practice improves SECO health, as suggested by

Title of attribute	**Value of attribute**
Name	Geographical members' distribution
Definition	Are the members of the SECO's community geographically distributed?
Method	Identify the geographical location of members from the mailing lists. Count the number of different geographical locations (e.g., countries).
Procedure	Data base query.
Interpretation	More is better. More geographical distribution of members implies more heterogeneity.
Source	SECO mailing lists
Data Type	Numeric
Study-Proposed	[S1]
Study-Used	Not provided
Category	Heterogeneity, Visibility
Type of Ecosystem	Open-source ecosystem

Table 4.2: Attributes related to one of the metrics (Geographical members' distribution)

the primary studies.

Study-Proposed. Publications that proposed the practice.

Health indicator. The health indicator the practice is related to, as suggested by the primary studies.

Type of ecosystem. The type of ecosystem where the practice may be used, as suggested by the primary studies.

I observed that some of the primary studies proposed the same practice but with different names. I also found that some of the practices had the same name in different primary studies, but they had different attributes. I did my best to align duplicated practices. Also, since my goal is to build a taxonomy of practices that can be used to improve the health of SECOs, I excluded practices that did not have a clear implementation. For example, Wnuk et al. [72] proposed *Partner development programs* as a practice but they did not provide any clear description for it. As it does not have a clear implementation, I excluded it from my taxonomy.

I collected 188 practices but excluded 13, which were duplicates or did not have a clear implementation. I placed all the practices I collected (with their related attributes) in two lists: one contains the 174 practices included in my taxonomy, and the other lists the 13 excluded practices. Both of these lists of practices and their detailed attributes are available at `https://github.com/Armanyousefzade/Software-ecosystem-health-practices`. These lists are also available in the Appendix chapter. I show an example of the attributes for one practice in Table 4.3. Chapter 6 describes the taxonomy I built based on the practices from the SLR. I describe all the categories and sub-categories of the taxonomy, and also I present the practices in each of these categories and sub-categories.

In this chapter, I presented the findings collected from my SLR. In the next chapter, I describe the taxonomy I built for the metrics from the SLR. Besides, I explain all the categories and sub-categories of the taxonomy and the metrics in each category and sub-category.

Title of attribute	**Value of attribute**
Name	Provide guidelines informing about actions that are allowed and not allowed to keep backward compatibility.
Key areas	Technical (related to product development (core and applications), technologies used, code rules, among others)
Study-Proposed	S20
Health indicator	Productivity, Niche Creation
Ecosystem	KDE

Table 4.3: Attributes related to one of the practices (Provide guidelines informing about actions that are allowed and not allowed to keep backward compatibility.)

Chapter 5

A Taxonomy of Software Ecosystem Health Metrics

This chapter describes the taxonomy I created for software ecosystem health metrics, consisting of three interconnected dimensions in a hierarchical structure: categories, sub-categories, and metrics. As mentioned above, at the highest level, the taxonomy has three top-level categories: *Productivity, Robustness, and Niche creation.* At the middle level, several sub-categories are assigned to each category. At the lowest level of the taxonomy, the metrics clustered in each sub-category are displayed.

In the rest of this chapter, I describe the categories and sub-categories in the taxonomy. Figure 5.1 shows an overview of the categories and sub-categories in the taxonomy. To avoid a lengthy exposition, I do not discuss all of the following metrics, but I mention the metrics applied to measure health in previous research. As mentioned above, all of the metrics and their related attributes are available in detail at `https://github.com/Armanyousefzade/Software-ecosystem-health`.

5.1 Niche creation

Metrics in the Niche creation category measure an ecosystem's ability to produce value by increasing diversity [3, 35]. Table 5.1 presents the sub-categories in this category and includes example attributes from a single metric for each sub-category. Figure 5.2 shows all sub-categories and all metrics in the niche creation category.

Size of ecosystem (people). The eight metrics in this sub-category measure the number of members in a SECO. There are different types of members in a SECO, such

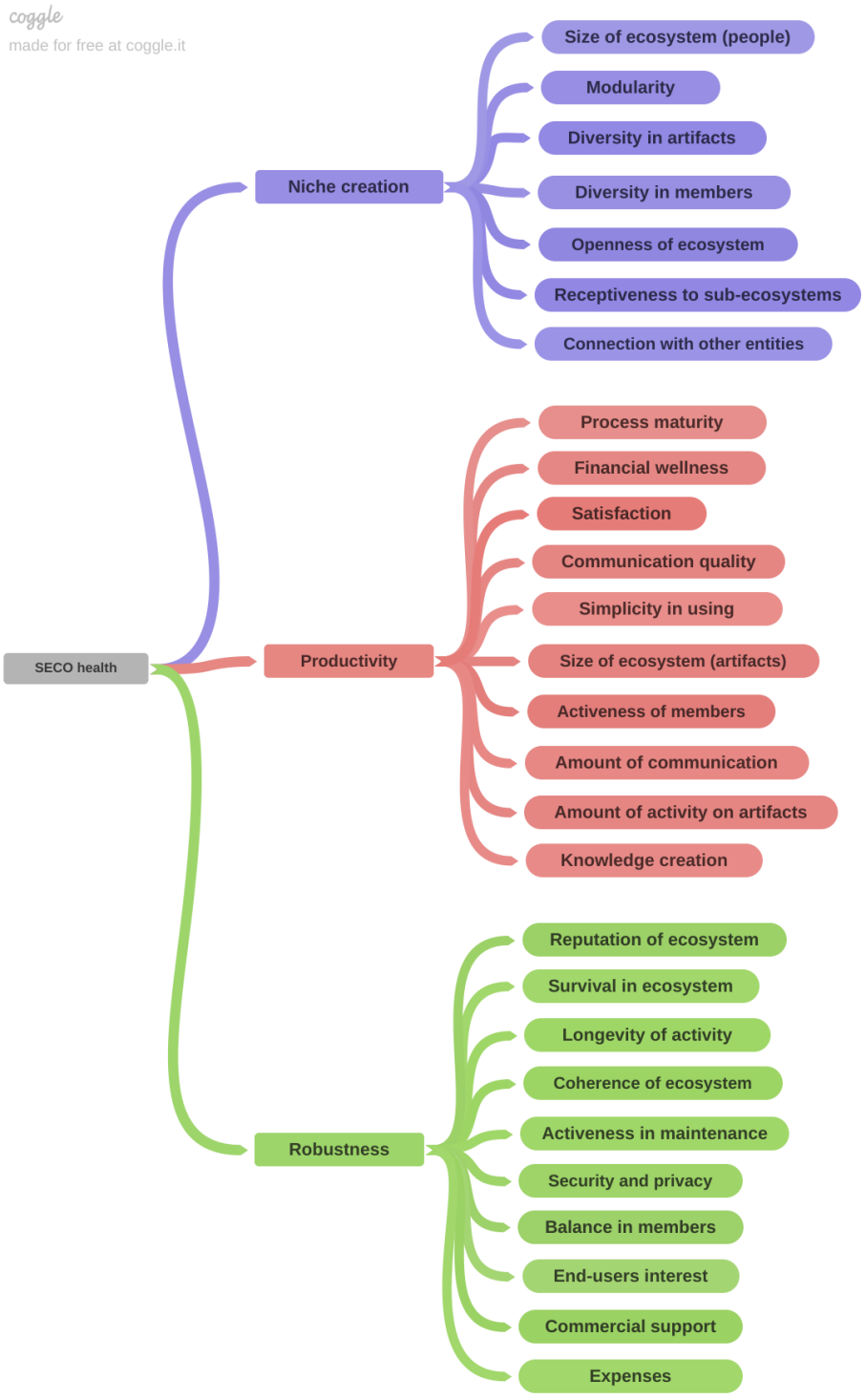

Figure 5.1: Categories and sub-categories of the software ecosystem health taxonomy

as users, contributors, followers, etc. More members may indicate that the ecosystem's community has a better structure for maintaining its products [45]. Among the metrics in this sub-category, *number of followers* was applied to measure the health of cloud PaaS providers [43], *number of unique developers* was applied to evaluate e-commerce ecosystems [2], and *number of registered users* was applied to assess the health of open-source SECOs [69].

Modularity. The two metrics in this sub-category measure a product's decomposition into sub-assemblies and parts. This division promotes element standardization and increases the diversity of products, but they can also be an indication of how dimensions in the subsystem can be reused. As companies strive to rationalize their product lines and provide increasing product diversity at a lower cost, attention has been paid to the concept of modularity [28]. The two metrics in this sub-category—*the number of modules shared and reused by partners* and *the number of modules developed by partners*—were applied to assess the health of SECOs [62].

Diversity in artifacts. Nine metrics comprise this sub-category, and they measure the diversity of the artifacts in a SECO, taking into account the technology, programming languages, supporting hardware devices, applications, etc. A large variety of artifacts is an indicator that there are many niches, platforms, domains, etc., in which a new player can become active [37]. Out of the metrics in this sub-category, *number of unique programming languages* was applied to measure the health of cloud PaaS providers [43], *variety in ecosystem projects technologies* was used to evaluate the SECO health of cryptocurrencies [5], *number of open source code categories* was applied to assess the health of open-source SECOs [69], and *variety in supporting hardware devices* was applied for SECO health evaluation [12].

Diversity in members. The five metrics in this sub-category measure the diversity of members in a SECO, taking into account their geographic locations, natural languages, activity types, organizations where they are affiliated, etc. More diversity in members leads to an increased capability to create meaningful variety over time by creating new valuable functions [35]. Although other aspects of diversity have been considered in this sub-category's metrics, diversity in gender and culture has not been considered. This gap can be filled in future research. Out of the metrics in this sub-category, from the literature I reviewed, only *variety in developer type* was applied for the health measurement of data-scarce SECOs and specifically Apple's ResearchKit [68].

Openness of ecosystem. The two metrics in this sub-category measure how

open a SECO is for users and developers to contribute freely. Contribution in a SECO can be made in different ways. For instance, developers can send a pull request to submit their code changes. *Open source code usage* as a metric in this sub-category measures what percentage of these submissions are successfully applied to the software in a SECO. Both of the metrics in this sub-category—*open source code usage* and *openness of ecosystem for users to freely contribute*—were applied to assess the health of open-source SECOs [69].

Receptiveness to sub-ecosystems. The four metrics in this sub-category measure to what extent a super-ecosystem is receptive to sub-ecosystems. An example of a sub-ecosystem is the ecosystem around the Google Assistant application, inside the larger Android "super-ecosystem". All Google Assistant applications have to be Android applications. However, not all Android applications use Google Assistant. Therefore, the Google Assistant ecosystem is a subset, or a sub-ecosystem, of the whole Android ecosystem. These metrics also evaluate how well sub-ecosystems can grow inside a super-ecosystem. Larger sub-ecosystems positively impact super-ecosystems' health as they are likely to introduce more external users into the super-ecosystem [51]. All of the metrics in this sub-category—*number of new sub-ecosystems*, *average size of sub-ecosystems*, *number of active sub-ecosystems*, and *variety in sub-ecosystems*—were applied to evaluate the health of sub-ecosystems [51].

Connection with other entities. These four metrics measure how much a SECO is connected with other entities. Entities can be ecosystems, companies, institutions, research communities, etc. In the case of changes to the environment and other disruptions, the SECO's connections with other entities affect the ecosystem's ability to survive and absorb shock [68]. Among the metrics in this sub-category, *proportion of subsystems in the system solved by third parties* was applied to assess the health of open-source SECOs [69], *outbound links to other ecosystems* was applied for health measurement of data-scarce SECOs and specifically Apple's ResearchKit [68], and *number of intersecting sub-ecosystems* was applied to evaluate the health of sub-ecosystems [51].

5.2 Productivity

Metrics in this category measure productivity, which is the efficiency in which an ecosystem converts inputs into outputs [3, 35]. Table 5.2 presents the sub-categories

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Size of ecosystem (People)	Number of contributors	How many people are contributing to different types of activities in the SECO community?	More is better. The number of active and mature contributors is a measure that indicates a healthy SECO community [37].	[27]
Modularity	Number of modules developed by partners	What is the number of modules developed by partners?	More is better. A higher number of modules developed by partners shows more modularity of the platform [62]	[62]
Diversity in artifacts	Number of context types of SECO project applications	Do the SECO projects have different applications in different contexts?	More is better. A wide variety of SECO project applications contexts will be more supportive of niche creation. [37]	[27] [37]
Diversity in members	Geographical distribution of members	Are the members of the SECO community geographically distributed?	More is better. A wider distribution implies more heterogeneity. [27]	[27]
Openness of ecosystem	Open-source code usage	What percent of submissions by developers are successfully applied to the software?	More is better. [69]	[69]
Receptiveness to sub-ecosystems	Number of active sub-ecosystems	What is the number of active sub-ecosystems?	More is better. A higher number indicates more receptiveness of the ecosystem to new sub-ecosystems.	[51]
Connection with other entities	Outbound links to other ecosystems	What other SECOs are the contributors active in?	The multi-homing activities of developers may or may not be beneficial for the robustness of the SECO [68].	[68]

Table 5.1: Some of the metrics for the niche creation category—a single metric is used as an example for each of the sub-categories.

in this category and includes example attributes from a single metric for each sub-category. Figure 5.3 shows all sub-categories and all metrics in the productivity category.

Process maturity. Three metrics in this sub-category measure process maturity in a SECO. Process maturity can be evaluated based on different criteria, such as democratic decision-making, the existence of brainstorming in the development process, the existence of review and testing before submission, etc. Two metrics in this sub-category—*process maturity based on developers opinion* and *democratic decision making*—were applied to measure the health of open-source SECOs [69]. *Reviewing and testing submissions* was used to assess mobile OS-centric ecosystems [11].

Financial wellness. These four metrics evaluate the financial wellness of a SECO, taking into account the growth of ecosystem profits, market share, percentage of developers' share from revenue, etc. Out of these metrics, *market share* was applied to measure the health of open-source content management systems [67], open-source

Figure 5.2: Sub-categories and metrics in the niche creation category within the software ecosystem health taxonomy

SECOs [69], and cryptocurrency ecosystems [10]. In addition, *developer revenue share* was applied to assess mobile OS-centric ecosystems [11].

Satisfaction. These five metrics measure members' satisfaction in a SECO, taking into account factors such as customer complaints, user ratings, employee satisfaction rate, etc. A high level of developer satisfaction binds developers to the ecosystem. Therefore, contributor satisfaction is an indicator of the robustness of an ecosystem [68]. Also, customer and user satisfaction can be seen as an indicator of the productivity of the ecosystem. Out of the metrics in this sub-category, only

contributor satisfaction was applied to evaluate data-scarce SECOs and specifically Apple's ResearchKit [68].

Ease of use. These three metrics measure how simple and easy it is to use or develop a SECO, taking into account factors such as the existence of documentation for the software platform, etc. Out of these metrics, *glossary of terms* and *documentation of the platform* was applied for SECOs health assessment [12], and *ease of use* was applied to evaluate the health of open-source SECOs [69].

Size of ecosystem (artifacts). These seven metrics measure how many artifacts there are in a SECO. Artifacts include application program interfaces (APIs), forks, repositories, apps, etc. An active community creates many artifacts, so it is possible to use these metrics to assess the activity level in an ecosystem [27]. Also, APIs make it possible to connect software within or even outside an ecosystem, enabling better communication between clients and the ecosystem [5]. Out of the metrics in this sub-category, *number of total repositories* and *number of unique repositories* were applied to measure the health of cloud PaaS providers [43], *number of commits to the software framework* was applied to evaluate data-scarce SECOs and specifically Apple's ResearchKit [68], *number of APIs for the ecosystem* was applied to measure the SECO health of cryptocurrencies [5], and *number of available apps* was applied to evaluate mobile OS-centric ecosystems [11].

Activeness of members. These eight metrics measure how active members of a SECO are, considering factors like the number of members making new features requests, the amount of time developers are willing and able to contribute to the development effort, etc. The number of active developers shows how dependent an ecosystem is on individual developers. A high number of active developers is the best defense for an ecosystem to survive massive changes, so a higher number of active developers shows that the ecosystem is relatively more robust [5, 37]. I believe that members' activeness impacts how much an ecosystem can convert inputs to outputs, so it is also an indicator of productivity. Out of the metrics in this sub-category, *active developers of unique repositories in the past year* and *active developers per segment of time* were applied to assess the health of cloud PaaS providers, *active contributors* was applied for the evaluation of cryptocurrency ecosystems [10], *number of users log in at least once a week* was applied to evaluate open-source SECOs [69], *users with at least one product actively running* was applied for measuring the health of antivirus ecosystems [42], and *number of active developers* was applied to assess the SECO health of cryptocurrencies [5].

Communication quality. These two metrics evaluate the quality of communications between members of a SECO, considering factors such as the positivity of vocabulary in communications, the existence of a common language, etc. These metrics are essential for community managers to consider, but they may be quite challenging to measure. This may be why no studies have applied them to measure health in practice so far.

Amount of communication. These seven metrics measure to what extent members of a SECO are active in communicating with each other, considering factors like the number of messages per day, number of questions with the tag of the ecosystem attached in Stack Overflow, etc. When developers can ask questions on knowledge bases (e.g., Stack Overflow), the ecosystem will benefit from the fact that it has a community where people help each other [5]. Out of the metrics in this sub-category, *level of contribution per community user* was applied to evaluate the SECO health of open-source content management systems [67], and *number of questions with tag of the ecosystem attached in Stack Overflow* was used to measure the SECO health of cryptocurrencies [5].

Amount of activity on artifacts. These fifteen metrics measure how active artifacts are in a SECO, considering factors like added KLOC in the last 30 days, the number of files changed per day, etc. The number of active artifacts measures an ecosystem's robustness because it shows what artifacts are being updated as the ecosystem changes. The number of new artifacts measures an ecosystem's productivity because it indicates the growth rate over time [24]. Out of the metrics in this sub-category, *up-to-datedness of modules* was applied to evaluate the SECO health of open-source content management systems [67], *number of commits per day*, *files changed per day*, and *average files changed or added or removed per commit* were applied for assessing open-source communities' health [55], *percentage of actively running products of the ecosystem* was applied for measuring the health of antivirus ecosystems [42], *number of active projects* was applied to evaluate data-scarce SECOs and open-source software companies [24, 68], *number of new projects* was applied for evaluating the SECO health of cryptocurrencies [5], *number of new apps* was applied to assess mobile OS-centric ecosystems, and *added KLOC in the last 30 days* was applied to evaluate SECO health [12].

Knowledge creation. These three metrics measure the amount of knowledge created by the SECO members, considering factors like the number of scientific publications generated by the community. Some other potential metrics for this sub-

category, like the number of blog posts and tutorials about the ecosystem, are missing in the literature and can be considered in future research. In this sub-category, only *scientific publications generated by the community* was applied to evaluate the health of SECOs [12].

Related sub-category	**Name of metric**	**Definition of metric**	**Interpretation of metric**	**Study**
Process maturity	Reviewing and testing submissions	Are submissions reviewed and tested?	A yes answer is better [11].	[11]
Financial wellness	Profit of the ecosystem	What is the expected profit of the ecosystem?	More is better. More profit indicates a more robust SECO [50].	[50]
Satisfaction	Total number of customer complaints	What is the total number of customer complaints?	Fewer is better. A lower number of customer complaints indicates higher stakeholder satisfaction [50].	[50]
Communication quality	Positiveness of vocabulary in communications	How positive is the message vocabulary content in the SECO?	A more positive vocabulary is better [27].	[27]
Ease of use	Glossary of terms	Does the platform have a glossary of terms to be used in the projects?	Not provided	[12]
Size of ecosystem (Artifacts)	Number of unique repositories	What is the number of unique repositories in the SECO?	More is better. A higher number of unique repositories shows more spin-offs [43].	[43]
Activeness of members	Number of active developers	What is the number of active developers in the SECO?	The number of active developers shows how dependent a SECO is on individual developers. A higher number of active developers indicates that the SECO is relatively more robust [5].	[5]
Amount of communication	Volume of messages per day	How many messages are sent per day?	More is better [55].	[55]
Activeness of artifacts	Number of major releases per year	How is the SECO community evolving?	More is better. The number of released projects shows the evolution in the SECO community [27].	[27]
Knowledge creation	Number of patents	How many patents does the community have?	More is better. A higher number shows more visibility for the SECO [37].	[27]

Table 5.2: Some of the metrics for the productivity category—a single metric is used as an example for each of the sub-categories.

5.3 Robustness

Metrics in this category measure robustness, an ecosystem's ability to survive disruption, such as technological and market changes [3, 35]. Table 5.3 presents the sub-categories associated with this category and includes example attributes from a single metric for each sub-category. Figure 5.4 shows all sub-categories and all metrics

Figure 5.3: Sub-categories and metrics in the productivity category within the software ecosystem health taxonomy

in the robustness category.

Reputation of ecosystem. The five metrics in this sub-category evaluate a SECO's reputation, taking into account factors like the number of web pages referencing the ecosystem's website. These metrics can be used by stakeholders who need to decide if they should participate in the ecosystem; for example, high contributor ranking implies more trustworthiness [37]. From the literature I reviewed, only one metric in this sub-category, *findability of the ecosystem*, was applied to evaluate open-source content management systems [67].

Survival in ecosystem. These five metrics measure how possible it is for members of a SECO to survive in and not leave the ecosystem, considering factors like the number of members leaving the ecosystem per month, etc. These metrics may be particularly important for managers to assess the retention of newcomers in the ecosystem. Out of the metrics in this sub-category, only *leavers per month* was applied to measure the health of e-commerce ecosystems [2].

Longevity of activity. These six metrics evaluate to what extent members' activities in a SECO are continuous, considering factors such as time interval between each developer's first and last contributions. These metrics indicate how active a community is and how it regenerates. A community that regenerates can be considered more sustainable since it is more likely to survive the loss of essential contributors over time [63]. Out of the metrics in this sub-category, only *number of repositories updated at least once* was applied to evaluate the health of cloud PaaS providers [43].

Coherence of ecosystem. These five metrics measure how coherent a SECO is, considering factors like collaborations and connections between the ecosystem members. More links imply more inter-relatedness, which is an indicator of robustness [27]. Out of the metrics in this sub-category, only *the degree of cooperation between different development groups* was applied to assess the health of open-source SECOs [69].

Activeness in maintenance. These seven metrics evaluate how well-maintained the SECO is, considering factors like how quickly problems are reported and resolved in the ecosystem, and the average time to fix critical bugs. The feedback cycle represents the time for developers to provide and receive feedback on a problem. The shorter the time, the more developers actively maintaining the stability of the ecosystem, and the healthier the ecosystem is [69]. Out of the metrics in this sub-category, only *time for developers to provide feedback on problems (feedback cycle)* was applied to evaluate open-source SECOs [69].

Security and privacy. The only metric I found in the literature review that fits this sub-category is *ability to resist viruses*, which evaluates the security and privacy in a SECO. This metric was applied to measure the health of open-source SECOs [69]. Since security and privacy are important factors for all the stakeholders in an ecosystem [69], there is a gap in this area that perhaps needs to be filled by defining more useful metrics in this sub-category.

Balance in members. These seven metrics measure the extent to which there is a balance between members of a SECO, taking into account their amount of experience, activity, expertise, etc. Since measuring the metrics in this sub-category is relatively tricky, researchers have yet to apply them to measure the health of SECOs.

End-users' interest. These six metrics measure how interested the end-users are in a SECO, considering factors like end-user ratings of the ecosystem. The usage of an ecosystem and the ratings from end-users indicate how important an ecosystem is to its end-users and can be used as a predictor of robustness [68]. Out of the metrics in this sub-category, *number of page views of the official website(s)*, *number of end-users of the released applications*, and *end-user interest in search statistics* were applied to evaluate cryptocurrency ecosystems [10], *end-user rating* was applied to evaluate data-scarce SECOs and specifically Apple's ResearchKit [68], and *recent users in the last 30 days* was applied to the health measurement of SECOs [12].

Commercial support. These three metrics measure how successful a SECO is in attracting commercial support, considering factors like the amount of money the ecosystem receives from other companies. Commercial patronage, capital contributions, and donations are indicators of acceptance by commercial organizations, showing that the ecosystem will likely stick around [10]. Out of the metrics in this sub-category, *commercial patronage* and *capital contributions and donations* were applied to evaluate cryptocurrency ecosystems [10].

Expenses. These six metrics measure a SECO's expenses, considering a wide range of expenses from marketing costs to quality management costs. No studies to date have applied any of the metrics in this sub-category to measure the health of SECOs. This could be for various reasons; for example, data collection to measure these metrics may be too difficult, etc.

In this chapter, I describes the taxonomy I built for the metrics from the SLR. In the next chapter, I describe the taxonomy I built for the practices from the SLR. Besides, I explain all the categories and sub-categories of the taxonomy and the practices in each category and sub-category.

Figure 5.4: Sub-categories and metrics in the robustness category within the software ecosystem health taxonomy

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Reputation of ecosystem	Web and Social media hits	Do the SECO community projects have visibility on the web?	More is better. More web and social media hits indicate more visibility for the SECO [27].	[27]
Survival in ecosystem	Leavers per month	How many leavers does the SECO have per month?	Fewer is better [2].	[2]
Longevity of activity	First and last contributions of each developer	How great is the difference between each developer's first and last contributions over the project's history?	Less is better. Because a regenerating community can be considered more sustainable since it is more likely to survive the loss of important contributors over time [63].	[63]
Coherence of ecosystem	Number of nodes to disconnect the ecosystem	How is the SECO connected?	More is better. A higher number of nodes to disconnect the ecosystem indicates more ecosystem cohesion [27].	[27]
Activeness in maintenance	Frequency of the contributions to the bug report system	How frequent are the contributions to the bug report system?	More is better [63]	[63]
Security and privacy	Ability to resist viruses	How great is the SECO's ability to resist viruses?	More is better. More ability to resist viruses indicates more product quality [69].	[69]
Balance in members	Developers' activity on release distribution	What is the distribution of developers' participation in the SECO community?	Lower is better. A lower value indicates a more uniform distribution.	[27]
End-users interest	Number of end-users of the released applications	What is the number of end-users of the released applications?	More is better. The usage of a project is also highly indicative of how important a project is to its end-users and can be used as a predictor of robustness [37].	[68] [10] [37]
Commercial support	Capital contributions and donations	How much support does the SECO receive from other companies?	More is better. Capital contributions and donations are indicators of acceptance by commercial organizations, showing that the project will probably not go away soon. [68].	[37] [10]
Expenses	Internal quality control cost	What is the cost of internal quality control in a SECO?	Less is better. Less internal quality control cost indicates higher productivity [50].	[50]

Table 5.3: Some of the metrics for the robustness category—a single metric is used as an example for each of the sub-categories.

Chapter 6

A Taxonomy of Software Ecosystem Health Practices

In this chapter, I describe the taxonomy I created for software ecosystem health practices. The taxonomy consists of three interconnected components in a hierarchical structure: categories, sub-categories, and practices. Most of my primary studies were unanimous on the three high-level categories of practices: *Productivity, Robustness, and Niche creation*. So, at the highest level, my taxonomy has also these three top-level categories. At the middle level, several sub-categories are assigned to each category. At the lowest level of the taxonomy, the practices clustered in each sub-category are displayed.

In the rest of this chapter, I describe the categories and sub-categories in the taxonomy. Figure 6.1 shows an overview of the categories and sub-categories in the taxonomy. As mentioned in chapter 4, all the practices and their related attributes collected through my systematic literature review are available in detail at `https://github.com/Armanyousefzade/Software-ecosystem-health-practices`.

6.1 Niche creation

Practices that I have grouped in the niche creation category are practices that impact an ecosystem's ability to produce value by increasing diversity [3, 35]. Table 6.1 presents the sub-categories in this category and includes example attributes from a single practice for each sub-category. Figure 6.2 shows all sub-categories and all practices in the niche creation category.

Translate artifacts to improve niche creation. When an ecosystem is better at supporting translation tasks, the ecosystem can provide a better environment for diverse languages, and as a result, its niche creation ability could be increased. These practices cover different translation tasks supported by a software ecosystem, such as translation of documents, tutorials, website content, social media posts, emails, etc. When the content in a software ecosystem is translated into different languages, more people can contribute, which increases the diversity of the ecosystem. The six practices that are recommended by the literature I reviewed that relate to this sub-category of practices are:

- Provide information about the translation process (i.e., guide ecosystem members on contributing to translation activities and explain which tools to use and how they should be used) [20].
- Provide tutorials to teach how to translate and how to become a contributor [16].
- Provide guidelines for creating an environment to work with translations and code development [16].
- Control the translation work and separates artifacts into two groups for translation: stable artifacts that have experienced no or few changes, and unstable artifacts that are continually changing and need to be checked [16].
- Provide specific tools to optimize translation work. For example, a tool called Localize[1] searches text to translate. It also suggests words to use in a translation [16].
- Provide a freeze period to stabilize the translation before the launch of a version [16].

Keep the Schedule of members flexible to improve niche creation. When an ecosystem provides a flexible schedules for its members, various contributors with different schedules are able to participate in and contribute to the ecosystem. For example, it is easier for some people to work in the mornings, whereas others may concentrate better on their tasks in the afternoons or evenings. Some people are more comfortable working on weekends, but others do not work at all on weekends. The two practices suggested by the literature that fall in this sub-category are:

[1] `https://localizejs.com/`

- Provide flexibility in the time to work to help support better code quality [20].
- Provide flexibility in the translation schedule and negotiate deadlines [16].

Promote social relationships with members in other countries to improve niche creation. An ecosystem can be connected to different countries, bringing diversity to the ecosystem and improving niche creation. For example, suppose an ecosystem has members from various countries. In this case, the diversity in the ecosystem will be increased as members from different countries can bring diverse values to the ecosystem and provide a welcoming atmosphere and environment for different cultures, languages, and ideas. Two practices that were recommended in the literature are grouped in this sub-category:

- Promote social relationships with members in other countries. This can be achieved by different mechanisms such as annual meetings for all the members to meet each other in person, or by sending gifts to people in other countries based on their interests and culture [20].
- Create an opportunity to practice another natural language such as English, French, etc., to make contributions easier for people from other countries [16].

6.2 Productivity

The practices I found in the literature that I grouped according to this category, are practices that are claimed to improve productivity. By productivity, we refer to the efficiency in which an ecosystem converts inputs into outputs (as mentioned above) [3, 35]. Table 6.2 presents the sub-categories I assigned to this category and includes example attributes from a single practice for each sub-category. Figure 6.3 shows all sub-categories and all practices in the productivity category.

Use APIs to improve productivity. Using APIs helps an ecosystem utilize the services provided by other software ecosystems or projects, saving a lot of time and money from an ecosystem and improving its productivity in converting inputs to outputs. If an ecosystem does not use APIs, it should spend a lot of money and time building their own version of these services. In contrast, by using APIs provided by other ecosystems, the ecosystem has access to a wide range of services and data by spending much less time, money, and energy and with better quality. The three practices I grouped into this sub-category are:

Related sub-category	Name of practice	Key area	Source
Translate artifacts to improve niche creation	Provide specific tools to optimize translation work. For example, a tool called Localize[2] searches text to translate. It also suggests words to use in a translation.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Keep the Schedule of members flexible to improve niche creation	Provide flexibility in the time to work to help support better code quality.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, making decisions, and so on.)	[16]
Promote social relationships with members in other countries to improve niche creation	social relationships with members in other countries. This can be achieved by different mechanisms such as annual meetings for all the members to meet each other in person, or by sending gifts to people in other countries based on their interests and culture.	Social (related to working together in the community)	[16]

Table 6.1: Some of the practices as examples for the niche creation category

- Use Applications Programming Interfaces (APIs) to consume software services provided by other ecosystems and business parties [18].
- Create and Stabilize APIs to facilitate integration with other applications and ecosystems [72].
- Document APIs to share knowledge about using and consuming them with other developers in the software ecosystem [14].

Automate tasks to improve productivity. A lot of development tasks crucial to an ecosystem can be done automatically, such as optimizing code, performing static analysis of code, testing code, etc. When an ecosystem does not have support to automate tasks, members may spend a lot of unnecessary time and energy manually doing things. The five practices I found in the literature I reviewed and that I grouped into this sub-category are:

- Provide an automated process to support continuous software engineering [61] and software system releases [20].
- Provide tools for code optimization, static analysis of code, code review, and test automation [16].
- Provide continuous integration tools that check code every day and report errors [16].
- Generate code documentation automatically [16].
- Use automatized tests to gather problems, bugs, and errors in the code [14].

Plan releases to improve productivity. An ecosystem can have various plans for different tasks and releases. This can help the ecosystem be organized as every task is done according to a specific schedule to enhance productivity. Planning helps prioritize tasks so people focus on what's important; ecosystem members do not spend their time and energy on tasks that do not have a high priority. Having accurate plans help an ecosystem reach its goals and priorities. Without them, ecosystems can move in directions that conflict with its goals and priorities. As a result, a software ecosystem should continuously make new plans and evolve its existing plans based on its problems, priorities, and goals. The three practices I grouped into this sub-category are:

- Define a schedule of releases that will affect the work of the entire community [16].
- Provide a feature plan that will be used for the implementation of new features [16].
- Plan releases according to a timeline or feature road map [63].

Resolve financial and marketing issues to improve productivity. A software ecosystem with different types of expenses needs to be in a good financial situation to afford its expenses. Besides, to make software products with high quality, a software ecosystem should do marketing to make its products known by its target communities. This can increase the financial income of the ecosystem and also improve its financial situation. Also, ecosystems with sound financial and marketing plans tend to be more successful and can attract better people. The nine practices that were discussed by the literature and I grouped into this sub-category are:

- Create groups to lead and react to financial and business issues and changes [18].
- Provide a nonprofit corporation to manage legal and financial issues that can sometimes make a lot of trouble for ecosystems [16].
- Define and support a particular group to manage all business issues [16].
- Market the ecosystem to attract customers and end-users for software products of the ecosystem [72].
- Do collaborative marketing to increase the market share of the ecosystem among its competitors [72].
- Create sales partner programs and new sales channels to increase the revenue of the ecosystem [72].
- Attract companies that invest money to support the ecosystem financially [16].
- Promote networking in the work market to improve market share [16].
- Establish a financial board to manage financial resources. The software developers do not have enough experience to manage financial resources, so ecosystems need such a board to manage their financial resources [14].

Avoid misunderstanding to improve productivity. When misunderstandings happen in an ecosystem, productivity can be negatively affected as resources are wasted in resolving issues. As a result, having agreements that avoid misunderstandings can save a lot of time, energy, and effort, and increase productivity. Suppose the ecosystem does not have some agreements to avoid misunderstanding. In that case, different types of misunderstandings can damage trust between members of the ecosystem, which can have an unfixable effect on an ecosystem's reputation. Every ecosystem should have some plans to avoid misunderstandings, especially between its members and end-users. The three practices I grouped into this sub-category are:

- Create a code of conduct to avoid mistreatment among members [20].
- Reach an agreement in the community to answer questions that are not registered in the tutorials or guidelines [16].

- Support use of a common written and spoken language to avoid misunderstandings [14].

Provide physical and digital infrastructure to improve productivity. When an ecosystem provides physical and digital infrastructure, people can contribute easier and with more efficiency. If tasks are easier to complete, more outputs will be produced given a certain amount of time, and as a result, productivity will be enhanced. If an ecosystem does not provide physical and digital infrastructure for its contributors and developers, it will be slow and inefficient in making software and resolving issues. As a result, the ecosystem will have much less productivity because it produces much less output and products. The ecosystem management board's wise decision is to provide all of the necessary infrastructures for its members to make them more productive. For sure, the spent expenses on providing this infrastructure will be returned to the ecosystem. The eight practices I grouped into this sub-category are:

- Create personal blogs and wikis to inform people about important development and architectural issues [14].
- Provide hardware and software resources to be used by the community [14].
- Provide resources to test code that needs special hardware, operational systems, or network connections. This happens because developers do not always have access to all the infrastructure needed to test code [16].
- Use of repository hosting or version control services to maintain revision and version history [18].
- Use technology to reuse GUI components to facilitate development work [16].
- Support the ecosystem's participation on aggregate sites such as Stack Overflow, Reddit, Hacker News, and so on [20].
- Use of websites, wikis, tutorials and wizards to share knowledge in the ecosystem [18].
- Provide a virtual machine with preconfigured build environments, web-based IDEs, or a content management tool [20].

Use tools to improve productivity. Tools can help people complete more tasks with less time and effort. Software ecosystems are no exception to this rule, and supporting the use of different tools can enhance the productivity of the ecosystem. A wide range of tools have been built for creating software. These tools have significantly increased software development speed and made software development much more efficient and productive by reducing software developers' obstacles in their work. These tools help developers in a wide range of tasks, like software documentation, requirements collection, software debugging, programming, etc. The selection of suitable tools is critical for an ecosystem. The ecosystem's management board should ensure that the members use suitable tools based on the ecosystem's priorities and goals. Two practices I grouped into this sub-category are:

- Use of tools to register requirements [18].
- Use tools to compute quality metrics [14].

Do Articulation work to improve productivity. Every community needs to be managed to reach its goals. As complicated communities of people and software tasks, software ecosystems need to be managed well to be productive and reach their predefined goals. To reach a balance, tasks should be fairly divided between the members of the ecosystem. If a small group of people in an ecosystem produce most of the ecosystem's output, there is no balance between members. In this case, if this small group of members leaves the ecosystem for some reason, the ecosystem's productivity can decrease significantly and suddenly, which can be unfixable in some cases and even lead to the destruction of the ecosystem. Four practices I grouped into this sub-category are:

- Tag all tasks by the degree of difficulty (easy, medium, difficult) to help developers select tasks based on their skills and experience [20].
- Keep the list of tasks updated to inform people who are working on a solution [20].
- Each person chooses the work they would like to complete among available tasks in the ecosystem. For example, translate files, develop code, test applications, and so on. This will help the members work on tasks that interest them [16].
- Provide checklists of tasks required to complete the work. For example, before adding code to the repository, the developer needs to document, test, integrate

with other applications, have the code reviewed by another person, and confirm any translations. This will help the developers remember important and necessary tasks [16].

Train members to improve productivity. In every group, there are different levels of experience, knowledge, and skills. So training people to bring them to an acceptable level of skill and knowledge is essential for an ecosystem's productivity. Suppose the senior developers do not spend enough time and energy on training junior developers. In this case, the junior developers will never have a chance to show their potential and increase the ecosystem's productivity. But on the other hand, if the junior developers are trained well by more experienced developers, they can significantly increase the ecosystem's productivity by creating more software products for the ecosystem. The five practices I grouped into this sub-category are:

- Provide tutorials and video courses for the technologies used by the ecosystem to help less experienced members improve their knowledge and skills on these technologies [14].
- Provide instruction about introducing the ecosystem, installing technologies, configuring the development environment, and using dependencies with other ecosystems to help junior developers enhance their skills [20].
- Provide mentoring programs to give new developers the skills and knowledge they need to be successful in the ecosystem [14].
- Encourage different architects with different specialty levels to share their knowledge with other members to make sure everyone has the minimum required skills and knowledge [14].
- Teach experienced members how to use tools to be more productive. For example, the use of shortcut keys, scripts, and plugins. These help experienced members use the tools in more efficient ways [16].

6.3 Robustness

The practices in this category improve the robustness of an ecosystem, which is an ecosystem's ability to survive disruption such as technological and market changes

[3,35]. Table 6.3 presents the sub-categories associated with this category and includes example attributes from a single practice for each sub-category. Figures 6.4 and 6.5 show all sub-categories and all practices in the robustness category.

Maintain backward compatibility to improve robustness. The world of software technologies is changing rapidly with a continual stream of new versions of tools and programs. An essential feature for each software project or ecosystem is remaining compatible to survive these considerable technology changes. An ecosystem should always update its products and make them better, but at the same time, it should also maintain compatibility with previous versions. In this case, end-users of previous versions that do not update their products continuously can keep using their products without trouble. On the other hand, if the ecosystem does not maintain compatibility across their products, users will experience difficulties when products are updated and new versions are released. As a result, people may stop using the products, which can harm the ecosystem's robustness. Four practices I grouped into this sub-category are:

- Assign deciding about removing obsolete code to architects' leaders, respecting backward compatibility. This will make sure correct decisions are made about removing obsolete code while supporting backward compatibility [14].
- Execute the code with several different versions of operating systems and compilers to maintain compatibility and avoid errors [16].
- Provide guidelines about actions that are allowed and not allowed to keep backward compatibility [16].
- Keep backward compatibility for a defined period to allow the community to update its software [14].

Manage the responsibilities to improve robustness. In software ecosystems, there is a wide range of responsibilities and tasks which need to be assigned to the members in a way that makes sense for everyone and is based on the experience, skills, and knowledge of the people involved. Managing these responsibilities is very important for the robustness of ecosystems. If they are not managed well, it can result in unhappy members that leave and damage the ecosystem's reputation. In assigning responsibilities, a lot of factors should be considered by managers. For example, some members are only interested in marketing the software products, so

the managers should not assign development tasks to these people, instead assigning development tasks to members who are actually interested and skilled in software development. Four practices I grouped into this sub-category are:

- Create confidence in members based on their work history and provide different levels of responsibilities for members based on their level of experience and skills [16].
- Hold elections to democratically assign responsibilities to members to ensure a democratic procedure is taken to assign responsibilities to members [16].
- Define a hierarchy for members with different levels of responsibilities to organize vertical responsibilities structure [16].
- Allocation of responsibilities by the developer/architect's interests and discussing it with the other members to make sure that developers and architects be assigned to duties they are interested in [14].

Manage requirements for applications to improve robustness. Software applications can have differing requirements. Managing this wide range of requirements is challenging because it can create conflicts that threaten the ecosystem's robustness and stability. If requirements are not managed well in an ecosystem, too many unnecessary requirements can be created, making development, maintaining, and using the software difficult. The two practices I grouped into this sub-category are:

- Create requirements for applications by developers and end-users to ensure that requirements are based on both developers' and end-users' perspectives [16].
- Provide documentation explaining the requirements of applications belonging to the ecosystem to ensure that requirements are clear for all members [20].

Apply continuous improvement to enhance robustness. An ecosystem should always learn from its failures and successes, and improve its procedures and mechanisms. Continuous improvement increases the power of the ecosystem to overcome change. Failures, mistakes, and wrong procedures happen in every ecosystem, but it is important that the ecosystem learns lessons from its mistakes and improves its mechanisms to avoid future issues. An ecosystem that always tries to improve itself will be more robust and reduce its mistakes and failures. This sub-category's three practices are:

- Use lessons learned by past mistakes to modify and improve practices and make sure that past mistakes will not be repeated [16].
- Create a process for proposing significant enhancements to ensure that community members' ideas are heard. For example, some projects have disciplined processes that allow community members to propose improvements for formal discussion by the community [63].
- Conduct and control refactoring in the project as necessary to make sure that projects are improved and refactored when it is necessary [16].

Update and release new versions to improve robustness. As mentioned earlier, the world of software technology is continuously changing, so a software ecosystem needs to remain current to survive. In addition to software technology, the user's requirements also change continuously, and software ecosystems need to react to these changes and create a product that responds to users' evolving needs. Otherwise, if software ecosystems do not update their products and release new versions, they can lose their market share. Their users will choose another ecosystem that updates their products more frequently and is more responsive to their needs. As a result, it is important for the robustness of the ecosystem to keep updated frequently. The two practices I grouped into this sub-category are:

- Identify and dismiss outdated information on websites to ensure that the websites' information and content are updated [20].
- Mark code as deprecated in features removed from the software to ensure that only the correct code exists [16].

Promote the community collaboration to improve robustness. An important factor for software ecosystems to survive big changes is how connected their members are. Holding meetings and events allow the members of an ecosystem to connect and share their knowledge and experience, which improves the balance and robustness in an ecosystem. Suppose an ecosystem has a strong community and its members collaborate actively with each other and share their ideas. In this case, the ecosystem can be much more responsive to changes, can react faster in emergencies and, as a result, be more robust in facing technological shocks and change. Conversely, suppose an ecosystem's members do not collaborate frequently. In this case, they can not make important decisions fast enough to react to and survive big changes. The ten practices I grouped into this sub-category are:

- Provide active members with trips to events sponsored by the ecosystem to help them build their relationships with other community members [16].
- Conduct an annual general meeting to discuss face-to-face questions from the community [16].
- Conduct annual meetings among translators of projects to improve the collaboration of translators in the ecosystem[16].
- Promote happy hours and dinners to facilitate social integration during events [16].
- Provide video conference sessions with questions and answers about relevant topics to improve the communication of ideas in the ecosystem [20].
- Hold online meetings in a time zone convenient for most community members to make sure people from different time zones can collaborate [14].
- Provide online meetings to discuss architectural problems and make sure that members of the ecosystem have a chance to collaborate on fixing architectural problems [14].
- Provide financial resources to support face-to-face meetings [14].
- Provide meetings (sprints) face to face to accelerate the development of critical issues [14].
- Provide specific face-to-face meetings with teams for different applications to solve development problems with their interdependent modules [14].

Welcome newcomers to the ecosystem for improving robustness. The new members of an ecosystem should be welcomed into a pleasant environment when they join. A positive experience can stick with a new member for a long time. On the other hand, a negative experience can make a new member leave the ecosystem quickly, which is a bad sign for its robustness. As new members bring new skills and diversity to an ecosystem, welcoming them is important for retention. The members of ecosystems usually do not forget their first weeks after they join, and their first impressions affect most of their future decisions in the ecosystem. The eleven practices I grouped into this sub-category are:

- Inform newcomers about the required technical background. Identify the specific technologies they need to know or learn to contribute to the ecosystem [20].
- Point newcomers to easy tasks filtered by difficulty, skills needed, and topics [20].
- Provide a newcomer-specific page or portal guiding their first steps, including architectural information [20].
- Provide a generation of (semi)automated documentation filtered by up-to-date information relevant to newcomers [20].
- Establish mentoring by designating a few experienced members to communicate with new members [20].
- Request that newcomers write informative subjective lines when desiring an answer from the community. They should post questions that are direct and to the point [20].
- Provide a dictionary to newcomers to facilitate their learning of the technical jargon and acronyms of the community [20].
- Provide a message template for newcomers to use to interact with the community [20].
- Provide clear directions to enable newcomers to find the artifacts that need to be changed [20].
- Provide a development manual for newcomers to know how to start contributing [14].
- Lower entry barriers to make it easier for different people to join the ecosystem [72].

Manage mailing lists to improve robustness. Mailing lists provide an environment for the ecosystem's members to help each other and share their knowledge quickly and easily. If members can share their insights, they can better overcome conflict and change. Mailing lists are not the only tools for the ecosystem's members to collaborate and share their ideas. Still, because of their unique advantages, they

have been vastly used by different ecosystems. The most important feature they provide in collaboration is safety, speed, transparency, and simplicity. The five practices I grouped into this sub-category are:

- Use of mailing lists to share ideas, discuss problems, and make decisions [18].
- Answer questions on the mailing list right away to make sure that questions and issues are solved quickly [20].
- Use tools to support communication in the group, such as mailing lists, forums, IRCs, wikis, blogs [16].
- Make decisions based on discussions in the mailing list to ensure that the ecosystem's decision-makers consider the ideas shared in the mailing list [16].
- Use discussion mailing lists divided by different topics, such as test, development, etc., to help members find relevant messages and discussions more easily [16].

Manage dependencies to improve robustness. Dependencies are a non-separable part of each software ecosystem. Many software products and projects are dependent on each other, and a problem in one project or product can cause a chain of troubles in other dependent projects. So managing these dependencies can avoid potential problems that may damage the robustness of an ecosystem. Sometimes the dependencies on other strong and robust software ecosystems make the software ecosystem more robust, but too many dependencies on weak ecosystems can threaten their robustness. As a result, dependencies can be considered opportunities or threats, and this is dependent on how well the dependencies are managed in the ecosystem. The five practices I grouped into this sub-category are:

- Provide information about dependencies among artifacts to make dependencies clear [20].
- Provide rules for dependencies to guide ecosystem members on how to work with dependencies [18].
- Use metadata to provide data about the dependencies of the system [18].
- Use scripts to evaluate improper dependencies [16].

- Divide the parts of the software into layers and define restrictions to manage dependencies among the layers [16].

Manage versions to improve robustness. As an ecosystem evolves and new technologies are used, new versions of the products and projects are released in the market. Managing all of these versions, which are sometimes released monthly or even weekly, takes a lot of time and effort but is important for an ecosystem's to reach its short-term and long-term goals. Ecosystems should have an accurate schedule for their version releases, and they should plan how their versions differ from each other and respond to the changing needs of their users. There are various methods for managing released versions, and the ecosystem's managing board should select suitable methods consistent with the ecosystem's goals and priorities. The five practices I grouped into this sub-category are:

- Release new versions of the product. Release processes in open source often include creating several alpha, beta, and release-candidate versions delivered by the developers to obtain feedback from the community. Active users of an OSS system are often willing to test these versions and report problems they find. Release processes also often include running a test suite or performing other forms of formal testing [63].

- Use agile practices, such as continuous delivery, building small versions, frequent releases, and daily meetings, to increase the efficiency of the ecosystem [18].

- Conduct incremental transitions rather than releasing the software product all at once. This helps build the next versions of software products based on the received feedback on previous versions [16].

- Develop two versions in parallel, an old and a new version, until the new version is stable and ready to replace the old version [16].

- Backport corrections in the current release to previous stable releases. When a stable and unstable development branch of a project is maintained simultaneously, so-called backports are often necessary to move corrections or selected improvements made to the development branch into the stable branch [63].

Create partnerships to improve robustness. When an ecosystem has strong partners, it can better survive change and shock from the outside world. Different

partners have various strengths, which an ecosystem can leverage. In the case of technological shocks and large changes, sometimes other strong parties can help an ecosystem survive by assisting in different areas and covering its weak points. The three practices I grouped into this sub-category are:

- Co-develop, i.e., joint develop projects with other companies to take advantage of their services and achievements [72].
- Form alliances to survive significant changes, shocks, and disruptions [72].
- Make partners explicit to show a powerful picture of the ecosystem to the outside world [72].

Design architecture to improve robustness. In every software ecosystem, architecture is the base of the software project. If the architecture is designed in a way that meets the ecosystem's goals, many future issues and challenges can be avoided, and a lot of time and energy can be saved. So investing time and energy in organizing software architecture in an ecosystem is a worthwhile investment. Well-designed software architecture can make software development and suitability much more efficient in later software development cycles. On the other hand, if enough time and energy has not been spent on designing a software ecosystem's architecture's, developers may need to spend a lot of their time fixing issues brought on by the poorly designed architecture. The six practices I grouped into this sub-category are:

- Recommend a tool to create UML diagrams of the architecture to make all the UML diagrams consistent [14].
- Remove code based on the architect's experience to make sure that code is removed correctly [18].
- Build the architecture based on plugins to facilitate the decoupling of applications [14].
- Provide autonomy to work on the architecture by the degree of commitment and experience of work in the community [14].
- Follow the core architects' recommendations in making decisions to ensure that core architects' ideas are considered in making important decisions [14].

- Make design decisions to guide developers when working on how system components communicate (between systems, between the system and external entities, between elements of the system) [14].

Resolve problems and bugs in the software to improve robustness. Problems, issues, bugs, and errors usually happen in software ecosystems. An important factor in each ecosystem is how the ecosystem handles these issues and fixes them. If an ecosystem has solid processes for resolving issues when they occur, they can be fixed in an optimized way that does not stop the ecosystem's development process. Also, the software ecosystem must detect and fix errors and bugs fast because the issues in a software product can cause many problems for its users and damage the software ecosystem's robustness, and potentially even cause the ecosystem to lose its market share. This sub-category's three practices are:

- Use a bug triage task force to remove repeated or non-existent bugs [16].
- Create a detailed step-by-step tutorial linking information about common problems and possible solutions to help the ecosystem members resolve issues and errors more easily [20].
- Report and handle issues with the product in the shortest amount of time possible to ensure that end-users do not suffer for long [63].

Inform changes and decisions to the ecosystem community for improving robustness. Decisions and changes are made continuously in big software ecosystems with large numbers of members, but transparency can help members stay aware of important decisions and changes which have a significant impact on them. If members be aware of important changes and decisions, they will trust the ecosystem more, resulting in a more robust ecosystem. On the other hand, when the decision-making process and the changes are not clear to members, they do not consider themselves part of the ecosystem—they are just like followers of a small group's decisions which can harm the robustness of the ecosystem. The nine practices I grouped into this sub-category are:

- Discuss critical changes that will impact applications to ensure the community is aware of these important updates [14].
- Keep meeting minutes available to the community to know all decisions made in the meeting [14].

- Discuss changes and announce them broadly to the community [18].
- Provide an accountability report to the community every three months [16].
- Announce features that each member will implement to avoid duplicated work [16].
- Communicate API changes in advance to make the community aware [16].
- Make strategy explicit for the community to make them aware of the important strategies followed in the ecosystem [72].
- Create a document for the community to inform them about the next steps [14].
- Publish architectural changes widely to make changes transparent to the community [14].

Document to improve robustness. Software ecosystems regularly have members departing and new people joining. When members leave an ecosystem, they share their knowledge with other members through the documents they leave behind. In these situations, documentation is essential so new members can easily obtain needed information and quickly come up to speed with established members. Sometimes the members of ecosystems forget the details about their previous decisions. Documentation helps developers and other software members easily remember all the details related to past decisions and actions that have been taken. Also, there are usually several teams of developers in software ecosystems who make different decisions and perform various actions that directly impact the whole ecosystem. If every team properly documents their decisions and actions, other teams can easily be notified. As a result, teams can make consistent decisions which will help the ecosystem reach their goals. The Four practices I grouped into this sub-category are:

- Keep documentation organized and indexed based on the level of difficulty, from beginners to experts, to help community members find documents based on their level of experience [20].
- Provide documentation about translation rules that developers must follow to prepare code for other languages [20].
- Provide documentation about the operation of projects, such as wikis, user guidelines, and meeting event logs, so that critical information is recorded for future reference [16].

- Build maps of modules and run-time elements during face-to-face meetings to document the modules' structure for future reference [14].

Standardize procedures to improve robustness. If every member follows their own methods to complete a task, the ecosystem will reach an unstable state after a short time, damaging its robustness. Software engineering includes a wide range of mechanisms and methods for software development. However, only a subset of these methods are standard and verified by experts and experienced software engineers and researchers. It is important for software ecosystems to standardize their methods and procedures. If an ecosystem uses standard methods and procedures, its products can support different software and hardware devices. It can also use the software services provided by third parties, like other ecosystems or APIs. Besides, following standard procedures have already been tested and discussed by a large community. A lot of information is available about standard procedures that can be used by software ecosystem members to learn about them. The seven practices I grouped into this sub-category are:

- Provide a design style for all application development [16].
- Define standard regulations and processes to be followed by community members [18].
- Specify specific tools that should be used by the community for testing, communication, code review, debugging, and navigation [14].
- Require that all infrastructure be under control of the ecosystem and be based on technologies created by the ecosystem [16].
- Define minimal quality criteria requirements, such as documentation, automatized tests, and dependence restrictions, to add an application to the ecosystem [14].
- Provide a manifest which the project must follow to be considered part of the ecosystem [16].
- Provide standard code recommendations and examples of code to be used by junior developers [18].

Assess code to improve robustness. Software ecosystems, especially those that follow agile methodologies, define peer review and testing procedures to maintain software product and project quality. These practices are important because they directly improve the software's quality, which allows the ecosystem to survive changes better and be more robust. In software ecosystems, developers have different levels of experience. Senior developers are experienced in a wide range of skills, whereas junior developers have limited skills and experience. Like peer code review, the code assessment process helps senior developers supervise junior developers and ensure junior developers develop the software with high quality. In other words, peer code review is an important method for training less experienced developers in software ecosystems. The eight practices I grouped into this sub-category are:

- Perform code inspections to find bugs, errors, and issues in the code [18].
- Review all code before accepting it into a release to ensure the code is free of bugs and errors [16].
- Use scripts to perform initial code reviews and catch errors [16].
- Use a group of members to perform tests in a beta version [16].
- Peer review changes. In some projects, changes proposed by developers with direct commit rights are also subject to review by other community members [63].
- Test programs produced by the project. Most projects doing repeatable testing do it by defining an automated test suite. If no test suite is available, there may be explicitly defined manual test cases, but this is much less likely [63].
- During code review, provide feedback about architecture, acceptable code practices, refactoring, and show the best ways to approach the problem [14].
- Define a team to test the performance and behavior of the application [14].

Manage technical decisions to improve robustness. A simple technical decision can have a lot of consequences on the quality of software products. As a result, ecosystems do their best to have robust practices around making decisions and maintaining product quality. Some ecosystems assign technical decisions to a small group of senior developers, while other ecosystems seek input on important matters

from all members. Each of these approaches has its advantages and disadvantages, and the ecosystem managing board should select the appropriate method for making technical decisions based on their priorities, goals, and features. The two practices I grouped into this sub-category are:

- Make technical decisions independently within each project to ensure that technical decisions are based on the attributes of the specific project [16].
- Assign a lead maintainer for each project to make technical decisions [16].

Increase Security to improve robustness. Privacy and security are factors that must be taken into consideration by different stakeholders of software ecosystems. In the world of technology, a leak of information from an ecosystem to the outside can severely damage its reputation; It can have irreversible impacts that can even destroy an ecosystem. As a result, one of the most important factors affecting the robustness of an ecosystem concerns its security practices. In some ecosystems, a group of members is assigned to ensure that security practices are taken seriously by other members. For example, malicious test emails are sent to members in an attempt to get them to click fake links or download documents and leak private information. If some members do not pay enough attention and interact with the malicious email, the security group will follow-up with education about proper security practices. The five practices I grouped into this sub-category are:

- Create a security committee to make important decisions about security matters [18].
- Provide different security levels for the ecosystem's members based on their tasks in the ecosystem to ensure that each member has access to only the information needed to perform their work [14].
- Use tools to publish known cybersecurity vulnerabilities. For example, Common Vulnerabilities and Exposures (CVE®) [14].
- Maintain a team to manage security problems and issues in the ecosystem [14].
- Look for and remove malicious code that harms the security of the ecosystem [14].

In this chapter, I described the taxonomy I built for the practices from the SLR. In the next chapter, I include a discussion about my findings and taxonomies. The next chapter also discusses the contributions and importance of my work and my work implications for various practitioners and researchers.

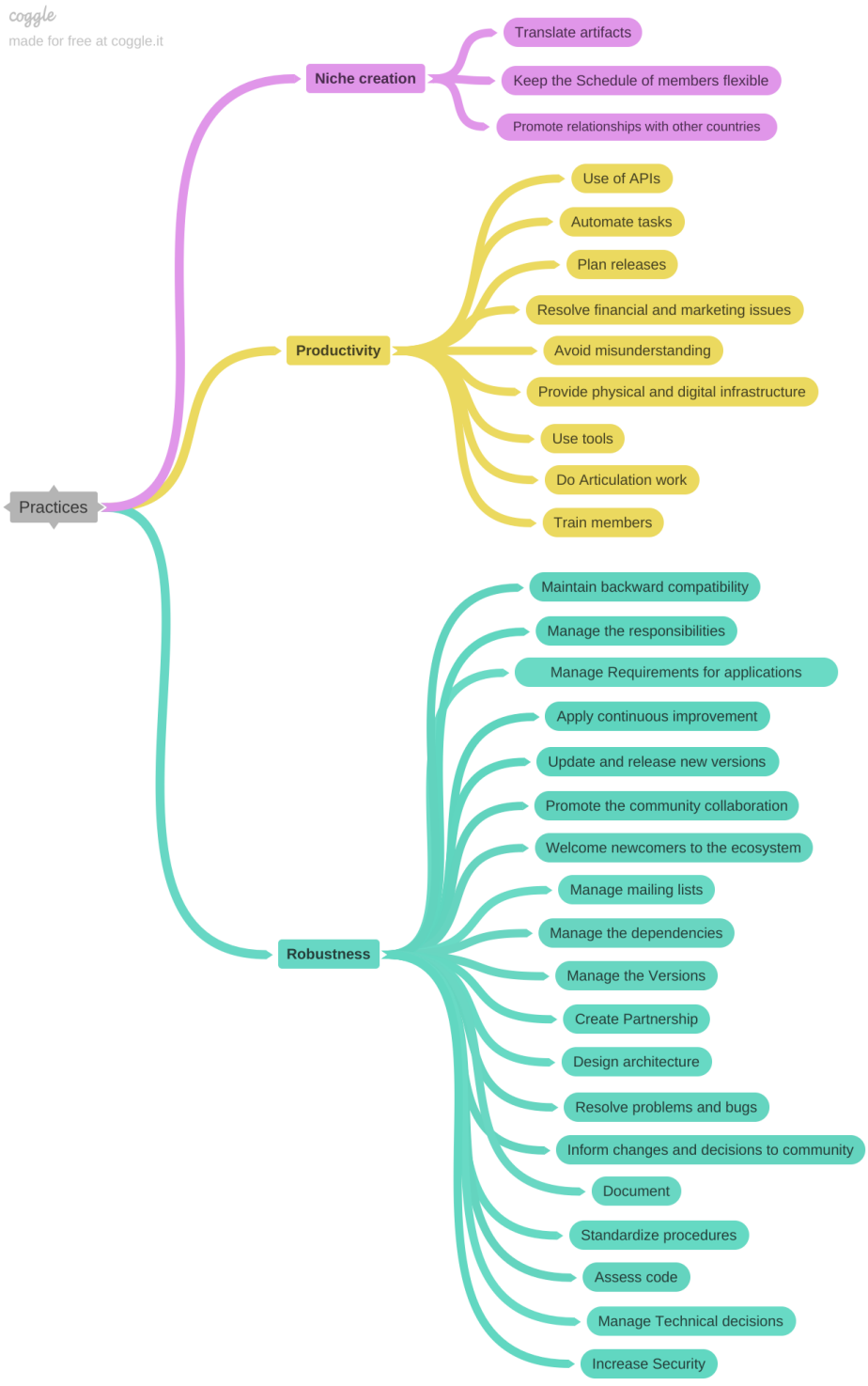

Figure 6.1: Categories and sub-categories of the software ecosystem health practices taxonomy

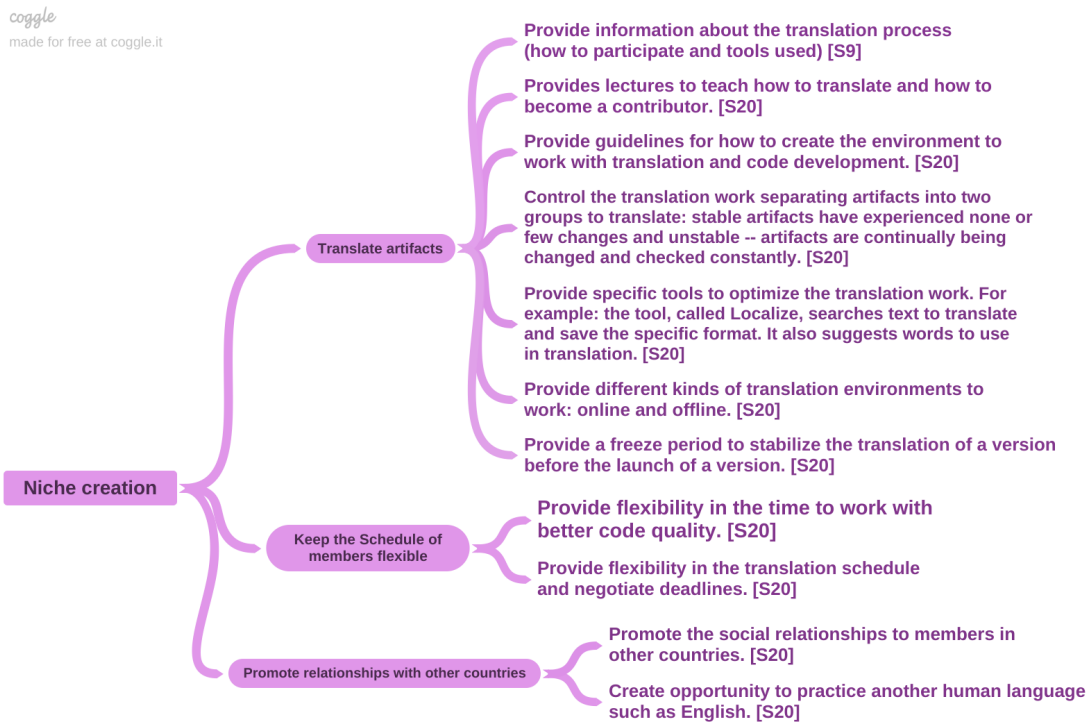

Figure 6.2: Some of the practices for the niche creation category—a single practice is used as an example for each of the sub-categories.

Related sub-category	Name of practice	Key area	Source
Use of APIs to improve productivity	Document APIs daily	Architectural Knowledge	[14]
Automate tasks to improve productivity	Provide an automated process to support continuous software engineering [61] and software system releases	Quality Management (QM) - refers to managing the quality of the various structures designed into the architecture	[20]
Plan releases to improve productivity	Define a schedule of releases that will affect the work of the entire community.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, making decisions, and so on.)	[16]
Resolve financial and marketing issues to improve productivity	Attract companies that invest money to support the ecosystem.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Avoid misunderstanding to improve productivity	Setting a code of conduct to avoid mistreatment among members	External Management (EM) -related to managing the interfaces third-party developers	[20]
Provide physical and digital infrastructure to improve productivity	Provide hardware and software resources to be used by the community	Resources Management	[14]
Use tools to improve productivity	Provide a virtual machine with preconfigured build environments, web-based IDEs, or a content management tool	External Management (EM) -related to managing the interfaces third-party developers	[20]
Do Articulation work to improve productivity	Tag all tasks with their degree of difficulty (easy, medium, difficult)	External Management (EM) -related to managing the interfaces third-party developers	[20]
Train members to improve productivity	Provide tutorials and video courses for the technologies used by the ecosystem to help less experienced members improve their knowledge and skills on these technologies	Architectural Knowledge	[14]

Table 6.2: Some of the practices for the productivity category—a single practice is used as an example for each of the sub-categories.

Figure 6.3: Sub-categories and practices in the productivity category within the software ecosystem health practices taxonomy

Related sub-category	Name of practice	Key area	Source
Maintain compatibility to improve robustness	Assign deciding about removing obsolete code to architects' leaders, respecting backward compatibility.	Change Management	[14]
Manage the responsibilities to improve robustness	Create confidence in members based on their work history and provide different levels of responsibilities for members based on their level of experience and skills.	Social (related to working together in the community)	[16]
Manage requirements for applications to improve robustness	Create requirements for applications by developers and end-users to ensure that requirements are based on both developers' and end-users' perspectives.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Apply continuous improvement to enhance robustness	Use lessons learned by past mistakes to modify and improve practices and make sure that past mistakes will not be repeated.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Update and release new versions to improve robustness	Identify and dismiss outdated information on websites to ensure that the websites' information and content are updated.	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Promote the community collaboration to improve robustness	Conduct an annual general meeting to discuss face-to-face questions from the community.	Social (related to working together in the community)	[16]
Welcome newcomers to the ecosystem for improving robustness	Point newcomers to easy tasks filtered by difficulty, skills needed, and topics.	External Management (EM) -related to managing the interfaces third-party developers	[20]
Manage mailing lists to improve robustness	Answer questions on the mailing list right away to make sure that questions and issues are solved quickly	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Manage dependency to improve robustness	Provide information about dependencies among artifacts to make dependencies clear	Changes Management (CM) - related to maintaining the changes and reducing the impact caused by the changes on the software architecture.	[20]
Manage versions to improve robustness	Conduct incremental transitions rather than releasing the software product all at once. This helps build the next versions of software products based on the received feedback on previous versions.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Create partnerships to improve robustness	Make partners explicit to show a powerful picture of the ecosystem to the outside world.	External Management	[14]
Design architecture to improve robustness	Build the architecture based on plugins to facilitate the decoupling of applications.	Change Management	[14]
Resolve problems and bugs in the software to improve robustness	Create a detailed step-by-step tutorial linking information about common problems and possible solutions to help the ecosystem members resolve issues and errors more easily.	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Inform changes and decisions to the ecosystem community for improving robustness	Discuss critical changes that will impact applications to ensure the community is aware of these important updates.	Change Management	[14]
Document to improve robustness	Keep documentation organized and indexed based on the level of difficulty, from beginners to experts, to help community members find documents based on their level of experience.	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Standardize procedures to improve robustness	Require that all infrastructure be under control of the ecosystem and be based on technologies created by the ecosystem.	Choice of Technology	[14]
Assess code to improve robustness	Review all code before accepting it into a release to ensure the code is free of bugs and errors	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Manage technical decisions to improve robustness	Make technical decisions independently within each project to ensure that technical decisions are based on the attributes of the specific project.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Increase Security to improve robustness	Provide different security levels for the ecosystem's members based on their tasks in the ecosystem to ensure that each member has access to only the information needed to perform their work.	External Management	[14]

Table 6.3: Some of the practices for the robustness category—a single practice is used as an example for each of the sub-categories.

Figure 6.4: Sub-categories and practices in the robustness category within the software ecosystem health practices taxonomy—part 1

Figure 6.5: Sub-categories and practices in the robustness category within the software ecosystem health practices taxonomy—part 2

Chapter 7

Discussion

This chapter includes a discussion about my findings and the taxonomies of metrics and practices I developed. I review the contributions of my thesis and the implications of my work for practitioners and researchers.

This thesis aims to build taxonomies of metrics and practices that are used to measure and improve software ecosystem health. To reach this goal, I conducted a systematic literature review to collect all the software ecosystem health metrics and practices. Then I categorized these collected metrics and practices to build comprehensive taxonomies that can be used by practitioners and researchers to measure and improve the health of software ecosystems based on their perspective regarding software ecosystem health.

7.1 Metrics

"If you cannot measure it, you cannot improve it" is an adage about the importance of using metrics. Since the beginning of software engineering, metrics (such as SLOCs) have been used to measure features of software development projects, their processes, and their contributors. Measuring the qualities of software ecosystems brings a new level of complexity: on the one hand, a software ecosystem is composed of several interrelated software projects, but on the other hand, there is a wide variety of interested stakeholders.

Different types of stakeholders are likely to be interested in measuring different aspects of the ecosystem. For example, an organization evaluating whether or not it is worth joining an ecosystem is likely interested in measuring how the ecosystem

can help them achieve their goals, while an organization already participating might be interested in knowing if they are receiving a return on investment for their contributions to the ecosystem. The literature I surveyed reflects this diversity: in the 40 papers I analyzed, I was able to identify 142 different metrics for measuring software ecosystem health.

An important factor in using metrics is how we interpret their values. For example, suppose we have a metric for the cost of using software products of an ecosystem. The ecosystem owners may be happy if the software products are more expensive because they can gain more profit, but on the other hand, the end-users who should pay for these products prefer to spend less money to use these products. As a result, in addition to various metrics that different stakeholders are interested in, different stakeholders interpret the same metrics differently.

Many of the metrics proposed in the literature are theoretical and have not been used or evaluated in practice. This can have different reasons. A possible reason can be that these metrics are difficult to be computed. For example, *simplicitly in using the products of software ecosystem* as a metric is difficult to be measured. Another possible reason for not using a metric in practice is not having access to the data required for measuring the metric. For example consider *profit of ecosystem* as a metric. To measure the profit of an ecosystem, access to the financial data is required, which is sometimes impossible. In this thesis, I mention whether each metric has been evaluated in practice or not, and if it has been evaluated, which studies have done this evaluation. I also observed metrics that were derived from another metric, with only small modifications. For example, suppose *number of software applications* as a metric: several metrics can be derived from this, like *number of active software applications*, *number of deactivated software applications*, or *number of newly installed software applications*. In this thesis, I tried to align these metrics.

7.2 Practices

A practice provides a means to resolve a specific problem in a structured and verifiable manner. Instead of addressing the entire issue, they address a particular aspect of a problem [36]. I believe that practices have a direct impact on the health of software ecosystems. A practice produces a result that can be expressed and measured by a metric, whether good or bad. For example, suppose a software ecosystem follows practices in the "Welcome newcomers to the ecosystem for improving robustness"

sub-category. In that case, the survival rate of newcomers in an ecosystem will be increased, and its results can be directly expressed and measured by metrics in the "Survival in ecosystem" sub-category. These metrics may be particularly important for managers to assess the retention of newcomers in the ecosystem.

Practices would be a way to lead a software ecosystem to achieve good health. All the methods used in the primary studies focused their efforts on assessing health through metrics. Just six studies specifically included practices in their assessments [S8, S9, S13, S19, S20, S22], and four of them have been written by the same authors. This lack of attention to the importance of practices and instead concentrating on metrics in the software ecosystem health literature is a gap that should be filled in future research. In this work, I took an important step to shed light on this gap and show the importance of practices for improving software ecosystems' health.

Most of the practices in the literature I reviewed were collected through interviews with stakeholders in software ecosystems. These stakeholders suggested the practices they believe positively impact the health of software ecosystems. Although some empirical studies have been conducted to validate certain practices [14], future empirical studies are needed to determine which of these practices positively impact software ecosystem health, and on the other hand, which of them have unexpected negative impacts.

There are varying opinions on the impact of practices on software ecosystem health. As I mentioned, most practices were collected by interviewing senior software engineers and other stakeholders in the ecosystems under study. Based on their experiences, they suggested practices that they felt improve the software ecosystem's health. These people might disagree with each other's opinions but that it might be good because it captures different perspectives regarding the concept of software ecosystem health and improve various aspects of software ecosystem health.

All the practices in our taxonomy are consistent with each other, and no two practices contradict each other. However, some of the practices have more robust and direct relations with each other. For example, "Tag all tasks by the degree of difficulty (easy, medium, difficult) to help developers select tasks based on their skills and experience" as a practice should be definitely conducted before the following practice because it makes the conducting of the following practice more efficient and productive: "Allocation of tasks by the developers' interests and discussing it with the other members to make sure that developers be assigned to tasks they are interested in."

7.3 How My Taxonomies can be Used by Practitioners and Researchers

Practitioners have different needs, priorities, and goals in software ecosystems. They usually use metrics and practices to support their challenging decision-making process. For example, if they want to join a new ecosystem, they measure a few metrics of the ecosystem to gain some quick insights into the ecosystem's health. Without the taxonomies I created, it is hard for practitioners to select a set of core metrics that give them helpful insights among the many available and unorganized metrics. Using my taxonomy, they can select the categories and sub-categories that are important for supporting their decisions. They can then use only the metrics in the selected sub-categories and ignore the rest of the sub-categories.

Researchers who wanted to create new metrics/practices, evaluate the existing metrics/practices, apply the metrics/practices for evaluating software ecosystems, and find the relations between metrics/practices may benefit from the taxonomies I created. Because there are so far unorganized metrics/practices in the literature, it is hard to see in which area there is a lack of metrics/practices. It is also hard to decide on measuring and improving each aspect of software ecosystems, which metrics/practices should be applied.

7.4 Contributions

The significant contribution from my thesis is organizing all of the identified metrics and practices into categories and sub-categories. The three main categories are Niche Creation, Productivity, and Robustness.

Metrics and practices under the niche creation category measure and improve an ecosystem's ability to grow and continue to exist. A significant aspect of this category is measuring and enhancing diversity, both of the software artifacts that compose it and its members. This category is also concerned with how an ecosystem fits its environment (e.g., how widely known or used).

Productivity metrics and practices measure and improve the ability of an ecosystem to convert inputs into outputs. Measuring and improving productivity in an ecosystem is tricky. Because different stakeholders/participants can belong to various organizations, it is difficult, if not impossible, to obtain information such as the

number of hours invested in the ecosystem and the tasks that such participants are expected to complete. For this reason, productivity is usually measured by observing the public actions that participants perform but without knowledge of how much time or effort they spend doing these things. As a consequence, these metrics tend to measure observable artifacts. For example, the artifacts' size, participants' activity, both in terms of development and communication, and whether the development processes are mature or not.

Finally, regarding robustness, these metrics and practices aim to evaluate and improve the ecosystem's ability to adapt to change. Factors such as the age of the ecosystem, how active its developers are, its participants' diversity, whether there is commercial support for users of the software created by the ecosystem, etc. are likely to impact whether the ecosystem can cope with disruptions.

The goals of my systematic literature review and taxonomy are to provide:

- A more recent comprehensive survey;
- A taxonomy that combines the results of previous categorizations and updates them with newer metrics and practices that have been proposed since; and
- A taxonomy that provides researchers and practitioners with the rationale for creating and using specific metrics and practices.

7.5 Implications for Stakeholders

As I mentioned above, software ecosystems are composed of a wide variety of stakeholders, including those considering joining the ecosystem, those who want to decide about investing in the ecosystem, those considering using the ecosystem's products, etc. I expect my taxonomy to simplify finding the metrics and practices that a stakeholder needs to use based on their goals. Furthermore, the categories and subcategories help clarify the different aspects of concern to ecosystems and provide an easier way for stakeholders to approach the many possible metrics and practices proposed in the literature.

All of the metrics and practices in my taxonomy can be used to measure and improve health at the ecosystem level, and I have included in my summary which metrics have been used and are shown to be useful. Sharing insights into which metrics have been used may help stakeholders be more confident in using them.

We also note that many of the metrics and practices can also measure and improve health at the project level, with some small modifications. For example, *the number of active developers* is a metric that can be used at both the ecosystem level and the project level. Thus my taxonomy also includes a broad set of metrics and practices that may help evaluate and improve health at the software project level.

7.6 Implications for Researchers

While researchers proposed a vast number of metrics and practices, I observed that they are primarily offered without empirical evidence. This points to an opportunity to do empirical studies that evaluate these metrics' effectiveness and practices for specific goals. Also, as one would expect, I identified that many metrics and practices are defined because they are easy to implement (this is especially true for metrics that measure entities or individuals, such as number of commits, number of developers, etc.). In contrast, other metrics and practices are much more challenging to measure and apply, if at all, such as market share of the ecosystem or simplicity in using the ecosystem.

Similarly, research is needed to ask those involved in a software ecosystem about what aspects of the ecosystem they need to measure and improve and about their experiences using currently implemented metrics and practices. Different stakeholders are likely to have different needs. For example, many metrics have been proposed to measure the size of an ecosystem: number of passive users, number of contributors, number of files in the ecosystem, number of bug-related activities, number of releases, etc. While each of these metrics evaluates a specific aspect of the ecosystem, it is not clear how they can be used to assess whether or not the ecosystem is healthy.

Perhaps one of the fundamental problems is that there is no agreed-upon definition of a SECO's health. Therefore, we should not consider these metrics and practices as a measure or improvement of the ecosystem; instead, each measure or improve a specific aspect of the ecosystem's health.

Another research aspect that is important to consider is that the purpose of evaluating the health of a SECO is usually to determine if the ecosystem is unhealthy (i.e., there are signs of potential problems in the ecosystem that might translate into a failure of the software the ecosystem is created around). Most SECO health metrics are defined to provide a raw value as a result (i.e., ten commits/day, seven core developers, two releases a year). However, these numbers do not directly address how

they should be interpreted: Is ten commits/day a sign of a healthy ecosystem? Is one commit a day unhealthy?

Researchers also need to address the interpretation of metrics. One way to do this may be to define some *reference* ecosystems measured with the results used as a reference. Hence, when a new ecosystem is measured, one can compare the result of a metric to the known ecosystems. This way, a value of 10 commits/day might be translated to "the number of commits per day is very similar to those of ecosystem A".

And finally, another aspect to consider is the creation of *indices*. Most of the metrics surveyed measure a single element of the ecosystem: in statistical terms, they measure one variable of the ecosystem, such as the number of contributors, market share, etc. In statistics, an index is created by combining different measurements into a single one. While no index is perfect, each index tries to abstract the information of different metrics into a single value. In other words, instead of interpreting a multi-dimensional space, one has to interpret a single dimension.

In this chapter, I included a discussion about my findings and taxonomies. The next chapter mentions the limitations which I faced in my research.

Chapter 8

Limitations

In this chapter, I mention the limitations which I faced in my research. Some of these limitations are in conducting the systematic literature review, and the rest are in building the taxonomies based on the SLR findings.

I aimed to select a search string that was general enough to find all the papers related to my research questions. Also, I chose the most important databases in the area of software engineering. However, like any other systematic literature review, the choice of search string and databases may have excluded some papers related to my research questions. There may be other metrics, practices, or evaluations of the metrics I found in literature I did not see through my search process. Bias may have also been introduced in the process of study selection and data collection from the primary studies.

Bias could have also been introduced in the process of categorizing the metrics and practices. First, another researcher (a PhD student in my research group) and I independently clustered the metrics. Then, we discussed the differences to find a consensus. When consensus wasn't reached, my opinion was used. After clustering the metrics independently by two of us and comparing our results, I gained enough context and experience to cluster the practices easily on my own.

It is worthwhile to mention that some of the categories in my taxonomies can have some overlaps with each other if I also consider indirect relations. It means that if I consider the indirect relations, some metrics or practices could be placed in multiple sub-categories or categories in my taxonomies. In this work, I only considered the most direct relations to categorize my taxonomies' metrics and practices. As a result, my taxonomies' categories do not have any overlap. However, I identify this as a threat to the validity of my work.

There are circumstances beyond my control that may have influenced the results of my study. For instance, the search engines' performance could have affected the completeness of the primary studies found.

In this chapter, I described the limitations which I faced in my research. The next chapter presents future work that can be done based on my work.

Chapter 9

Future Work

This chapter presents the future work that can be done based on my work. The information presented in this chapter is an important contribution to my work and gives a clear research path for researchers who want to research software ecosystem health.

The taxonomies presented in this thesis are the most comprehensive taxonomies of SECO health metrics and practices to my knowledge. The comprehensiveness of these taxonomies can be evaluated by new metrics and practices proposed in future studies, and the extent to which the new metrics and practices can be placed in the categories and sub-categories of these taxonomies.

I believe that stakeholders measure an ecosystem because it supports their decision-making process. There are different types of decisions that stakeholders can make, such as joining a new ecosystem as a contributor or watcher, investing time, energy, or money in an ecosystem, using the products or applications of an ecosystem as end-users creating code dependency on an ecosystem. To make each of these decisions, stakeholders need to consider specific metrics; pragmatically, they can't use all of them. Future research can consider how specific metrics help in this regard and map to particular types of decisions.

Future work could also categorize the metrics and practices in my taxonomies according to the needs of different types of stakeholders involved in a SECO. Different types of stakeholders might need small subsets of the metrics and practices to obtain insights into the SECO's health. For example, newcomers to a SECO may want to know if they will survive in the ecosystem. In this case, they can use the metrics in the *survival in ecosystem* category and ignore the other categories. As another example, suppose that investors need to decide on investing in a SECO. The investors may only

care about the SECO's financial wellness to see if they will receive a return on their investments so that they can look at the metrics in the *financial wellness* category and ignore the rest.

The metrics and practices included in my taxonomy are at different levels of practicality. Some metrics are more abstract, such as *simplicity in using the ecosystem*, whereas some of them are very concrete, such as *added KLOC in the last 30 days*. Finding suitable methods to measure and apply the more abstract metrics and practices would be beneficial and points to future work.

There are various methods for measuring and applying the metrics and practices in my taxonomies. Some metrics need to be measured by data mining approaches. In contrast, we need to ask stakeholders directly through surveys and interviews or observe their behaviors to measure other metrics. There may be a relation between the simplicity of the method for measuring or applying a metric or practice and the extent to which a metric or practice is measured or applied in practice. Future research can compare these methods based on simplicity, practicality, etc. Future research may also find alternative and more practical ways to measure and apply specific metrics or practices.

There are different types of relationships between some of the metrics and practices in my taxonomies. For example, *customer satisfaction percentage* as a metric has an inverse correlation with *total number of customer complaints*. It means that when one of them increases, the other one decreases, and vice versa. Finding these relationships between metrics and practices can be a future research direction.

The metrics in my taxonomy can be used to compare one ecosystem's health over time or compare an ecosystem's health with other ecosystems. Future research can investigate which of these comparisons is more important for different types of stakeholders; specific metrics should be used for each of these comparisons.

Some of the metrics are measured over time like *revenue increase*, and the rest can be measured at any moment like *profit of the ecosystem*. Future research can compare these two categories of metrics based on their importance and impact in measuring health.

In this chapter, I presented future work that can be done based on my work. The next chapter concludes the thesis and provides a summary of the work completed.

Chapter 10

Conclusions

Software ecosystems have been studied extensively in the last few years. Specifically, SECO health is a concept that has gained significant attention due to the broad range of challenges that various types of stakeholders face in an ecosystem, from making a strategic decision for relying on an ecosystem to deciding about investing in an ecosystem. Researchers have proposed many metrics and practices to evaluate the health of SECOs from different viewpoints. To the best of my knowledge, no previous study has tried to categorize all of these metrics and practices and make comprehensive taxonomies. In this thesis, I took the first step to fill this gap.

As part of a two-phase study, I performed a systematic literature review to collect all the metrics, practices, and definitions of software ecosystem health. I then synthesized my findings from the SLR to create a definition for SECO health and build comprehensive taxonomies of metrics and practices.

I believe that my two taxonomies are the first step towards comprehensive frameworks for measuring and improving the health of SECOs. Different types of stakeholders can use the metrics, practices, and categories of my taxonomies to measure and improve the health of SECOs based on their various needs. My taxonomies also provide a way to measure and improve the health of SECOs from different viewpoints.

Bibliography

[1] Khubaib Amjad Alam, Rodina Ahmad, Adnan Akhunzada, Mohd Hairul Nizam Md Nasir, and Samee U Khan. Impact analysis and change propagation in service-oriented enterprises: A systematic review. *Information Systems*, 54:43–73, 2015.

[2] Daniel Alami, María Rodríguez, and Slinger Jansen. Relating health to platform success: exploring three e-commerce ecosystems. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, pages 1–6, 2015.

[3] Carina Alves, Joyce Oliveira, and Slinger Jansen. Understanding governance mechanisms and health in software ecosystems: a systematic literature review. In *International Conference on Enterprise Information Systems*, pages 517–542. Springer, 2017.

[4] Amel Ben Hadj Salem Mhamdia. Performance measurement practices in software ecosystem. *International Journal of Productivity and Performance Management*, 62(5):514–533, 2013.

[5] Matthijs Berkhout, Fons van den Brink, Mart van Zwienen, Paul van Vulpen, and Slinger Jansen. Software ecosystem health of cryptocurrencies. In *International Conference of Software Business*, pages 27–42. Springer, 2018.

[6] Jan Bosch. From software product lines to software ecosystems. In *Proceedings of the 13th international software product line conference*, pages 111–119. Carnegie Mellon University, 2009.

[7] Jan Bosch. Architecture challenges for software ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, pages 93–95, 2010.

[8] Jan Bosch and Petra Bosch-Sijtsema. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67–76, 2010.

[9] Jan Bosch and Petra Bosch-Sijtsema. Esao: A holistic ecosystem-driven analysis model. In *International Conference of Software Business*, pages 179–193. Springer, 2014.

[10] Susanne Boshuis, Tim Braam, Alejandra Pedroza Marchena, and Slinger Jansen. The effect of generic strategies on software ecosystem health: the case of cryptocurrency ecosystems. In *2018 IEEE/ACM 1st International Workshop on Software Health (SoHeal)*, pages 10–17. IEEE, 2018.

[11] Piers RJ Campbell and Faheem Ahmed. An assessment of mobile os-centric ecosystems. *Journal of theoretical and applied electronic commerce research*, 6(2):50–62, 2011.

[12] Iuri Carvalho, Fernanda Campos, Regina Braga, José Maria N David, Victor Stroelle, and Marco Antônio Araújo. Heal me-an architecture for health software ecosystem evaluation. In *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*, pages 59–65. IEEE, 2017.

[13] Kevin Crowston, James Howison, and Hala Annabi. Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practice*, 11(2):123–148, 2006.

[14] Simone da Silva Amorim, Sandro Santos Andrade, John D McGregor, Eduardo Santana de Almeida, and G Chavez Christina von Flach. Tailoring the nfr framework for measuring software ecosystems health. In *WER*, 2018.

[15] Simone da Silva Amorim, Eduardo Santana de Almeida, John D McGregor, and Christina von Flach G. Chavez. Towards an evaluation method for software ecosystem practices. In *Proccedings of the 10th European Conference on Software Architecture Workshops*, pages 1–4, 2016.

[16] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach Garcia Chavez. Understanding the effects of practices on kde

ecosystem health. In *IFIP International Conference on Open Source Systems*, pages 89–100. Springer, Cham, 2017.

[17] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and G Chavez Christina von Flach. Software ecosystems' architectural health: another view. In *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*, pages 66–69. IEEE, 2017.

[18] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. Software ecosystems architectural health: challenges x practices. In *Proccedings of the 10th European Conference on Software Architecture Workshops*, pages 1–7, 2016.

[19] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G Chavez. The architect's role in software ecosystems health. In *Proceedings of the 2nd Workshop on Social, Human, and Economic Aspects of Software*, pages 1–4. ACM, 2017.

[20] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. Educating to achieve healthy open source ecosystems. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, pages 1–7, 2018.

[21] Simone da Silva Amorim, Félix Simas S Neto, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G Chavez. How has the health of software ecosystems been evaluated?: A systematic review. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 14–23. ACM, 2017.

[22] Awdren de Lima Fontão, Rodrigo Pereira dos Santos, and Arilo Claudio Dias-Neto. Mobile software ecosystem (mseco): a systematic mapping study. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 653–658. IEEE, 2015.

[23] Erik den Hartigh, Michiel Tol, and Wouter Visscher. The health measurement of a business ecosystem. In *Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting*, pages 1–39, 2006.

[24] Joost Dijkers, Rowan Sincic, Nicole Wasankhasit, and Slinger Jansen. Exploring the effect of software ecosystem health on the financial performance of the open source companies. In *2018 IEEE/ACM 1st International Workshop on Software Health (SoHeal)*, pages 48–55. IEEE, 2018.

[25] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. Measuring the quality of open source software ecosystems using queso. In *International Conference on Software Technologies*, pages 39–62. Springer, 2014.

[26] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. Open source software ecosystems: A systematic mapping. *Information and software technology*, 91:160–185, 2017.

[27] Óscar Hernán Franco Bedoya, David Ameller, Dolors Costal Costa, and Javier Franch Gutiérrez. Queso v2. 0 a quality model for open source software ecosystems: List of measures. 2016.

[28] JK Gershenson, GJ Prasad, and Y Zhang. Product modularity: definitions and benefits. *Journal of Engineering design*, 14(3):295–313, 2003.

[29] Sami Hyrynsalmi and Matti Mäntymäki. Is ecosystem health a useful metaphor? towards a research agenda for ecosystem health research. In *Conference on e-Business, e-Services and e-Society*, pages 141–149. Springer, 2018.

[30] Sami Hyrynsalmi, Karl Michael Popp, Peter Buxmann, Thomas Aidan Curran, Gerald Eichler, and Slinger Jansen. To redefine ecosystem health, or not to redefine? a view of scientific knowledge on the" software ecosystem health" concept. In *Proceedings of the European Workshop on Software Ecosystems 2015*, pages 47–51. Synomic Academy, Books on Demand, 2016.

[31] Sami Hyrynsalmi, Jukka Ruohonen, and Marko Seppänen. Healthy until otherwise proven: some proposals for renewing research of software ecosystem health. In *2018 IEEE/ACM 1st International Workshop on Software Health (SoHeal)*, pages 18–24. IEEE, 2018.

[32] Sami Hyrynsalmi, Marko Seppänen, Tiina Nokkala, Arho Suominen, and Antero Järvi. Wealthy, healthy and/or happy—what does 'ecosystem health'stand for? In *International Conference of Software Business*, pages 272–287. Springer, 2015.

[33] Marco Iansiti and Roy Levien. Keystones and dominators: Framing the operational dynamics of business ecosystems. *Boston, Estados Unidos*, 2002.

[34] Marco Iansiti and Roy Levien. Keystones and dominators: Framing operating and technology strategy in a business ecosystem. *Harvard Business School, Boston*, pages 24–25, 2004.

[35] Marco Iansiti, Roy Levien, et al. Strategy as ecology. *Harvard business review*, 82(3):68–81, 2004.

[36] Ivar Jacobson, Pan Wei Ng, and Ian Spence. Enough of processes-lets do practices. *J. Object Technol.*, 6(6):41–66, 2007.

[37] Slinger Jansen. Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology*, 56(11):1508–1519, 2014.

[38] Slinger Jansen and Michael A Cusumano. Defining software ecosystems: a survey of software platforms and business network governance. *Software ecosystems: analyzing and managing business networks in the software industry*, 13, 2013.

[39] Slinger Jansen, Michael A Cusumano, and Sjaak Brinkkemper. *Software ecosystems: analyzing and managing business networks in the software industry*. Edward Elgar Publishing, 2013.

[40] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. A sense of community: A research agenda for software ecosystems. In *2009 31st International Conference on Software Engineering-Companion Volume*, pages 187–190. IEEE, 2009.

[41] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[42] Fanny Lalonde Lévesque, Anil Somayaji, Dennis Batchelder, and Jose M Fernandez. Measuring the health of antivirus ecosystems. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 101–109. IEEE, 2015.

[43] Garm Lucassen, Kevin Van Rooij, and Slinger Jansen. Ecosystem health of cloud paas providers. In *International Conference of Software Business*, pages 183–194. Springer, 2013.

[44] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264–275, 2010.

[45] Konstantinos Manikas and Klaus Marius Hansen. Reviewing the health of software ecosystems–a conceptual framework proposal. In *Proceedings of the 5th international workshop on software ecosystems (IWSECO)*, pages 33–44. Citeseer, 2013.

[46] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems–a systematic literature review. *Journal of Systems and Software*, 86(5):1294–1306, 2013.

[47] Konstantinos Manikas and Dimosthenis Kontogiorgos. Characterizing software activity: The influence of software to ecosystem health. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, pages 1–6, 2015.

[48] Tom Mens, Bram Adams, and Josianne Marsan. Towards an interdisciplinary, socio-technical analysis of software ecosystem health. *arXiv preprint arXiv:1711.04532*, 2017.

[49] David G Messerschmitt and Clemens Szyperski. Software ecosystem: Understanding an indispensable technology and industry (chapter 6: Organization of the software value chain, 2003.

[50] Amel Ben Hadj Salem Mhamdia. Performance measurement practices in software ecosystem. *International Journal of Productivity and Performance Management*, 2013.

[51] Yannick Mijsters, Amr Mustafa, Ionut Mihai, and Slinger Jansen. On the nature of software sub-ecosystems and their health. In *2018 IEEE/ACM 1st International Workshop on Software Health (SoHeal)*, pages 25–32. IEEE, 2018.

[52] John Yates Monteith, John D McGregor, and John E Ingram. Proposed metrics on ecosystem health. In *Proceedings of the 2014 ACM international workshop on Software-defined ecosystems*, pages 33–36, 2014.

[53] James F Moore. Predators and prey: a new ecology of competition. *Harvard business review*, 71(3):75–86, 1993.

[54] James F Moore. *The death of competition: leadership and strategy in the age of business ecosystems*. HarperBusiness New York, 1996.

[55] Marc Oriol, Oscar Franco-Bedoya, Xavier Franch, and Jordi Marco. Assessing open source communities' health using service oriented computing concepts. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–6. IEEE, 2014.

[56] Patrizio Pelliccione. Open architectures and software evolution: the case of software ecosystems. In *2014 23rd Australian Software Engineering Conference*, pages 66–69. IEEE, 2014.

[57] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, 2015.

[58] Konstantinos Plakidas, Daniel Schall, and Uwe Zdun. Evolution of the r software ecosystem: Metrics, relationships, and their impact on qualities. *Journal of Systems and Software*, 132:119–146, 2017.

[59] Konstantinos Plakidas, Srdjan Stevanetic, Daniel Schall, Tudor B Ionescu, and Uwe Zdun. How do software ecosystems evolve? a quantitative assessment of the r ecosystem. In *Proceedings of the 20th International Systems and Software Product Line Conference*, pages 89–98. ACM, 2016.

[60] Filip Radulovic and Raúl García-Castro. Extending software quality models-a sample in the semantic technologies domain. 2011.

[61] Henning Schulz, Tobias Angerstein, and André van Hoorn. Towards automating representative load testing in continuous software engineering. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 123–126, 2018.

[62] Maha Shaikh and Natalia Levina. Selecting an open innovation community as an alliance partner: Looking for healthy communities and ecosystems. *Research Policy*, 48(8):103766, 2019.

[63] Martín Soto and Marcus Ciolkowski. The qualoss open source assessment model measuring the performance of open source communities. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 498–501. IEEE, 2009.

[64] Lamia Soussi, Zeena Spijkerman, and Slinger Jansen. A case study of the health of an augmented reality software ecosystem: Vuforia. In *International Conference of Software Business*, pages 145–152. Springer, 2016.

[65] Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[66] Jose Teixeira and Sami Hyrynsalmi. How do software ecosystems co-evolve? In *International Conference of Software Business*, pages 115–130. Springer, 2017.

[67] Sonny Van Lingen, Adrien Palomba, and Garm Lucassen. On the software ecosystem health of open source content management systems. In *5th International Workshop on Software Ecosystems (IWSECO 2013)*, page 38. Citeseer, 2013.

[68] Paul Van Vulpen, Abel Menkveld, and Slinger Jansen. Health measurement of data-scarce software ecosystems: a case study of apple's researchkit. In *International Conference of Software Business*, pages 131–145. Springer, 2017.

[69] Lei Wang, Jing Wan, and Xinshu Gao. Toward the health measure for open source software ecosystem via projection pursuit and real-coded accelerated genetic. *IEEE Access*, 7:87396–87409, 2019.

[70] Claudia Werner and Slinger Jansen. A systematic mapping study on software ecosystems from a three-dimensional perspective. *Software Ecosystems: analyzing and managing business networks in the Software Industry*, pages 59–81, 2013.

[71] Andrea Wiggins, James Howison, and Kevin Crowston. Heartbeat: measuring active user base and potential user interest in floss projects. In *IFIP International Conference on Open Source Systems*, pages 94–104. Springer, 2009.

[72] Krzysztof Wnuk, Konstantinos Manikas, Per Runeson, Matilda Lantz, Oskar Weijden, and Hussan Munir. Evaluating the governance model of hardware-

dependent software ecosystems–a case study of the axis ecosystem. In *International Conference of Software Business*, pages 212–226. Springer, 2014.

[73] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 38. Citeseer, 2014.

[74] Donald Wynn Jr. Assessing the health of an open source ecosystem. In *Emerging Free and Open Source Software Practices*, pages 238–258. IGI Global, 2007.

[75] Abubakar Zakari, Sai Peck Lee, Rui Abreu, Babiker Hussien Ahmed, and Rasheed Abubakar Rasheed. Multiple fault localization of software programs: A systematic literature review. *Information and Software Technology*, page 106312, 2020.

Appendix A

List of Selected Primary Studies

In this section, I presents the final set of selected primary studies based on the inclusion and exclusion criteria of the SLR, including ID, authors, year, title, and publication venue corresponding to each primary study.

ID	Author(s)	Year	Title	Publication venue	Reference
S1	Franco Bedoya, Óscar Hernán, David Ameller, Dolors Costal Costa, and Javier Franch Gutiérrez	2016	QuESo V2. 0 a quality model for open source software ecosystems: List of measures	UPCommons	[27]
S2	den Hartigh, Erik, Michiel Tol, and Wouter Visscher	2006	The health measurement of a business ecosystem	Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting	[23]
S3	Lucassen, Garm, Kevin Van Rooij, and Slinger Jansen	2013	Ecosystem health of cloud PaaS providers	International Conference on Software Business	[43]
S4	Jansen, Slinger	2014	Measuring the health of open source software ecosystems: Beyond the scope of project health	Information and Software Technology	[37]
S5	Alami, Daniel, María Rodríguez, and Slinger Jansen	2015	Relating health to platform success: exploring three e-commerce ecosystems	Proceedings of the 2015 European Conference on Software Architecture Workshops	[2]
S6	Van Lingen, Sonny, Adrien Palomba, and Garm Lucassen	2013	On the software ecosystem health of open source content management systems	5th International Workshop on Software Ecosystems (IWSECO 2013)	[67]
S7	Oriol, Marc, Oscar Franco-Bedoya, Xavier Franch, and Jordi Marco	2014	Assessing open source communities' health using Service Oriented Computing concepts	IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)	[55]
S8	Soto, Martín, and Marcus Ciolkowski	2009	The QualOSS open source assessment model measuring the performance of open source communities	3rd International Symposium on Empirical Software Engineering and Measurement	[63]
S9	da Silva Amorim, Simone, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez	2018	Educating to achieve healthy open source ecosystems	Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings	[20]
S10	Wang, Lei, Jing Wan, and Xinshu Gao	2019	Toward the Health Measure for Open Source Software Ecosystem Via Projection Pursuit and Real-Coded Accelerated Genetic	IEEE Access (Volume: 7)	[69]
S11	Lévesque, Fanny Lalonde, Anil Somayaji, Dennis Batchelder, and Jose M. Fernandez	2015	Measuring the health of antivirus ecosystems	10th International Conference on Malicious and Unwanted Software (MALWARE)	[42]
S12	Manikas, Konstantinos, and Klaus Marius Hansen	2013	Reviewing the health of software ecosystems-a conceptual framework proposal	Proceedings of the 5th international workshop on software ecosystems (IWSECO)	[45]
S13	da Silva Amorim, Simone, Sandro Santos Andrade, John D. McGregor, Eduardo Santana de Almeida, and G. Chavez Christina von Flach	2018	Tailoring the NFR Framework for Measuring Software Ecosystems Health	21st Workshop on Requirements Engineering	[14]

Continued on next page

ID	Author(s)	Year	Title	Publication venue	Reference
S14	Van Vulpen, Paul, Abel Menkveld, and Slinger Jansen	2017	Health measurement of data-scarce software ecosystems: a case study of Apple's ResearchKit	International Conference of Software Business	[68]
S15	Berkhout, Matthijs, Fons van den Brink, Mart van Zwienen, Paul van Vulpen, and Slinger Jansen	2018	Software ecosystem health of cryptocurrencies	International Conference of Software Business	[5]
S16	Boshuis, Susanne, Tim Braam, Alejandra Pedroza Marchena, and Slinger Jansen	2018	The effect of generic strategies on software ecosystem health: the case of cryptocurrency ecosystems	IEEE/ACM 1st International Workshop on Software Health (SoHeal)	[10]
S17	Dijkers, Joost, Rowan Sincic, Nicole Wasankhasit, and Slinger Jansen	2018	Exploring the effect of software ecosystem health on the financial performance of the open source companies	IEEE/ACM 1st International Workshop on Software Health (SoHeal)	[24]
S18	da Silva Amorim, Simone, Félix Simas S. Neto, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez	2017	How has the health of software ecosystems been evaluated? A systematic review	Proceedings of the 31st Brazilian symposium on software engineering	[21]
S19	da Silva Amorim, Simone, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez	2016	Software ecosystems architectural health: challenges x practices	Proccedings of the 10th European Conference on Software Architecture Workshops	[18]
S20	da Silva Amorim, Simone, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach Garcia Chavez	2017	Understanding the Effects of Practices on KDE Ecosystem Health	IFIP International Conference on Open Source Systems	[16]
S21	da Silva Amorim, Simone, Eduardo Santana de Almeida, John D. McGregor, and Christina von Flach G. Chavez	2016	Towards an evaluation method for software ecosystem practices	Proceedings of the 10th European Conference on Software Architecture Workshops	[15]
S22	Wnuk, Krzysztof, Konstantinos Manikas, Per Runeson, Matilda Lantz, Oskar Weijden, and Hussan Munir	2014	Evaluating the governance model of hardware-dependent software ecosystems–a case study of the axis ecosystem	International Conference of Software Business	[72]
S23	Mijsters, Yannick, Amr Mustafa, Ionut Mihai, and Slinger Jansen	2018	On the nature of software sub-ecosystems and their health	IEEE/ACM 1st International Workshop on Software Health (SoHeal)	[51]
S24	Mhamdia, Amel Ben Hadj Salem	2013	Performance measurement practices in software ecosystem	International Journal of Productivity and Performance Management	[50]
S25	Campbell, Piers RJ, and Faheem Ahmed	2011	An assessment of mobile OS-centric ecosystems	Journal of theoretical and applied electronic commerce research 6, no. 2	[11]

Continued on next page

ID	Author(s)	Year	Title	Publication venue	Reference
S26	Carvalho, Iuri, Fernanda Campos, Regina Braga, José Maria N. David, Victor Stroelle, and Marco Antônio Araújo	2017	HEAL ME-An Architecture for Health Software Ecosystem Evaluation	IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)	[12]
S27	Shaikh, Maha, and Natalia Levina	2019	Selecting an open innovation community as an alliance partner: Looking for healthy communities and ecosystems	Research Policy 48, no. 8	[62]
S28	Monteith, John Yates, John D. McGregor, and John E. Ingram	2014	Proposed metrics on ecosystem health	Proceedings of the 2014 ACM international workshop on Software-defined ecosystems	[52]
S29	Soussi, Lamia, Zeena Spijkerman, and Slinger Jansen	2016	A case study of the health of an augmented reality software ecosystem: Vuforia	International Conference of Software Business	[64]
S30	Plakidas, Konstantinos, Daniel Schall, and Uwe Zdun	2017	Evolution of the R software ecosystem: Metrics, relationships, and their impact on qualities	Journal of Systems and Software	[58]
S31	da Silva Amorim, Simone, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez	2017	The Architect's Role in Software Ecosystems Health	Proceedings of the 2nd Workshop on Social, Human, and Economic Aspects of Software	[19]
S32	da Silva Amorim, Simone, John D. McGregor, Eduardo Santana de Almeida, and G. Chavez Christina von Flach	2017	Software ecosystems' architectural health: another view	IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)	[17]
S33	Manikas, Konstantinos, and Dimosthenis Kontogiorgos	2015	Characterizing software activity: The influence of software to ecosystem health	Proceedings of the 2015 European Conference on Software Architecture Workshops	[47]
S34	Iansiti, Marco, and Roy Levien	2002	Keystones and dominators: Framing the operational dynamics of business ecosystems	Harvard Business School	[33]
S35	Mens, Tom, Bram Adams, and Josianne Marsan	2017	Towards an interdisciplinary, socio-technical analysis of software ecosystem health	arXiv preprint	[48]

Continued on next page

ID	Author(s)	Year	Title	Publication venue	Reference
S36	Hyrynsalmi, Sami, Marko Seppänen, Tiina Nokkala, Arho Suominen, and Antero Järvi	2015	Wealthy, Healthy and/or Happy — What does 'Ecosystem Health' Stand for?	International Conference of Software Business	[32]
S37	Hyrynsalmi, Sami, Jukka Ruohonen, and Marko Seppänen	2018	Healthy until otherwise proven: some proposals for renewing research of software ecosystem health	IEEE/ACM 1st International Workshop on Software Health (SoHeal)	[31]
S38	Franco-Bedoya, Oscar, David Ameller, Dolors Costal, and Xavier Franch	2014	Measuring the quality of open source software ecosystems using queso	International Conference on Software Technologies	[25]
S39	Hyrynsalmi, Sami, and Matti Mäntymäki	2018	Is ecosystem health a useful metaphor? towards a research agenda for ecosystem health research	Conference on e-Business, e-Services and e-Society	[29]
S40	Alves, Carina, Joyce Oliveira, and Slinger Jansen	2017	Understanding governance mechanisms and health in software ecosystems: A systematic literature review	International Conference on Enterprise Information Systems	[3]

Table A.1: Final set of selected primary studies based on the inclusion and exclusion criteria of the SLR

Appendix B

List of Included Metrics in My Taxonomy

Here, I presents all the 142 metrics I included in my taxonomy (with their related attributes)

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Connection with other entities	Number of partners and embeddedness	Are there organizations that can provide different types of support to an OSSECO-community?	More is better. More partnerships indicate a strong project that is well embedded in the community	[27,37]
Size of ecosystem (People)	Number of passive users	How many people are just downloading and using the software produced by the OSSECO-community?	More is better. Passive users are essential for a sustainable OSSECO-community	[27]
Size of ecosystem (People)	Number of contributors	How many people are collaborating in different types of activities in the OSSECO-community?	More is better. Number of active and mature contributors is a measure that indicates a healthy OSS-community	[27]
Size of ecosystem (People)	Size of network community	What is the size of the OSSECO-community social network?	More is better. Networks with more nodes and connections are bigger and have a better structure.	[27]
Size of ecosystem (People)	Number of ecosystem-community members	How many people are in the OSSECO-community?	More is better. More members indicate that the OSS-community have a good structure for maintaining its products	[27]
Activeness in maintenance	Bug tracking activity	Is the OSSECO- community active in the bug tracking system?	Bug tracking activity is an indicator of OSSECO-community activity Any of measures can be used to assess the activity in the OSSECO-community, e.g.: A, B, F= can be used to assess the OSSECO community users activity. E, G= Can be used to assess the OSSECO- community contributors activity. If E is near to 0 is better. B= Might play a vital contribution for members participation in the OSSECO- community activities	[27]
Activeness of artifacts	Date of last commit	How is the OSS-community ctuality?	More recently is better a OSS-community that has more actuality is more active.	[27]
Activeness of artifacts	number of major releases per year	How is the OSS-community evolving?	More is better The number of project releases show the evolution in an OSSECO-community	[27]
Activeness of artifacts	Date of last release	How is the OSSECO-community actuality?	More recently is better a OSS-community that has more actuality is more active.	[27]
Size of ecosystem (Artifacts)	Number of files per version	How is the size evolution of the OSS-community projects?	More is better. An activity OSS-community create many files. It is possible to use any measure A, B C to assess the OSS- community activeness	[27]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Amount of activity on artifacts	Community contributor activity	How is the commits activity in the OSSECO- community projects?	High is better Commits activity is an indicator of OSS-community activity [12].	[27]
Longevity of activity	Community commit rate	How long between commits of a OSS-community?	high is better. higher rate can be interpreted like a more active OSSECO- community.	[27]
Longevity of activity	Ecosystem- community activity period	Is it possible to know whether a OSS-community was active all the time or there have been some gaps of inactivity during its lifetime?	More recently is better A OSS-community that has more actuality is more active.	[27]
Amount of communication	Community Timelines	How is the OSSECO- community timeliness?	Low is better	[27]
Amount of communication	Number of events	How is the social activity in a OSSECO- community?	More people are better and more events are better	[27]
Diversity in members	Geographical members' distribution	Are the members of the OSSECO- community geographically distributed?	More is better More members distributed implies more heterogeneity.	[27]
Diversity in members	Member activity types	Are the OSSECO-community members distributed across different activity types?	Closer to 0 is better A zero value for these indices implies a uniform distribution. A value of 1 means inequality.	[27]
Diversity in artifacts	Variety in ecosystem-community projects	How many kinds of projects has the OSSECO- community?	More is better A large variety in projects is an indicator that there are many niches, platforms, domains, etc., in which a new player can become active.	[27,37]
Diversity in members	Organizations members' distributions	How are the affiliations of OSSECO- community members to organizations.	More is better	[27]
Survival in ecosystem	Contributor survival rate	What is the number of surviving contributors in the OSSECO community?	Higher is better	[27]
Survival in ecosystem	Community new members rate	Is the number of OSSECO-community members evolving?	Higher is better. More new members more survival likelihood	[27]
Survival in ecosystem	Community new contributors rate	Is the number of OSSECO-community contributors evolving?	Higher is better More new contributors more survival likelihood	[27]
Balance in members	Contributors commit time distribution	How long between commits of all contributors?	Lower is better	[27]
Longevity of activity	Statistical characteristics of commits	How is the variation of commits across OSSECO-community history?	High values are better	[27]
Balance in members	Community activities distribution	Are the activities distributions balanced in the OSSECO-community projects?	Closer to 0 is better A zero value for these indices implies a uniform distribution. A value of 1 means inequality.	[27]
Balance in members	Developers activity on releases distribution	How is the developers participation in the OSSECO- community?	Lower is better	[27]
Balance in members	Contributors expertise distribution	How is the expertise of the OSSECO- community contributors?	High balanced is better A zero value for these indices implies a uniform distribution. A value of 1 means inequality.	[27]
Longevity of activity	Longevity of contributors activity	How many time is a developer in the OSSECO- community?	High is better	[27]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Balance in members	Contributors experience distribution	How is the experience of the OSSECO- community contributors?	High is better	[27]
Balance in members	Number of projects per contributor	Does each OSSECO-ecosystem contributor have a OSSECO-community project in which it contributes?	Close to 0 is better. A zero value for these indices implies a uniform distribution. A value of 1 means inequality	[27,51]
Activeness of members	Number of members making new features requests	How many members are making new features requests?	More is better	[27]
End-users interest	Number of downloads	Are the OSSECO- ecosystem projects popular?	More is better	[27,37]
Size of ecosystem (People)	Number of ecosystem-community mailing list subscribers.	What is the contributors OSSECO- community size?	More is better	[27]
Reputation of ecosystem	Number of scientific publications referencing the ecosystem-community	Are the OSSECO- community data sources used by researchers?	More is better	[27]
Reputation of ecosystem	Web and Social media hits.	Have the OSSECO- community projects visibility on the web?	More is better	[27]
Knowledge creation	Number of patents	How many patents has the community?	More is better	[27]
Reputation of ecosystem	Contributors ratings and reputation.	How is the OSSECO- ecosystem contributors' reputation? describing how well the developer is contributing and performing within the ecosystem.	High is better. High contributors ranking implies more Trustworthiness.	[27,37]
Commercial support	Community acceptance	How is the OSSECO- community acceptance by commercial organizations?	High is better. OSSECO-community acceptance is better	[27]
Reputation of ecosystem	Number of web pages referencing the ecosystem- community web page	How many page referenced the OSS-community web page?	More is better	[27]
Coherence of ecosystem	percentage of nodes with betweenness centrality	What is the ability of a node to act as a mediator in the community?	Closer to 1 is better	[27]
Coherence of ecosystem	Cluster of collaborating developers	How to identify clusters of developers in an OSS- community?	A cluster per project is better.	[27]
Coherence of ecosystem	Number of nodes to disconnect the ecosystem	How is the OSSECO- ecosystem connected?	More is better.	[27]
Coherence of ecosystem	Number of connections with partners	How many connections have the partners in the OSSECO network?	More is better. More number of connections implies more Interrelatedness.	[27]
Communication quality	Number of synonyms in vocabulary map	How to obtain a general overview of the domain language of the OSSECO?	A common language is better. Less synonyms is better	[27]
Communication quality	Positiveness of vocabulary in communications	How is the message vocabulary content in the OSSECO?	A positive vocabulary is better.	[27]
Balance in members	partners distribution in projects	Is the partner distribution over the ecosystem species equality?	Close to 0 is better A zero value for these indices implies a uniform distribution. A value of 1 means inequality	[27]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Diversity in artifacts	Number of context types of ecosystem projects applications	Have the OSSECO projects different types of context applications?	More is better. A wide variety of OSSECO project applications contexts, will be more supporting for niche creation.	[27,37]
Diversity in members	Number of natural languages supported	Is the OSSECO multi-language?	More is better. A wide variety of supported languages, will be more supporting for niche creation.	[27,37]
Diversity in artifacts	Variety in ecosystem projects technologies.	Does the OSSECO projects support different technologies?	wide variety of technologies, will be more supporting for niche creation	[5, 27, 37]
Diversity in artifacts	Number of ecosystem platform extensions	How many platform extensions have the OSSECO?	More is better. Each extension is a potential ecosystem niche	[27]
Diversity in artifacts	Number of markets in which the ecosystem is participating	Are the OSS-ecosystem projects in multiples markets?	More is better. A wide variety of markets, will be more supporting for niche creation	[27]
Knowledge creation	Number of knowledge artifacts	Are the contributors adding knowledge to the OSSECO?	More is better	[27]
Financial wellness	Market share	How is the OSSECO projects market shared? The percentage market cap for the currency under investigation of the total market cap of cryptocurrencies?	More is better.	[10,37, 67,69]
Amount of communication	Number of new communities	Are the OSSECO creating new communities continuously?	More is better.	[27]
Activeness of members	Active developers of unique repositories in the past year	Not all repositories are created equal. Many are copies of original projects or slight derivates. This sub-indicator only takes into account non-fork repositories, i.e. unique projects that were started from scratch instead of based on another project.	More is better.	[43]
Activeness of members	Active developers per segment of time	The week to week variety in number of actively contributing developers measures group activity in time between releases or cycle time	More is better.	[43]
Size of ecosystem (Artifacts)	Number of Total repositories	Repositories are a direct indicator of the number of spin offs for a PaaS.	More is better	[43]
Size of ecosystem (Artifacts)	Total number of followers	Followers are GitHub members who have starred a repository, a mechanism that is used to be kept up-to-date with changes. A follower does not actively contribute to development. Thus a follower is an indicator of passive interest in the project.	More is better	[43]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Amount of activity on artifacts	Number of Unique repositories	Popular repositories on GitHub have many forks, which are often direct copies of the original repository. Multi-platform repositories that are exceptionally popular increase the total number of repositories for all PaaS providers, although one or more of these might not be that popular at all. To discount this phenomenon, this sub-indicator is calculated as the total number of original repositories.	More is better	[43]
Diversity in artifacts	Number of unique programming languages	The diversity of programming languages that are used to develop for a PaaS indicate how broad of an interest there is in leveraging the strength of that PaaS.	More is better	[43]
Longevity of activity	Number of repositories updated at least once	Many repositories and forks of repositories are created but never updated. Counting the number of repositories that are updated at least once after creation provides an indication of the interest longevity of developers in the PaaS.	More is better	[43]
Amount of activity on artifacts	number of lines of code added or changed or removed over a time period	The Productivity metrics on the project level are the metrics that indicate how much the projects contribute to the total ecosystem. The most important metric is typically the number of lines of code added or changed over a time period, as this is a prime indicator for the activity of contributors in the ecosystem.	More is better. the number of lines of code added or changed over a time period, is a prime indicator for the activity of contributors in the ecosystem	[37,55]
Activeness in maintenance	Number of new tickets	Number of new tickets indicate how quickly problems are reported and resolved in the ecosystem.	More is better. Number of new tickets indicate how quickly problems are reported and resolved in the ecosystem.	[37]
Commercial support	Commercial patronage	How much is acceptance by commercial organizations? How much is the support a cryptocurrency receives from other companies?	More is better. commercial patronage and capital contributions and donations are indicators of acceptance by commercial organizations, also showing that the project will probably not go away soon.	[10,37]
Commercial support	Capital contributions and donations	What is the amount of money that a cryptocurrency has received other than investments in the currency?	More is better. capital contributions and donations are indicators of acceptance by commercial organizations, also showing that the project will probably not go away soon.	[10,37]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Activeness of members	Active contributors	What is the number of people that have made a contribution to the source code since the second half of 2016 and that have over 50 commits since the creation of the cryptocurrency?	More is better. The Robustness metrics on the project level indicate how well the project deals with change. Typically, the best defense for a project to survive a big change is in numbers: a large number of active developers, for instance, is indicative of a robust project, and the more robust projects there are in an ecosystem, the healthier it can be considered	[10,37]
End-users interest	Number of page views of the official website(s)	What is the amount of page views of the official website(s)?	More is better.	[10,37]
Size of ecosystem (People)	User satisfaction or ratings	Not provided	More is better. User satisfaction is a clear indicator that tells whether a project is robust or not.	[37]
Survival in ecosystem	Leavers per month	How many leavers does the E-commerce ecosystem have per month?	WooCommerce is leading the largest drop, perhaps due to the novelty of the platform.	[2]
Size of ecosystem (People)	Number of unique developers	How many unique developers does the E-commerce ecosystem have?	More is better.	[2]
Amount of activity on artifacts	Up-to-dateness of modules	What Percentage of modules have been updated in 2012?	More Percentage is better.	[67]
Amount of activity on artifacts	Findability of the ecosystem	How much is the Findability of the ecosystem?	operationalize a more effective SEO strategy is better. Less cases of unfindable modules is better. More findability is better.	[67]
Amount of communication	Level of contribution per community user	What is the Average number of posts and topics per member in platform's forum communities?	More is better	[67]
Amount of activity on artifacts	number of commits per day	What is the number of commits per day in open source ecosystem?	Not provided. Passive users are essential for a sustainable OSSECO-community	[55]
Amount of activity on artifacts	files changed per day	How many files are changed per day in open source software ecosystem?	Not provided	[12,27, 55]
Amount of activity on artifacts	average files changed or added or removed per commit	What is the average number of files changed per commit?	Not provided	[55]
Activeness in maintenance	mean time to repair a bug	How much is the mean time to repair a bug?	Not provided	[37,55]
Amount of communication	volume of messages per day	How much is the volume of messages per day?	Not provided	[55]
Size of ecosystem (People)	average time to resolve a critical bug	What is the average time to resolve a critical bug?	Not provided	[55]
Activeness in maintenance	number of open problem	How many open problems are there in the open source community?	Not provided	[63]
Activeness in maintenance	frequency of the contributions to the bug report system	How frequent is the contributions to the bug report system?	Not provided	[63]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Longevity of activity	first and last contributions of each individual developer	When is the first and last contributions of each individual developer over the project's history?	A regenerating community can be considered more sustainable since it is more likely to survive the lost of important contributors over time.	[63]
Activeness of members	amount of time developers are able and willing to contribute to the development effort	How much time are the developers able and willing to contribute to the development effort?	Not provided	[63]
Amount of communication	activity of mailing lists and discussion forums	How many messages are posted to project mailing lists and how many responses are obtained from those messages?	Not provided	[27,63]
Size of ecosystem (People)	Number of the registered users	How many are the registered users? (Unit: ten thousands)	More is better.	[69]
Size of ecosystem (People)	Number of open source code categories	How many are the open source code categories?	More is better.	[69]
Activeness of members	Number of users log in at least once a week	How many users log in at least once a week?	More is better.	[69]
Openness of ecosystem	Open source code usage	How much is Open source code usage?	More is better.	[69]
Security and privacy	Ability to resist viruses	How much is the ability to resist viruses? i.e, whether being attacked by a virus in using	More is better.	[69]
Ease of use	Ease of use	How easy is it for users to use the system?	More is better.	[69]
Connection with other entities	proportion of subsystems in the system solved by third parties	What is the proprtion of subsystems in the system that are solved by third parties	More is better.	[69]
Openness of ecosystem	openness of ecosystem for users to freely contribute	To meet the individual needs of users, how much software is open to the user to freely combine and how satisfied users are.	More is better.	[69]
Coherence of ecosystem	the degree of cooperation between different development groups	How much is the degree of cooperation between different development groups?	More is better.	[69]
Process maturity	Process maturity based on developers opinion	How much is the process maturity and the satisfaction of collaboration?	More is better.	[69]
Process maturity	Democratic decision making	In the development process, when solving problems, what is the level of brain storming?	More is better.	[69]
Activeness in maintenance	time for developers to feedback the problems (feedback cycle)	How much is the time for developers to feedback the problem	The feedback cycle represents the time for developers to feedback the problem. The shorter the time, the more developers are actively maintaining the stability of system, and the healthier the system.	[69]
Activeness of members	users with at least one product actively running	What is the percentage of users with at least one AV product actively running with up to date signatures?	Not provided	[42]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Amount of activity on artifacts	percentage of actively running products of the ecosystem	What is the percentage of actively running AV products using the latest signature files available?	Not provided	[42]
End-users interest	number of end users of the released applications	What is the number of end users of the released applications?	More is better. The usage of a project is also highly indicative of how important a project is to its end-users, and can be used as a predictor of robustness.	[10,68]
Size of ecosystem (Artifacts)	number of commits to the software framework	What is the number of commits to the software framework?	Not provided	[24,68]
Amount of activity on artifacts	Number of active projects	What is the number of active projects? how many projects are still being updated?	More is better. Number of active projects measures the robustness of an ecosystem because it shows what projects are still being updated while the ecosystem undergoes change.	[24,37, 68]
Satisfaction	Contributor satisfaction	How much is the satisfaction of the developers of applications in the ecosystem?	A high developer satisfaction binds developers to the ecosystem. Therefore, contributor satisfaction is an indicator of robustness.	[68]
End-users interest	End user rating	How is the end users ratings?	A high rating from end users is essential because they have to use the developed applications and participate in research to allow this ecosystem to function. Therefore, the end user rating is an indicator of the robustness of the ecosystem.	[68]
Connection with other entities	Outbound links to other ecosystems	What are the other ecosystems where the contributors are active in?	The multi-homing activities of developers may or may not be beneficial for the robustness of the ecosystem.	[68]
Diversity in members	Variety in developer type	Variety in developer type discusses who the contributors to this ecosystem are.	Their size and location may also influence ecosystem health. Iansiti and Levien state that a healthy ecosystem possesses the capabilities to increase meaningful diversity over time through the creation of new valuable functions. In this article, it is argued that both variety in projects and developer type lead to an increased capability to create meaningful diversity in the software ecosystem.	[68]
Size of ecosystem (People)	number of new projects	how many related GitHub projects, such as plug-ins or wallets are released from January 1st, 2017 till December 15th, 2017? What is evolution of the number of new projects belonging to an ecosystem over time?	More is better. This number contribute to the productivity of the ecosystem. [5] New projects measures the productivity of an ecosystem because it indicates the rate of growth over time.	[5]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Amount of communication	number of questions with the tag of the ecosystem attached in StackOverflow	What is the number of questions with the tag of the cryptocurrency attached?	More is better. When developers can ask questions on knowledge bases (e.g. Stack Overflow), the ecosystem will benefit from it by the fact that there is a community within an ecosystem where people help each other with questions regarding the development	[5]
Size of ecosystem (Artifacts)	number of APIs for the ecosystem	What is the number of distinct APIs for the ecosystem?	More is better. APIs make it possible to connect software within or even outside an ecosystem, which enables better communication between clients and the ecosystem.	[5]
Activeness of members	Number of active developers	What is the number of active developers?	The number of active developers shows how dependent an ecosystem is on individual developers. A higher number of active developers shows that the ecosystem is relatively more robust.	[5]
End-users interest	end user Interest in Search Statistics	How often a Google search is performed?	This metric shows which ecosystem was the most popular search term on Google and which ecosystem has the most interest over time. Van Lingen et al. defines the findability on Google an indicator of ecosystem health, but only if the ecosystems are compared with each other. This data cannot be used to compare with the exchange rate, because this data is relative, but it can be used to compare the cryptocurrencies among themselves.	[5, 10, 24, 37]
Open source ecosystem	Number of forks	What is the number of times repositories are forked?	More is better. Number of forks measures productivity because it indicates developer, in the sense that developers are experimenting with a repository or that developers are interested in the continued existence of the repository.	[24,43]
Receptiveness to sub-ecosystems	Number of new sub-ecosystems	How many sub-ecosystems in the super-ecosystem have been created in the past time period?	This metric is an indicator for both super- and sub-ecosystem health. For super-ecosystems, this metric is an indicator of how productive the ecosystem is in terms of creating new sub-ecosystems, which can help grow a superecosystem. For sub-ecosystem health, this is an indicator of how receptive a super-ecosystem is to sub-ecosystems, if this number is very small for a particular super-ecosystem, then they are likely not receptive to sub-ecosystems and that will make it hard to create a healthy sub-ecosystem in that super-ecosystem.	[51]
Connection with other entities	number of intersecting sub-ecosystems	What is the number of intersecting sub-ecosystems?	this is another indicator for the connectedness to other super-ecosystems similar to the number of outbound links to other SECO. Similar as for that metric, intersecting ecosystems, which, as discussed in section 5, are sub-ecosystems that exist in multiple super-ecosystems, can better withstand negative events in one super-ecosystem because they can get revenue or users from other super-ecosystems. This can also help the effected super-ecosystem, as these sub-ecosystems are a more stable factor for that super-ecosystem to rely on.	[51]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Receptiveness to sub-ecosystems	average size of sub-ecosystems	What is the average size of sub-ecosystems?	average size of sub-ecosystems and number of active sub-ecosystems, which for sub-ecosystem health further indicate the receptiveness of the ecosystem to new sub-ecosystems. The total number gives an overview of the receptiveness and the average size is an indicator of how well sub-ecosystems can grow inside the super-ecosystem and if the super-ecosystem is receptive to larger sub-ecosystems. Moreover, the super-ecosystem health is positively affected by larger sub-ecosystems, as they are likely to bring more users from outside to the super-ecosystem.	[51]
Receptiveness to sub-ecosystems	number of active sub-ecosystems	What is the number of active sub-ecosystems?	average size of sub-ecosystems and number of active sub-ecosystems, which for sub-ecosystem health further indicate the receptiveness of the ecosystem to new sub-ecosystems. The total number gives an overview of the receptiveness and the average size is an indicator of how well sub-ecosystems can grow inside the super-ecosystem and if the super-ecosystem is receptive to larger sub-ecosystems. Moreover, the super-ecosystem health is positively affected by larger sub-ecosystems, as they are likely to bring more users from outside to the super-ecosystem.	[51]
Receptiveness to sub-ecosystems	variety in sub-ecosystems	What is variety in sub-ecosystems?	this variety metric can help differentiate between types of projects more easily, as sub-ecosystems could be built around one specific type of project for the variety in project metric. Additionally, this metric can show the ability to create niche sub-ecosystems inside a super-ecosystem, which impacts the health of sub-ecosystems as well.	[51]
Financial wellness	Profit of the ecosystem	What is the expected profit of the ecosystem?	Not provided	[50]
Financial wellness	Revenue increase	What is the growth of ecosystem profits?	Not provided	[50]
Size of ecosystem (People)	Employees satisfaction rate	How much is the employees satisfaction rate?	Not provided	[50]
Size of ecosystem (People)	Employee turnover	How much is employee turnover?	Not provided	[50]
Size of ecosystem (People)	Customer satisfaction percentage	What percentage of customers are satisfied?	Not provided	[50]
Satisfaction	Total number of customer complaints	What is the total number of customer complaints?	Not provided	[50]
Expenses	Quality management cost	What is the cost of Quality management?	Not provided	[50]
Expenses	Rework cost	What is the cost of rework?	Not provided	[50]
Expenses	Marketing cost	What is the cost of marketing?	Not provided	[50]
Expenses	Internal quality control cost	What is the cost of internal quality control?	Not provided	[50]

Continued on next page

Related sub-category	Name of metric	Definition of metric	Interpretation of metric	Study
Expenses	Research and development cost	How much is the Amount spending on technology education?	Not provided	[50]
Activeness in maintenance	Upgrade and change cost	How much is the cost of change and upgrade?	Not provided	[50]
Size of ecosystem (Artifacts)	Number of available apps	How many apps are available in the ecosystem?	More is better	[11]
Financial wellness	Developer Revenue Share	What percent is the Developer Revenue Share?	Not provided	[11]
Amount of activity on artifacts	Number of new Apps	What is the number of new apps (Dec. 09 – Jan 10)?	More is better.	[11]
Process maturity	Reviewing and testing Submissions	Is Submissions reviewed and tested?	Not provided	[11]
Knowledge creation	scientific publications generated by the community	How many scientific publications were generated by the community?	Not provided	[12]
Diversity in artifacts	Variety in supporting hardware devices	How many hardware devices support the platform?	Not provided	[12]
Activeness in maintenance	Added KLOC in the last 30 days	How many KLOC were added in the repository in the last 30 days?	Not provided	[12]
Amount of activity on artifacts	Added artifacts in the last 30 days	How many artifacts were added in the repository in the last 30 days?	Not provided	[12]
End-users interest	Recent users in the last 30 days	How many users have used the platform in the last 30 days?	Not provided	[12]
Ease of use	Glossary of terms	Does the platform have a glossary of terms to be used in the projects?	Not provided	[12]
Ease of use	Documentation of the platform	Does the platform have documentation?	Not provided	[12]
Modularity	Number of modules shared and reused by partners	What is the number of modules shared and reused by partners?	Not provided	[62]
Modularity	Number of modules developed by partners	What is the number of modules developed by partners?	Not provided	[62]

Table B.1: All the 142 metrics I included in my taxonomy (with some of their related attributes)

Appendix C

List of Excluded Metrics from My Taxonomy

Here, I present all the 79 metrics I excluded from my taxonomy (with their related attributes). These excluded metrics were duplicated, did not have a clear implementation, or were only usable for a specific software ecosystem and not usable for other software ecosystems.

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
Number of changed files	How is the activity in the OSSECO- community repositories?	More is better High number of files changed implies more OSS-community activity and effort.	[27]	Duplicated with more specific metrics like "files changed per day"
Communication activity in the e-mail system	How is the communication between OSSECO-community members?	More is better High communications activities implies a high active OSSECO-community	[27]	Duplicated with more specific metrics like "activity of mailing lists and discussion forums"
Decline point	When the values of number of emails starts to decline?	Not provided	[27]	No clear implementation
Variety of ecosystem-community partners	How many kinds of partners has the OSSECO- community?	High is better Covariance with market indicates the variety of different partners a partner has.	[27]	No clear implementation
Active developers in the past year	The total number of active developers developing on top of a PaaS provides a direct measure of developer interest in the past year.	More is better.	[43]	Duplicated with more specific metrics like "Active developers of unique repositories in the past year"
Number of Forks	In theory, total repositories includes forks, but in practice this data is incomplete because of two factors. First, GitHub provides private repositories to premium members; effectively shielding us from accessing that data. Second, some repositories are forked but deleted sometime later. Luckily, for each repository GitHub counts the number of forks made regardless of these restrictions. For each PaaS the sum of all forks is taken to include potentially lost data.	More is better.	[43]	Duplicated with another metric with the same name [24]

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
Bug fix time	bug fix time indicates how quickly problems are resolved in the ecosystem.	Less is better. bug fix time indicates how quickly problems are resolved in the ecosystem.	[37]	Duplicated with another metric: "mean time to repair a bug"
Perceived ecosystem health	S6 described and measured a number of ecosystem health metrics. The outcome of these measurements serves as a factual, raw data dependent mean to measure the given ecosystems' health. In order to compare these findings to how a number of stakeholders (n=23) perceive the ecosystem's health, they carried out a brief survey.	Not provided	[67]	No clear implementation
Growth of the platform in number of developers	How rapid is the growth in the number of developers in the platform?	Faster growth and larger number of contributors is better.	[67]	No clear implementation
lines added per day	How many lines are added per day in open source software ecosystem?	Not provided	[55]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period"
lines removed per day	How many lines are removed per day in open source software ecosystem?	Not provided	[55]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period"
average lines added per commit	What is the average number of lines added per commit?	Not provided	[55]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period"

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
average lines removed per commit	What is the average number of lines removed per commit?	Not provided	[55]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period"
project code size	Not provided	Not provided	[63]	No clear implementation
Certificate and issuance	How is User's comprehensive evaluation of software solutions released by developers?	More is better.	[69]	No clear implementation
Out of date	What is the percentage of actively running antivirus products using out of date signatures?	Not provided	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems
Expired	What is the percentage of actively running expired antivirus products?	Not provided	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems
Snoozed	What is the percentage of AV products that are active but are not performing real-time monitoring?	Not provided	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems
off	What is the percentage of antivirus products that are not running, as they have been turned off?	Not provided	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems
richness	What is the total number of antivirus vendors within the ecosystem?	Not provided	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
concentration	The degree of concentration (D), also known as Simpson's diversity Index or Gini-Simpson Index, is a measure of the degree of concentration of individuals classified into types.	It can be interpreted as the probability that two organisms belong to different species. A value of 0 indicates no diversity and 1.0 indicates high diversity.	[42]	No clear implementation
Dominance	This index estimates dominance using the prevalence of the most abundant type, which refers to the AV vendor with the highest market share.	dominance of the AV ecosystem varies between 0.136 and 0.143. In economics, that would indicate that the AV market has a low concentration (¡0.5), ranging from perfect competition to an oligopoly.	[42]	No clear implementation
changes in antivirus status	What is the percentage of users by status that have a different antivirus status compared to the previous month?	Higher stability, both in AV status and AV vendor, is significantly associated with lower infection rates for all users, whether having an AV product installed or not.	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems
changes in antivirus vendor	What is the percentage of users that have a different antivirus vendor compared to the previous month?	Higher stability, both in AV status and AV vendor, is significantly associated with lower infection rates for all users, whether having an AV product installed or not.	[42]	Only usable for antivirus ecosystems and not usable for other software ecosystems

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
knowledge creation	The pilot interview aimed to shed light on metrics of ecosystem health that are related to the third-party developers. Therefore, the metric knowledge creation was added.	Not provided	[68]	No clear implementation
Variety in projects	Variety in projects measures what the goals and features are of the applications in the ecosystem	Their size and location may also influence ecosystem health. Iansiti and Levien state that a healthy ecosystem possesses the capabilities to increase meaningful diversity over time through the creation of new valuable functions. In this article, it is argued that both variety in projects and developer type lead to an increased capability to create meaningful diversity in the software ecosystem.	[68]	No clear implementation
average lines removed per commit	What is the average number of lines removed per commit?	Not provided	[55]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period
Lines of code added over time	Not provided	More is better. The number of lines of code added over time tells something about how productive the development of an ecosystem is.	[5]	Duplicated with another metric: "number of lines of code added or changed or removed over a time period"
Spin offs and forks	Not provided	Not provided	[5]	No clear implementation
Number of tickets	Not provided	Not provided	[5]	No clear implementation
Number of patents	Not provided	Not provided	[5]	No clear implementation

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
Potential for niche creation	in which possible areas can the SECO support niche creation?	More is better .Both measure niche creation, because if there is a plentiful amount of either potential or actual niche creation, then there is opportunity for a niche player to survive in the ecosystem.	[24]	No clear implementation
Actual niche creation	in which areas is the niche creation actually occurring?	More is better . Both measure niche creation, because if there is a plentiful amount of either potential or actual niche creation, then there is opportunity for a niche player to survive in the ecosystem.	[24]	No clear implementation
new related projects	How many are the new related projects?	In the productivity column there are the new related projects and the added knowledge about the sub-ecosystem. Both metrics are inspired by similar metrics in the network level, however, they are now applied on the sub-ecosystem. They are, therefore, also relevant to the health of the sub-ecosystem in the same way they are relevant to the health of the super-ecosystem on the network level. These indicators for the productivity in the sub-ecosystem also impact the health of the super-ecosystem, because productive sub-ecosystems bring more projects to the super-ecosystem as well.	[51]	Duplicated with another metric: Number of projects per contributor
added knowledge about the sub-ecosystem	How much is the added knowledge about the sub-ecosystem?	In the productivity column there are the new related projects and the added knowledge about the sub-ecosystem. Both metrics are inspired by similar metrics in the network level, however, they are now applied on the sub-ecosystem. They are, therefore, also relevant to the health of the sub-ecosystem in the same way they are relevant to the health of the super-ecosystem on the network level. These indicators for the productivity in the sub-ecosystem also impact the health of the super-ecosystem, because productive sub-ecosystems bring more projects to the super-ecosystem as well.	[51]	No clear implementation
Variety in sub-ecosystems	How much the sub-ecosystems are varied?	a more varied sub-ecosystem will also create more variety in the super-ecosystem impacting the health of the super-ecosystem in a similarly positive way.	[51]	No clear implementation

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
Absenteeism rate	Not provided	Not provided	[50]	No clear implementation
Remark number	Not provided	Not provided	[50]	No clear implementation
Rework percentage	Not provided	Not provided	[50]	No clear implementation
Number of audits	Not provided	Not provided	[50]	No clear implementation
Project deviation deadline	Not provided	Not provided	[50]	No clear implementation
Relationship between company number	Not provided	Not provided	[50]	No clear implementation
Number of participation in clusters	Not provided	Not provided	[50]	No clear implementation
Training value	How much is the overall value of training?	Not provided	[50]	No clear implementation
New technology value	How much is the overall value of new options created?	Not provided	[50]	No clear implementation
Patent number	Not provided	Not provided	[50]	No clear implementation
Free Apps	Does the ecosystem provide any free apps?	Not provided	[11]	No clear implementation
Paid Apps	Does the ecosystem provide any paid apps?	Not provided	[11]	No clear implementation
No. Downloads	Not provided	Not provided	[11]	No clear implementation
Registration Fee	Not provided	Not provided	[11]	No clear implementation
Potential Customer Base	Not provided	Not provided	[11]	No clear implementation
Apps Available outside Store	Are the apps available outside the store?	Not provided	[11]	No clear implementation
Payment Methods	What is the payment method?	Not provided	[11]	No clear implementation
Subscription Billing Allowed	Is the Subscription Billing Allowed?	Not provided	[11]	No clear implementation
Minimum App Cost (Ex. Free)	How much is the minimum app cost?	Not provided	[11]	No clear implementation
Maximum App Cost	How much is the maximum app cost?	Not provided	[11]	No clear implementation
Certification Requirement	What is the Certification Requirement?	Not provided	[11]	No clear implementation
Software Platform	Not provided	Not provided	[11]	No clear implementation

Continued on next page

Name of metric	Definition of metric	Interpretation of metric	Study	Reason to be excluded
semantic closeness between services	Not provided	Not provided	[12]	No clear implementation
types of nodes in the SECO peer-to-peer network	Which types of nodes are in the SECO peer-to-peer network?	Not provided	[12]	No clear implementation
Patent number	What is the number of patents?	Not provided	[12]	No clear implementation
Releases with participation of all developers in each project	In how many releases did each developer participate in each project?	Not provided	[12]	No clear implementation
Releases with participation of all developers	In how many total releases did each developer participate?	Not provided	[12]	No clear implementation
Modified files by each developers	How many files are modified by each developer in a month?	Not provided	[12]	Duplicated with another metric: "files changed per day"
Types of modified files by each developers	Which types of files are modified for each developer in a month?	Not provided	[12]	No clear implementation
Events with participation of	In what events did each community member participate?	Not provided	[12]	No clear implementation
People in each project event	How many people are in each project event?	Not provided	[12]	No clear implementation
jobs advertisements	How many jobs advertisements were divulged?	Not provided	[12]	No clear implementation
Number of readers in the community	How many readers does the community have?	Not provided	[12]	No clear implementation
Web page requests	How many web page requests were received?	Not provided	[12]	No clear implementation
Number of users' groups	How many users' groups exist in the community?	Not provided	[12]	No clear implementation
collapse plan	Is there a collapse plan for the SECO?	Not provided	[12]	No clear implementation
Number of users	Not provided	Not provided	[5]	No clear implementation

Continued on next page

Name of metric	**Definition of metric**	**Interpretation of metric**	**Study**	**Reason to be excluded**
Number of transactions	Not provided	The number of transactions (10) is measured over time to see if the ecosystem still grows.	[5]	No clear implementation
Change in Marketcap	What is the percentage of change in marketcap for the cryptocurrency since June 2017?	Not provided	[10]	No clear implementation
Variation in Project Applications	What is the number of ways a cryptocurrency is applied in society?	Not provided	[10]	Only usable for cryptocurrency ecosystems and not usable for other software ecosystems
Commits for projects	What is the number of times a project has been changed by a developer?	More is better. Commits measure productivity because the metric indicates which projects are often changed.	[24]	Duplicated with another metric: "number of commits to the software framework"
Culminating point	When the number of emails is maximum in the OSS-community history?	Not provided	[27]	No clear implementation

Table C.1: All the 79 metrics I excluded from my taxonomy (with some of their related attributes)

Appendix D

List of Included Practices in My Taxonomy

Here, I presents all the 174 practices I included in my taxonomy (with their related attributes)

Related sub-category	Name of practice	Key area	Source
Welcome newcomers to the ecosystem	Provide a newcomer-specific page or portal guiding their first steps, including architectural information	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Update and release new versions	Identify and dismiss outdated information on websites	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Welcome newcomers to the ecosystem	Provide generation of (semi-)automated documentation filtered to up-to-date information relevant to newcomers	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Managing mailing lists	Answer questions on the mailing list quickly	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Welcome newcomers to the ecosystem	Establish mentoring by designating a few experienced members to communicate with new members	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Problem management	Create a detailed step-by-step tutorial linking information about common problems and possible solutions	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Documentation	Provide updated documentation about code's organizational structure, and how the components, modules, classes, and packages are related to each other	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Provide physical and digital infrastructure	Support the participation of the ecosystem on aggregators' sites such as stack overflow, reddit, hacker news, and so on	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Documentation	Keep documentation organized and indexed providing a level of difficultly from beginners to experts	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Welcome newcomers to the ecosystem	Request newcomers to write informative subjective lines when desiring an answer of the community. They should post direct-to-the-point questions	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Welcome newcomers to the ecosystem	Provide a dictionary to newcomers to facilitate their learning of the technical jargon, acronyms of the community	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Update and release new versions	Identify and dismiss outdated information on websites	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Welcome newcomers to the ecosystem	Provide visual information such as the class diagrams or dependency views, and code visualization tools	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Member training	Provide video-classes about introducing the ecosystem, installing technologies, configuring the development environment, and using the dependencies with other ecosystems	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Managing requirements for applications	Provide a manifest explaining requirements of an application belonging to the ecosystem	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Community building and collaboration	Provide video-conference sessions with questions and answers about relevant topics	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Translate artifacts	Provide information about the translation process (how to participate and tools used)	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Translate artifacts	Provide information about how to report translation mistakes	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Documentation	Provide documentation about translation rules that developers must follow to prepare code for other languages	Architectural Knowledge Management (AK)- related to managing all learning about the software architecture	[20]
Welcome newcomers to the ecosystem	Point newcomers to easy tasks filtered by dificulty, skills needed, and topics	External Management (EM) -related to managing the interfaces third-party developers	[20]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Articulation work	Keep the list of tasks updated informing about who is working on the solution	External Management (EM) -related to managing the interfaces third-party developers	[20]
Avoid misunderstanding	Setting a code of conduct to avoid mistreatment among members	External Management (EM) -related to managing the interfaces third-party developers	[20]
Use tools	Provide a virtual machine with pre-configured build environments, web-based IDEs, or a container management tool	External Management (EM) -related to managing the interfaces third-party developers	[20]
Welcome newcomers to the ecosystem	Inform newcomers about technical background required. Identify which specific technologies they need to know or they should learn to achieve their goal of contributing to the ecosystem	External Management (EM) -related to managing the interfaces third-party developers	[20]
Articulation work	Tag all tasks in accordance with degree of difficulty (easy, medium, difficult)	External Management (EM) -related to managing the interfaces third-party developers	[20]
Welcome newcomers to the ecosystem	Provide a message template for newcomers to use to interact with the community	External Management (EM) -related to managing the interfaces third-party developers	[20]
Welcome newcomers to the ecosystem	Provide clear directions to enable newcomers to find the artifacts that need to be changed	Changes Management (CM) - related to maintaining the changes and reducing the impact caused by the changes on the software architecture.	[20]
Dependency management	Provide information about dependencies among artifacts	Changes Management (CM) - related to maintaining the changes and reducing the impact caused by the changes on the software architecture.	[20]
Automate tasks	Provide unit test automation for main functionalities	Quality Management (QM) - refers to managing the quality of the various structures designed into the architecture	[20]
Automate tasks	Provide an automated process to launch release of applications	Quality Management (QM) - refers to managing the quality of the various structures designed into the architecture	[20]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Provide physical and digital infrastructure	Use of repository hosting services	Not provided	[18]
Standardize procedures	Use of examples of code	Not provided	[18]
Provide physical and digital infrastructure	Use of tutorials and wizards	Not provided	[18]
Provide physical and digital infrastructure	Use of website and wikis	Not provided	[18]
Managing mailing lists	Use of mailing lists	Not provided	[18]
Use of APIs	Use of Applications Programming Interfaces (API)	Not provided	[18]
Dependency management	Provide rules for dependencies	Not provided	[18]
Dependency management	Use of metadata with dependencies of the system	Not provided	[18]
Use tools	Use of tools to register requirements	Not provided	[18]
Standardize procedures	Define regulations and process	Not provided	[18]
Resolve financial and marketing issues	Existence of groups to lead with financial and business issues	Not provided	[18]
Avoid misunderstanding	Existence of specific committee to lead with exception cases of conflicts	Not provided	[18]
Inform changes and decisions to community	Discuss changes and announced broadly to community	Not provided	[18]
Version management	Use of agile practices (continuous delivery, building small versions, frequent releases, and daily meetings)	Not provided	[18]
Code assessment	Perform code inspections	Not provided	[18]
Security	Existence of a security committee	Not provided	[18]
Architecture design	Remove code based on experience of the architect	Not provided	[18]
Use tools	Use of several tools	Not provided	[18]
Managing the responsibilities	Create confidence in a member, based on his work history. Based on this, provide different levels of responsibilities.	Not provided	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Managing the responsibilities	Require that at least two persons nominate a member to be promoted up the levels of responsibilities.	Social (related to working together in the community)	[16]
Community building and collaboration	Conduct annual meetings among translators by projects.	Social (related to working together in the community)	[16]
Resolve financial and marketing issues	Promote the networking in the work Market.	Social (related to working together in the community)	[16]
Promote social relationships with members in other countries	Promote the social relationships to members in other countries.	Social (related to working together in the community)	[16]
Promote social relationships with members in other countries	Promote the networking in the work Market.	Social (related to working together in the community)	[16]
Promote social relationships with members in other countries	Create opportunity to practice another human language such as English.	Social (related to working together in the community)	[16]
Community building and collaboration	Conduct a general annual meeting to discuss face-to-face several questions of the community.	Social (related to working together in the community)	[16]
Managing the responsibilities	Perform elections to assign responsibility levels for new members.	Social (related to working together in the community)	[16]
Member training	Teach experienced members how to use tools to be more productive. For example, the use of shortcut keys, scripts, and plug-ins.	Social (related to working together in the community)	[16]
Community building and collaboration	Active members can be given trips to events sponsored by the ecosystem.	Social (related to working together in the community)	[16]
Managing mailing lists	Use tools to support communication in the group such as: mailing lists, forums, IRCs, wikis, blogs to communication.	Social (related to working together in the community)	[16]
Community building and collaboration	Promote Happy Hours and/or dinners to facilitate the social integration during events.	Social (related to working together in the community)	[16]
Keep the Schedule of members flexible	Provide flexibility in the translation schedule and negotiate deadlines.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Avoid misunderstanding	Reach an agreement in the community to face questions that are not registered in the tutorials or guidelines.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Avoid misunderstanding	Reach an agreement in the community to face questions that are not registered in the tutorials or guidelines.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing mailing lists	Make decisions based on discussions in the mailing list.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Translate artifacts	Provides lectures to teach how to translate and how to become a contributor.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Resolve financial and marketing issues	Provide a nonprofit corporation to manage legal and financial issues.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing the responsibilities	Divide the activities into working groups responsible for several areas such as: marketing, infrastructure, design, community in order to keep the group healthy.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing technical decisions	Make technical decisions independently inside each project.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing mailing lists	Make decisions based on discussions in the mailing list.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Resolve financial and marketing issues	Attract companies that invest money to support the ecosystem.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Keep the Schedule of members flexible	Provide flexibility in the time to work with better code quality.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Apply continuous improvement	Use lessons learned by past mistakes to modify and improve the practice.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Plan releases	Define a schedule of releases that will affect the work of the entire community.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Plan releases	Provide a feature plan that will be used for the implementation.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Inform changes and decisions to community	Provide an accountability report every three months to community.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Resolve financial and marketing issues	Define and support a special group to manage all business issues.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing the responsibilities	Define a hierarchical structure for members with different levels of responsibilities.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Managing technical decisions	Assign a lead maintainer for each project for making technical decisions.	Business (related to aspects of management, strategic planning, and innovation, and organized activities such as marketing, taking decisions, and so on.)	[16]
Articulation work	Each person chooses the work they desire to perform among available tasks in the ecosystem. For example: file to translate, code development, test of applications, and so on.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Code assessment	Review all code before accepting into the release.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Code assessment	Do not review translations of artifacts.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide guidelines for how to create the environment to work with translation and code development.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Control the translation work separating artifacts into two groups to translate: stable artifacts have experienced none or few changes and unstable - artifacts are continually being changed and checked constantly.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide specific tools to optimize the translation work. For example: the tool, called Localize, searches text to translate and save the specific format. It also suggests words to use in translation.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Articulation work	Each person chooses the work they desire to perform among available tasks in the ecosystem. For example: file to translate, code development, test of applications, and so on.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide different kinds of translation environments to work: online and offline.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide a freeze period to stabilize the translation of a version before the launch of a version.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Automate tasks	Provide tools for code optimization, perform static analysis of code, code review, and test automation.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Articulation work	Provide checklists of tasks to conclude the work. For example: before adding the code to the repository, the developer needs to document, test, integrate with other applications, code must be reviewed by another person and translation must be checked.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Automate tasks	Provide continuous integration tools that check code every day and report errors.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Standardize procedures	Provide a manifest which the project must follow to be considered part of the ecosystem.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide specific tools to optimize the translation work. For example: the tool, called Localize, searches text to translate and save the specific format. It also suggests words to use in translation.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Standardize procedures	Require that all infrastructure must be under the control of the ecosystem and be based on the technologies created by the ecosystem.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Standardize procedures	Provide a design style to be followed for all applications.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Translate artifacts	Provide a freeze period to stabilize the translation of a version before the launch of a version.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Version management	Conducting incremental transitions rather than versions and releases all at once.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Code assessment	Use scripts to do an initial code review and catch errors.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Maintaining compatibility	Execute the code with several different versions of operational system and compilers, keeping compatibility and avoiding errors.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Dependency management	Use scripts to evaluate improper dependencies.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Inform changes and decisions to community	Each member announces the feature that he will implement to avoid duplicated work.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Maintaining compatibility	Keep backward compatibility for a long time (around 6 years).	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Maintaining compatibility	Provide guidelines informing about actions that are allowed and not allowed to keep backward compatibility.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Inform changes and decisions to community	Communicate changes in the API to the community in advance.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Automate tasks	Generate code documentation automatically.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Documentation	Provide documentation about operation of projects such as wikis, user guidelines, and meetings event log.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Managing mailing lists	Use discussion mailing lists divided by different topics such as test, development, etc.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Code assessment	Use a group of members to perform tests in beta version.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Problem management	Use a bug triage task force to remove repeated or non-existent bugs.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Apply continuous improvement	Conduct and control refactoring in project as necessary.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Provide physical and digital infrastructure	Use technology to reuse GUI components to facilitate the development work.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Managing requirements for applications	Developers and End Users create requirements for core and applications.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Version management	Develop two versions in parallel, an old and a new version until that new version is stable and ready to replace the old version	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Provide physical and digital infrastructure	Different types of contributors help to test code that needs special hardware, operational system or network connection. This happens because normally a developer does not have all devices to test code.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Update and release new versions	Mark the code as deprecated in features that will be removed from the software.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]
Inform changes and decisions to community	Make strategy explicit	Not provided	[72]
Use of APIs	Create API	Not provided	[72]
Creating partnership	Co-development i.e. joint development projects with other companies	Not provided	[72]
Creating partnership	Create partnership model	Not provided	[72]
Resolve financial and marketing issues	Do marketing	Not provided	[72]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Creating partnership	Form alliances	Not provided	[72]
Use of APIs	Stabilize API	Not provided	[72]
Welcome newcomers to the ecosystem	CRaise entry barriers	Not provided	[72]
Creating partnership	Make partners explicit	Not provided	[72]
Member training	Propagate operation knowledge	Not provided	[72]
Resolve financial and marketing issues	Collaborative marketing	Not provided	[72]
Resolve financial and marketing issues	Create sales partner program and create new sales channels	Not provided	[72]
Code assessment	Peer review of changes(In some projects, changes proposed by developers with direct commit rights are also subject to review by other community members)	Not provided	[63]
Apply continuous improvement	Propose significant enhancements (Some projects have disciplined processes that allow community members to formally propose enhancements for discussion by the community.)	Not provided	[63]
Problem management	Report and handle issues with the product (For obvious reasons, this process is present in almost all Open Source projects in some form or another.)	Not provided	[63]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Code assessment	Test the program(s) produced by the project (Most projects doing repeatable testing do it by defining an automated test suite. If no test suite is available, there may be explicitly defined manual test cases, but this is much less likely to happen.)	Not provided	[63]
Plan releases	Plan releases (Either releases are done on a time-based fashion or based on a feature road map.)	Not provided	[63]
Version management	Release new versions of the product (Release processes in Open Source often include the creation of a number of alpha, beta and release-candidate versions that are delivered by the developers in order to obtain feedback from the community (active users of an OSS system are often willing to test these versions and report about problems they may find). Release processes also often include running a test suite or performing other forms of formal testing.)	Not provided	[63]
Version management	Backport corrections in the current release to previous stable releases (When a stable and an unstable (development) branch of a project are maintained simultaneously, so-called backports are often necessary that move corrections or selected improvements made to the development branch into the stable branch.)	Not provided	[63]
Member training	Architect leaders provide some tutorials and/or video courses for the technology available	Architectural Knowledge	[14]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Documentation	Build maps of modules and run-time elements during face-to-face meetings	Architectural Knowledge	[14]
Provide physical and digital infrastructure	Create personal blogs and/or wikis to inform about the development and architectural issues	Architectural Knowledge	[14]
Code assessment	During code review, provide feedback information about architecture, good practices to code, doing refactoring and show the best way to solve the problems	Architectural Knowledge	[14]
Use of APIs	Document APIs daily	Architectural Knowledge	[14]
Member training	Provide mentoring programs to training new developers	Architectural Knowledge	[14]
Member training	Keep different architects with different levels of specialty to spread the knowledge in the community	Architectural Knowledge	[14]
Member training	Provide code recommendations, defining a standard in the community	Architectural Knowledge	[14]
Welcome newcomers to the ecosystem	Provide a development manual for newcomers to know how to start contributing	Architectural Knowledge	[14]
Inform changes and decisions to community	Keep a register of meetings available to the community to know all decisions of the meeting	Architectural Knowledge	[14]
Standardize procedures	The organization board defines some technologies that should be used by the whole community as tools for testing, communication, coding review, bugging manager, and navigation	Choice of Technology	[14]
Standardize procedures	The project leaders define specific technology that impact into the work of the project community	Choice of Technology	[14]
Architecture design	The organization board recommends a tool to create UML diagrams of the architecture to be used by KDE projects	Choice of Technology	[14]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Standardize procedures	Define minimal quality criteria requirements to add an application to the ecosystem (documentation, automatized tests, dependence restrictions)	Quality Management	[14]
Code assessment	Define a team to test performance and behavior of the application	Quality Management	[14]
Use tools	Use some tools to compute some quality metrics	Quality Management	[14]
Maintaining compatibility	Keep the backward compatibility for a medium or long time to allow the community update their software	External Management	[14]
Security	Provide different levels of security access for the parts of the ecosystems in accordance with the degree of commitment and tasks in the ecosystem	External Management	[14]
Automate tasks	Use automatized tests to gather problems with the code	External Management	[14]
Security	Use tools to publish known cyber security vulnerabilities. For example the Common Vulnerabilities and Exposures (CVE®)	External Management	[14]
Security	Keep a team to manager the security problems registered	External Management	[14]
Security	Maintainers checks and removes malicious code	External Management	[14]
Avoid misunderstanding	Preference to use the English language written to avoid misunderstanding	External Management	[14]
Community building and collaboration	Do online meetings in a time zone adequate for most of the community	External Management	[14]
Community building and collaboration	Provide several online meetings to discuss architectural problems by IRC, email	External Management	[14]
Creating partnership	Create partnership with third-party to solve problems of the core and their interfaces	External Management	[14]

Continued on next page

Related sub-category	Name of practice	Key area	Source
Managing the responsibilities	Allocation of responsibilities in accordance with the interest of the developer/architect and discussing with the other members	Resources Management	[14]
Provide physical and digital infrastructure	The organization board provide hardware and software resources to be used by the community	Resources Management	[14]
Resolve financial and marketing issues	Define a financial board to manage the financial resources	Resources Management	[14]
Community building and collaboration	Provide financial resources to support meetings face-to-face	Resources Management	[14]
Architecture design	The architect/maintainer taken design decisions to guide the developers considering the communication mechanisms (between systems, between your system and external entities, between elements of your system)	Design-Making	[14]
Architecture design	Application architects should follow decisions of the core architects and add their decisions in agreement with their recommendations	Design-Making	[14]
Community building and collaboration	Provide meetings (sprints) face-to-face to accelerate the development of critical issues	Design-Making	[14]
Architecture design	Provide autonomy to work on the architecture in accordance with the degree of commitment and experience of work in the community	Design-Making	[14]
Inform changes and decisions to community	Create a demand register to the community know what are the next steps	Design-Making	[14]

Continued on next page

Related sub-category	**Name of practice**	**Key area**	**Source**
Dependency management	Divide the parts of the software in layers defining restrictions for managing dependencies among the layers	Change Management	[14]
Inform changes and decisions to community	Discuss with the community about critical changes into architecture that will impact in the applications	Change Management	[14]
Inform changes and decisions to community	Publish widely the architectural changes for the community	Change Management	[14]
Maintaining compatibility	Only architects' leaders take decision about remove obsolete code, respecting the backward compatibility	Change Management	[14]
Community building and collaboration	Provide specifics face-to-face meetings with teams of different applications to solve development problems with their interdependent modules	Change Management	[14]
Architecture design	Build the architecture based in plug-ins to facilitate the coupling of applications	Change Management	[14]

Table D.1: All the 174 practices I included in my taxonomy (with some of their related attributes)

Appendix E

List of Excluded Practices from My Taxonomy

Here, I presents all the 13 practices I excluded from my taxonomy (with their related attributes). These excluded practices were either duplicates or did not have a clear implementation.

Name of practice	**Key area**	**Source**	**Reason to be excluded**
Provide guidelines for how to translate artifacts.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]	Duplicate with practices in "Translate artifacts" subcategory
Provide different access levels to the KDE repository to check-in artifacts for release.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]	No clear implementation
Develop code to be extensible. The use of plug-ins and compilation is separated between the core and applications.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]	No clear implementation
Number versions with 3 numbers indicating what was changed in the version.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]	No clear implementation
Developers handle manually change management.	Technical (related to product development (core and applications), technologies used, code rules, among others)	[16]	No clear implementation
Expand applicability	Not provided	[72]	No clear implementation
Develop complementary platforms	Not provided	[72]	No clear implementation
Develop new business models	Not provided	[72]	No clear implementation
Organize developer days	Not provided	[72]	No clear implementation
Partner development programs	Not provided	[72]	No clear implementation
Change submission and review (Submit changes (e.g., defect corrections, enhancements) to the project for potential inclusion. Also, review changes submitted by community members.)	Not provided	[63]	No clear implementation
Projects of the applications define quality criteria in accordance with the organization board recommendations	Not provided	[14]	No clear implementation
Define recommendations to reuse all resources available and tested by community	Design-Making	[14]	No clear implementation

Table E.1: All the 13 practices I excluded from my taxonomy (with some of their related attributes)