A Systematic Mapping Study of Empirical Research Methods in Software Ecosystems

Abdullai Larry, Shamshiri Hatef, Mahmud Hasan, Hamza Muhammad, Aittamaa Essi, Vuolasto Jaakko, Adisa Mikhail O., Luukkainen Roope, Hyrynsalmi Sonja M., Mässeli Niina, Azad Nasreen, Haque Bahalul, Joutsenlahti Juha-Pekka, Legesse Wondemeneh, Abdelsalam Ahmed, Gurzhii Anastasiia, Ikonen Jouni, Jansen Slinger, van Schothorst Casper

DOI: 10.1007/978-3-031-20706-8_13

A Systematic Mapping Study of Empirical Research Methods in Software Ecosystems*

Larry Abdullai[1], Hatef Shamshiri[1], Hasan Mahmud[1], Muhammad Hamza[1], Essi Aittamaa[1], Jaakko Vuolasto[1], Mikhail O. Adisa[1], Roope Luukkainen[1], Sonja M. Hyrynsalmi[1], Niina Mässeli[1], Nasreen Azad[1], Bahalul Haque[1], Juha-Pekka Joutsenlahti[1], Wondemeneh Legesse[1], Ahmed Abdelsalam[1], Anastasiia Gurzhii[1], Jouni Ikonen[1], Casper van Schothortst[2], and Slinger Jansen[1,2]

[1] LUT University, Finland
[2] Utrecht University, the Netherlands

Abstract. Empirical software ecosystems research is hard. Researchers insufficiently report their research methods, which makes replication hard. Furthermore, researchers rarely adhere to FAIR practices, further impeding reproducibility and validity of the studies. In this paper we present the results of a systematic mapping study of the research methods that are applied in empirical software engineering research in the field of software ecosystems. We show that authors do a poor job of reporting about their research methods. Simultaneously, we identify a set of guidelines for the community that can help authors in this growing field.

Keywords: Empirical research · Software ecosystems · Software engineering · Platform management · Research validity

1 Introduction

A software ecosystem is a set of actors that functions as a unit and collectively serves a market for services and software, usually based around a platform or technology [7]. There was a time when software was developed in-house and this was an old practice for software development. In the new era, the open ecosystem is playing a big role for software development as the employees of different companies are able to participate and do collaboration through distributed collaboration from different companies. When the software ecosystem has an open access that helps other actors to perform better. The cooperation basically could include the developer community, developers who are working in different companies, partners, competitors and so on [15]. Based on the role of the ecosystem, the companies can provide high quality user contributions for the projects. Thus the companies keeps the customers and contributors satisfied [1].

* This work was created at an extended course at LUT University in the spring of 2022. The data is available through this link: `https://docs.google.com/spreadsheets/d/1cpsD4uxhU9hIwHbUDuIjaG2o0FnHZCybfjo6idtEQ-U/edit?usp=sharing`.

Software engineering spans across social as well as technical boundaries. To understand how software engineers build and maintain complex, developing software systems, we must study the empirical methods they use and human actions such as their social and cognitive processes. We need to know how software developers produce software and how teams and organizations coordinate. Although there is a growing concern empirical software engineering, there is scant guidance regarding the most suitable research methodologies. Many researchers choose improper approaches because they don't comprehend their goals or alternatives.

The objective of our study is to explore how software ecosystem researchers use empirical research methods in their work, with the larger goal of maturing the field. We hypothesize that, similarly to other young domains such as software engineering [4], researchers can become better at executing and reporting their research. The research question that drove this research is:

RQ: How do software ecosystem researchers use empirical research methods?

The rest of the paper is organized as: Section 2 outlines the systematic mapping study research method, Section 3 outlines the result analysis of the extracted dataset, Section 4 describes the top 5 papers having the highest value in meeting the ACM Sigsoft empirical standards for software engineering, Section 5 represents the recommendation of the authors after the analysis, Section 6 concludes the paper and outlines the future research directions.

2 Systematic Mapping Study Research Method

To arrive at a representative and quality sample of studies from the field of software ecosystems, we commenced our systematic mapping study following the guidelines proposed by Kitchenham et al. [8] and Petersen et al. [11]. We started with defining our research question and designing our mapping studies protocols. This enabled us to extract keywords necessary to answering our research question and using them as a basis to form our search string **("software" AND "ecosystem" AND ("platform" OR "open source" OR "governance" OR "package")**. The keywords limited the search to finding scientific articles relating to software ecosystems, rather than any other forms of business ecosystems or ecological ecosystems. Since the ecosystem terminology is used in nearly all fiends, this strategy excluded literature not directly relating to software ecosystems. We used some of the orthodox guidelines in conducting systematic mapping. However, we applied some exotic flavor in our strategy for selecting the literature as illustrated in Figure 1 and described in the following paragraphs.

We first chose Google scholar as our primary database search engine. Authors were divided into groups of two and assigned to selecting a total of twenty papers from a particular year for the period 2011-2021. For example, two of the authors were assigned to selecting ten paper each from year 2021 and another group assigned to the preceding year in that order. The two researchers in a group then discuss and agree between themselves the suitability of the paper selected.

This strategy ensure that relevant papers were evenly selected from each year and also to avoid sampling and selection bias.

During the search, it became evident that, due to to the huge amount of literature (an average of 60,000 articles unique hits for each year within the selection year range) it was not practical to read every single article. The authors therefore relied on the "sort by relevance" feature of the Google Scholar search engine to arrange the generated literature. Other databases such as Scopus and Web of Science together with a set of more formal decision criteria were also utilized to compare with the most relevant papers from Google Scholar. The following set of exclusion and inclusion criteria were used: (i) Only studies relating to software ecosystems with focus on platform, open source, governance or package were selected. (ii) Only papers with twenty or more citations were selected with the exception of articles published in 2020 and 2021, to ensure a minimum degree of quality. (iii) Only full papers were selected. We referred to full research paper to mean articles with not less than seven pages. (iv) Only open access literature or those that could be accessed through legal sources were considered. Finally, (v) Systematic review studies were excluded from the list.

After applying these inclusion and exclusion decision criteria, the final literature selection process relied on the researchers' thorough reading of the full paper since the data set was arranged in order of relevance, to select ten papers each for analysis. Overall, a total of 89 sample papers were included in the mapping study. Although there is the likelihood that we might have missed some potentially relevant papers giving the selection approach we adopted, we believe that our strategy allowed us to evenly and fairly select papers year by year which is representative of the topic under investigation.

For the analysis of the research methods we have used the ACM Sigsoft standards for empirical research in software engineering [13] [3]. This set of standards includes quality guide for number of different research methods, e.g., case study, mixed methods and grounded theory. These empirical standards have been proposed to provide generic standards for the evaluation of research methods and should be considered a useful tool for researchers when creating their articles. For each article we have observed the ***Essential Attributes***, the ***Desirable Attributes***, and the ***Extraordinary Attributes*** as described by the standards.

Additionally, the importance of following the ACM Sigsoft standards is caused by the requirements set by the scientific community. Since we included scholarly peer-reviewed articles (e.g. journal papers) as well as conference proceedings, the predetermined criteria helped us focus on the necessary components that the chosen papers should contain. We used 19 criteria (e.g. justification for the choice of case(s) or object(s) that have been studied, description in rich detail, and presentation of the precise chain from observation to findings, etc) to evaluate the chosen research papers for further analysis.

[3] Latest standards `https://github.com/acmsigsoft/EmpiricalStandards/`

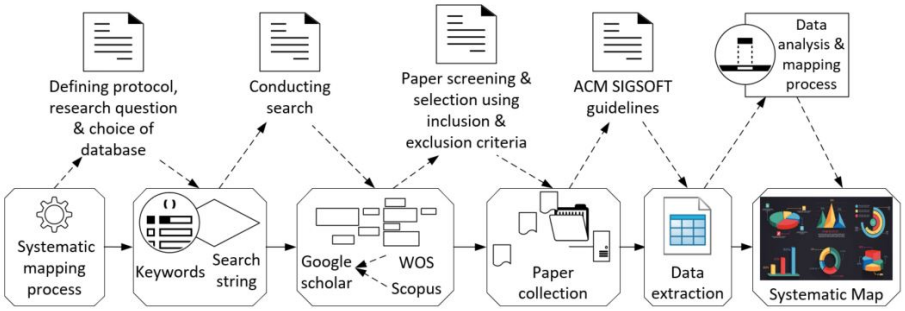

Fig. 1. Systematic mapping study design and execution process

3 Results

We conducted two analysis 1) classification of used research methods and 2) analysis of utilized ACM Sigsoft Case Study standards. In classification of used research methods we firstly identified research method from each selected paper and then counted frequency of each research method type. In ACM Sigsoft Case Study standard analysis, we analyzed the selected case study articles, including mixed method studies and repository mining case studies. Analysis results are presented in following Sections.

3.1 A Classification of used Research Methods

The term *research method* describes an approach to conducting research, specifically the numerous sorts of activities used to systematically address the research problem that is predicated on assumptions and provide a rationale for the decisions made. A research methodology may use a variety of research methods which are the tools used to gather and analyze data [9].

To determine the methodology adopted by the selected studies, a research method analysis on a sample of the collected papers was carried out. The results of the performed analysis can be seen in Figure 2. It can be noticed that the majority of researchers have conducted case studies to carry out the research. Furthermore, the top three research methods used are, in order:

• **Case study.** Case study research is a quantitative research method used to track the progress of projects, activities, or assignments. Throughout the study, data is collected for a specific purpose. Statistical analyses can be performed based on the data collected. The case study is typically used to track a specific attribute or to establish relationships between various attributes. A case study could, for example, be used to develop a model to predict the number of errors in testing. In these types of studies, multivariate statistical analysis is frequently used. Linear regression and principal component analysis are two of the analysis methods [21]

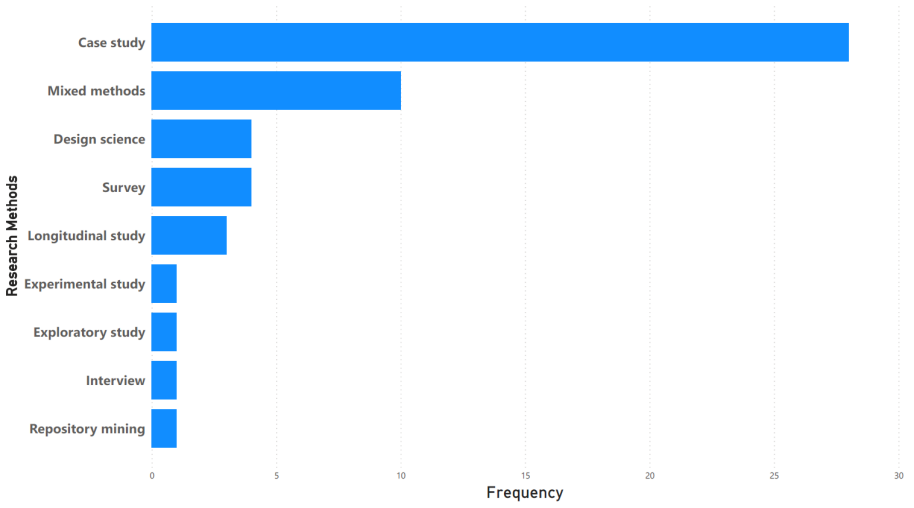

Fig. 2. Classifiable and identifiable research methods within the articles of the study.

• **Mixed methods.** Mixed methods research is a combination of multiple research methods. The method uses both quantitative and qualitative data collection and analysis strategies. The mixed methods research approach can be utilized if the researcher desires a more robust, multifaceted viewpoint, one example could be using Interviews to supplement statistical results from a survey [14].

• **Design science.** Design science research is a qualitative research methodology that simultaneously produces knowledge about the process used to create an artifact and the artifact itself, with the object of the study being the design process. Furthermore, Design science research seeks to provide information that is prescriptive for professionals in a field and to share empirical leanings from studies of the prescriptions used in context [16]. Such information is known as "design knowledge" since it aids practitioners in creating solutions to their issues [5].

On the other hand, the least used methods are exploratory studies, interviews and repository mining.

3.2 An analysis of utilized ACM Sigsoft Case Study standards

In Table 1 we present an in-depth analysis of the research methods encountered in the case study articles that were part of the mapping study. We have analyzed 51 articles that were either pure case studies, mixed method studies, or repository mining studies, hence the number is larger than the number of pure case studies mentioned in Figure 2.

When looking at the overall results, it is pleasing to observe that most studies check the ***essential aspects***. Most studies explain in rich detail the object of

study and why the case study was done. However, authors frequently neglect to mention the type of case study (e.g., exploratory or explanatory). More problematic is the lack of explanation of how the case studies lead from observations to findings. While this in part can probably be blamed on page limitations, it is nowadays increasingly common to also make the case study report available, in which this chain of evidence could be provided.

When looking at the ***desirable aspects***, things become more dire. While many studies use triangulation and present evidence such as interview quotations (approximately half), many do not. Furthermore, only approximately a fourth of studies use cross-checking techniques for corroborating evidence. Also, fewer than half include an in-depth discussion of the biases that the work potentially suffers from. If the field of software ecosystems is to be matured, it is essential that more studies report on these desirable aspects in the future.

Finally, the ***extraordinary attributes*** are rarely observed. While not problematic, these are aspects that researchers in the future may have to include in future studies. The authors of this work, for instance, have little experience with pre-publishing a case protocol beforehand. While this practice may be common in other fields, we are still maturing. One thing the field can pride itself with: we often publish multiple case studies and perform cross case analysis. This may be a sign that the field is maturing, that case studies are relatively ample available, and that we may wish to create better access to case materials, to encourage reuse and sharing of research assets.

4 Highlighting Five Studies from the Data Set

The five studies highlighted from the data sets are selected for having the highest value in meeting the ACM Sigsoft empirical standards for software engineering research, among our literature selections.

4.1 The Dynamics of Openness and the Role of User Communities A Case Study in the Ecosystem of Open Source Gaming Handhelds

Using the open source handheld games ecosystem as a case study, the paper reveals the influence of openness on games manufacturers' interactions with their user communities and customers' changes in preference. By using a longitudinal approach, the paper answered the two research questions: 1) how does the discrepancy between the openness provided by firms and the openness demanded by a user community emerge? And 2) what are the consequences for the ecosystem (especially the firms) when firms fail to address such a discrepancy?

For the first question the authors posit that often time, discrepancies occurred when the level of openness provided by the firm fell below the expectation of their targeted users' community (a case of misaligned information releases). Most firms are committed to achieving a symbiosis relationship with their users-developers

Table 1. Out of 51 studies, we find that they check these boxes. While many of the essential criteria are checked in high quality studies, many of the desirable aspects are not. Please note that we identified 51 case study articles, including mixed method studies and repository mining case studies, which is why this number is larger than the number of case studies in Figure 2.

Quality Factor Case Studies	**Out of 51**
Essential	
justifies the selection of the case(s) or site(s) that was(were) studied	46
describes the site(s) in rich detail	42
reports the type of case study	31
describes data sources (e.g. participants' demographics and work roles)	38
defines unit(s) of analysis or observation	43
presents a clear chain of evidence from observations to findings	37
Desirable	
provides supplemental materials such as interview guide(s), coding schemes, coding examples, decision rules, or extended chain-of-evidence tables	25
triangulates across data sources, informants or researchers	23
cross-checks interviewee statements (e.g. against direct observation or archival records)	16
uses participant observation (ethnography) or direct observation (non-ethnography) and clearly integrates these observations into results	18
validates results using member checking, dialogical interviewing, feedback from non-participant practitioners or research audits of coding by advisors or other researchers	19
describes external events and other factors that may have affected the case or site	25
uses quotations to illustrate findings	19
EITHER: evaluates an a priori theory (or model, framework, taxonomy, etc.) using deductive coding with an a priori coding scheme based on the prior theory	25
OR: synthesizes results into a new, mature, fully-developed and clearly articulated theory (or model, etc.) using some form of inductive coding (coding scheme generated from data)	6
researchers reflect on their own possible biases	18
Extraordinary attributes	
multiple, deep, fully-developed cases with cross-case triangulation	16
uses a team-based approach; e.g., multiple raters with analysis of inter-rater reliability (see the IRR/IRA Supplement)	8
published a case study protocol beforehand and made it publicly accessible (see the Registered Reports Supplement)	2

community in their openness by benefiting from the valuable contributions offered by the external community contributors and reducing the possibility of incurring negative outcomes that may jeopardize their productivity [20]. The study shows how users' empowerment through the online community put them in a very good position to influence a firm's openness levels users. Regarding the second research question, the inability of a firm to effectively manage the users' demands for openness and emerging conflict can constitute a huge risk to the firm's survival, its competitiveness and may possibly lose to an alternative user-initiated project springing up in the ecosystem thereby making openness a dimension of competition. A firm's degree of openness, therefore, relies on its strategy as well as extra-organizational actors (users and developers) [20].

The paper proposed a framework of the dynamics of openness built on the repeated patterns illustrating the open source game ecosystem. It highlighted the importance and dynamism of openness as a dimension for competition. It also emphasises the importance of user-driven innovation over the manufacturer-driven product lifecycle. Three examples (Gamepark, GP Holdings, Project Ninja) of firms that failed to manage conflict arising from openness are presented and one example (Open Pandora) that was able to manage their users' demand for openness and resulting conflict successfully. The Gamepark's failure is due to resentment from dissatisfied users' s community members who felt their demands are being ignored while asserting that the Gamepark's success is mostly from the users' contribution. This resulting conflict is a huge blow to the company as the users' community eventually launched an alternative product which eventually dies due to the same reason. Decisions regarding a firm's openness strategy required adequate planning and effective communication with the targeted users' community as such decision is often difficult to reverse [20].

The authors used multiple data sources (forums, expert interviews, and secondary archival data) to validate and support their argument and used triangulation in the analysis. The proposed framework provides an abstract way of identifying and managing discrepancies that emerge between openness offered by firms and openness demanded by the user communities as well as the consequences arising thereafter. In addition, the author cites real-life cases of Handheld gaming firms to support their findings and by stating the research limitations, the authors provide useful direction for the future of the research work. All the data presented also meets the FAIR principles respectively.

The major weakness of the paper is the ecosystem selection, which limited the choice to the handheld gaming industry ecosystem, thereby making it narrow and too specific. The paper also failed to consider the contribution of ordinary users (non-developer) during the semistructured interview process. Input from other related ecosystems as well as other users would have helped to strengthen the findings.

The longitudinal case study approach focuses on the open source gaming handheld industry and helps to validate the evolution and the dynamics of firms, user communities, and their interactions. We find that a suboptimal level of openness can pose a threat to a firm's very existence.

4.2 How do software ecosystems evolve? A quantitative assessment of the R ecosystem

Plakidas et al. [12] focus on the analysis of the emergence of software ecosystems by examining the structure and emergence of the R ecosystem as a case study of the open-source ecosystem. Package repository mining and statistical analysis method was used to extract R package data over a period of 12 months. The paper reveals metrics that help to identify and characterized a successful software ecosystem and compares it to approaches from related ecosystems literature by creating and comparing the prediction models based on package download frequency.

The software ecosystem consists of three main components namely: the software platform, the community of users, and the marketplace(s) respectively. The R ecosystem is classified into Platform Characteristics, Marketplace and Package Characteristics, and Community Characteristics. The paper was structured to provide answers to three research questions RQ1: How does the R software ecosystem evolve? The R ecosystem still enjoys strong growth in sizes and varieties since its inception RQ2: How do the community members collaborate, and how does this impact the software marketplace? The stakeholders play a significant role in boosting the marketplace by providing several packages that extend the functionality of the R core, with the active involvement of "insiders" as well as the single-package contributors. RQ3: What makes a software ecosystem marketplace product successful? A strongly-established community commitment and frequently maintained package contributed by experienced authors constituted a successful marketplace ecosystem [12].

The paper was well structured and clear answers were provided to the research questions. The paper employed a quantitative analysis of the R ecosystem to assess and quantify its emergence, and derive metrics on its core software components, the marketplace as well as its community, in addition to validating existing theories from the literature. The metrics and basic characteristics of the mined data are presented and supported with related analysis. Threats to validity are discussed from multiple perspectives and this help to strengthen the validity of the findings.'

One weakness identified from this research is the sourcing of data from Bioconductor (data downloaded were limited to the previous 12 months), which indicates incomplete versioning differentiation. A better approach would have been obtaining data from multiple repositories like Github or the package homepage.

4.3 Knowledge boundaries in enterprise software platform development: Antecedents and consequences for platform governance

The extant research on platforms has studied extensively the roles and actions of a platform leader and complementors, whether it is about governance or technological aspects. However, according to Foerderer et al [6] "a comprehensive

picture of knowledge management in platform ecosystems did not yet exist." This is important, because complementors need knowledge about the platform functionality – what they can do with it to provide add-on services or products – and about the design of interfaces – how they can do it. A lot of the prior research has studied consumer-focused platforms, so the authors decided to focus on enterprise software, which is considered complex by nature. This complexity makes it non-trivial for the complementors to develop add-ons. The authors discuss knowledge management as one of the strategic activities of a platform owner, using and extending a knowledge boundary framework by Carlile [2]. The paper presents reasons for what hinders knowledge from passing the firm boundaries within a platform ecosystem, and what are its consequences.

The authors lay their foundations of the platform governance and knowledge management and show how they are related. From there they identify the research gap: what are the causes for knowledge gaps when boundaries are crossed and how the platform owners can try to manage the caps. Regarding the results, providing add-on services and products is about technological issues, but at the same they are not enough. The authors link technological properties of a platform – functional extent and interface design – with knowledge boundaries. The paper presents a classification of platform owners' approaches to managing knowledge boundaries: broadcasting, brokering, and bridging. The research is a multiple case study with four platforms and interviewees from different industries, emphasis on the complementors. Empirical standard of ACM Sigsoft for case study is matched rather well. Data triangulation is used: archival data complements the interviews. The analysis phase utilizes grounded theory with both a priori scheme and emerging codes. Inter-coder discussion and agreements was used. These tactics are applied in a rigorous fashion in the context of enterprise software platforms to show how knowledge boundaries come about and how they can be managed.

The multi-case approach and extensive data set provide a solid starting point. For some grounded theory researchers the use of a priori coding scheme could be an issue, as grounded theory is about the emergence of concepts from the data. While the authors used a priori coding as a starting point based on the theoretical background and then allowed for the emergence of concepts related to knowledge boundary resources, it could be asked if something was missed or emphasized incorrectly due to it. Another question is related to the interviewees. The selection of CEOs and or CPOs was justified, but would for example a product manager point of view enriched the results? However, the interviewees had a chance to discuss and validate the research results, and according to the authors this strengthened the internal and external validity of the study.

4.4 Technology Ecosystem Governance

The case study by Wareham, Fox, and Cano Giner [17] identifies and describes three major tensions present in an enterprise software ecosystem: standard-variety, control-autonomy, and collective-individual. These tensions can lead to

exclusive either-or choices that actors have to make, or they can manifest as complementary options. The authors have two research questions, first: "How are the main tensions in technology ecosystems addressed in technology ecosystem governance?" and second "Tensions can manifest themselves as either contradictory or complementary logics. Are contradictory and complementary logics present in technology ecosystems? If so, how are they governed?". Based on the analysis of the tensions and their interactions the article then provides advice for the design of an ecosystem governance.

The article presents foundations of ecosystems around platforms: community of complementors, innovation, governance mechanisms, and generativity. It then focuses on the management of complementors (or third-party partners / providers) and the paradoxes between stability and evolvability that are present in the governance of modern ecosystem. Conceptual development in the article starts from tensions and how they can appear in three dimensions: outputs, actors, and identifications. They are studied in the context of ERP software, and the selection of the case is justified well. The selected ERP ecosystem provides a B2B setting with "severe heterogeneity across customers, complementors, and complements" – in other words, a real-life example. With their case study the authors show evidence for the presence of the tensions in the ecosystem, describe causes and effects, and finally present insights how to "harness tensions as enabling forces that serve the overall needs of the platform."

Strengths of the article include detailed representation of the case study and its various attributes. Selection of the interviewees aims to handle the diversity present in the ecosystem. Grounded theory is used as a method for analyzing the data and a coding scheme is presented as well. Quotes from the interviews provide a very good understanding. Additional data sources are used to supplement the interview data. Finally, the results are summarized in a comprehensive implications section.

As the data set of the study is extensive and the analysis process was described as iterative, a more detailed description of the analysis would have been interesting to read. Cross validation or external factors that may have affected the data collection were not described in detail. Any possible bias of the authors was not discussed, although the Methods section mention that there were multiple validation incidents.

4.5 What Do Package Dependencies Tell Us About Semantic Versioning?

Decan and Mens discuss the semantic versioning policy and how four software packaging ecosystems comply with it over time [3]. The topic is important as the modern software development is more and more dependent on reusable packages of source code. Packages form dependency networks that can be rather extensive. Semantic versioning (semver) provides rules for assigning version numbers, so that the users of a package are aware of the nature of the changes in the package version. The paper considers the semantic versioning issues with the following questions: "to what extent maintainers in different packaging ecosystems rely

on the semver policy to define the dependency constraints for the packages they maintain, and to what extent semver can be assumed to be followed by required packages."

The authors introduce the concept of semantic versioning and discuss how it is promising, but not everyone is using it. Dependencies in packaging ecosystems have been researched before and the emphasis has been on code analysis, which can be heavy to perform and are specific to a programming language. The authors suggest a complementary approach that is based on package metadata, which makes it language independent. Four packaging ecosystems that use semver were selected for the study. Selection and exclusion criteria were specified. The four ecosystems are of different age, so the authors selected a common time period that provides data for all the ecosystems. A repository mining approach was then used with the selected ecosystems and the time interval. Because different ecosystems use somewhat different notations, normalization was done for the dependency constraints.

The results are split into several sections, and each section summarizes the findings very clearly. According to the article, semver compliance "increases over time for all ecosystems, while ecosystem-specific notations, characteristics, maturity and policy changes play an important role in the degree of such compliance." Another key observation is that ecosystems seem to be more loose than semver for packages that are in their early phases.

Strengths of the article include the look on multiple package managers with a unifying approach that enables their comparison. Differences of the ecosystems are presented. A considerable merit of the article is its Threat to validity section. Construct, internal, conclusion, and external validity according to Wohlin et al. [19] are discussed.

5 Recommendations for Empirical Researchers

Following the ongoing discussions, it is apparent that researchers in Software Engineering (SE) are adopting a variety of research methods, techniques, strategies, practices and tools to increase the quality of their research. Despite widespread interest in empirical SE studies [4], some researchers fail in presenting explicitly rigorous empirical evidence of their research goals, research questions, research methods, and validity defense, as we find in this preliminary study. Based on our findings, we make the following recommendations presented in Table 2.

Firstly, we noticed that in many articles the research goal and research questions are not explicitly stated but rather more generic, vague and implicit. While some of these studies are well cited, it is unclear what their direct contribution is to the field and theory of software ecosystems. Similarly, scholars in [4] posit that, it is essential when selecting an appropriate research method to first clarify the research question. Therefore, as the first recommendation (`R1`), we recommend researchers to **explicitly state the research objective and research question**, possibly even highlighting them for increased readability.

Table 2. Recommendations for SE studies.

ID	Recommendation	Reasoning
R1	Explicitly state the research question and research objective	Clarify and highlight the objective
R2	Utilize at least minimum essential attributes from ACM Sigsoft standards	Improve and standardize the quality of empirical studies in SE
R3	Utilize desirable attributes from ACM Sigsoft standards when research setting allows it	Deepen the quality by verifying in more detail what and why is made
R4	Utilize FAIR data sharing principles	Enable external validation and future work by meeting principles of data findability, accessibility, interoperability, and reusability

Secondly and thirdly, it is striking to see how few articles are matching the essential and desirable criteria in the ACM Sigsoft empirical standards. We therefore encourage researchers to ensure that their **articles (R2) meet at least the essential the ACM Sigsoft standards for empirical software engineering research** and **when research setting allows it, a study should also (R3) utilize the desirable standards**. The ACM Sigsoft standards offer guidelines to improve the quality of empirical studies in SE.

Fourthly, to provide possibility for external validation and future works it necessary to get access to used data and extra materials, e.g., via appendix or link to external data storage. Therefore, we recommend researchers to provide data and extra materials, i.e., (R4) **utilize FAIR data sharing principles** [18] to make data findable, accessible, interoperable, and reusable.

Based on our observations in the studied articles, these recommendations can be applied, e.g., for case studies. Therefore, in case study once the research question is stated, attention should be paid to the case description (essential attribute): why was it selected and what makes it suitable for the study. When performing interviews, the triangulation of informants is an important phase (desirable attribute), i.e., selecting interviewees from different work roles. The studies we highlighted in Section 4 had a broad set of informants, intended to cover for the diversity. For example, interviewing only CEO level may introduce a bias. We encourage the researchers to utilizing multiple data sources (desirable attribute), as it is a feature that was present in many of the studies that matched ACM Sigsoft criteria well. Moreover, although validity checking and validity discussion (desirable attributes) is by no means a novelty, it was surprising to find studies that only touched the subject lightly or even omitted it. To meet desirable attributes adding a section for validity discussion is advised.

6 Conclusions and Future Work

In this study, we have explored how do software ecosystem researchers use empirical research methods, and how ACM Sigsoft empirical standards were utilized in case studies. For this, we conducted a literature study, with a pragmatic approach, to identify relevant literature in software ecosystem field, and it is likely that some relevant articles are missing.

We analyzed empirical software ecosystem papers from the years 2011-2021 and found out that the most common research methods used in empirical software ecosystem research were case study, mixed methods, design science and survey. The collected articles were scored based on ACM Sigsoft Case Study standards and we highlighted the five highest rated articles as examples of good articles according to aforementioned standards.

The scoring of the articles revealed a huge variety in the research quality of the papers. Therefore, to help researchers to improve the quality of their software ecosystem research we provide four recommendations presented in Table 2.

In the future, we plan to expand this data (and paper) to include a better coverage of the literature, for instance by snowballing from the works of Manikas et al. [10] and perhaps by sharpening some of the inclusion criteria. For this reason, we avoided generalization in this work and discuss only inductively from the data that we have gathered.

It has been insightful to perform the analyses of the articles against the ACM Sigsoft standards; they provide a useful tool for assessing study quality and as a guideline for designing future studies. However, collecting data has been tedious. We also recommend that reviewers in the future perform similar assessments with each review and make this data publicly available. In this way, we can ensure that it becomes easier to perform systematic mapping studies in the future.

Acknowledgments

We thank Kari Smolander for organizing the session that sprouted this research.

References

1. Bosch, J.: From software product lines to software ecosystems. In: SPLC. vol. 9, pp. 111–119 (2009)
2. Carlile, P.R.: Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. Organization science **15**(5), 555–568 (2004)
3. Decan, A., Mens, T.: What do package dependencies tell us about semantic versioning? IEEE Transactions on Software Engineering **47**(6), 1226–1240 (Jun 2021). https://doi.org/10.1109/TSE.2019.2918315
4. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: Selecting empirical methods for software engineering research pp. 285–311 (2008)

5. Engstrom, E., Storey, M.A., Runeson, P., Höst, M., Baldassarre, M.T.: How software engineering research aligns with design science: a review **25**, 2630–2660 (07 2020)
6. Foerderer, J., Kude, T., Schuetz, S.W., Heinzl, A.: Knowledge boundaries in enterprise software platform development: Antecedents and consequences for platform governance. Information Systems Journal **29**(1), 119–144 (2019)
7. Jansen, S., Cusumano, M.A., Brinkkemper, S.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar (2013)
8. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering **2**, 66 (01 2007)
9. Kuhn, T.S.: The structure of scientific revolutions, vol. 111. University of Chicago Press: Chicago (01 1970)
10. Manikas, K., Hansen, K.M.: Reviewing the health of software ecosystems–a conceptual framework proposal. In: Proceedings of the 5th international workshop on software ecosystems (IWSECO). pp. 33–44. Citeseer (2013)
11. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Information and software technology **64**, 1–18 (2015)
12. Plakidas, K., Stevanetic, S., Schall, D., Ionescu, T.B., Zdun, U.: How do software ecosystems evolve? a quantitative assessment of the r ecosystem. In: Proceedings of the 20th International Systems and Software Product Line Conference. pp. 89–98 (2016)
13. Ralph, P., Ali, N.b., Baltes, S., Bianculli, D., Diaz, J., Dittrich, Y., Ernst, N., Felderer, M., Feldt, R., Filieri, A., de França, B.B.N., Furia, C.A., Gay, G., Gold, N., Graziotin, D., He, P., Hoda, R., Juristo, N., Kitchenham, B., Lenarduzzi, V., Martínez, J., Melegati, J., Mendez, D., Menzies, T., Molleri, J., Pfahl, D., Robbes, R., Russo, D., Saarimäki, N., Sarro, F., Taibi, D., Siegmund, J., Spinellis, D., Staron, M., Stol, K., Storey, M.A., Taibi, D., Tamburri, D., Torchiano, M., Treude, C., Turhan, B., Wang, X., Vegas, S.: Empirical standards for software engineering research (2020), https://arxiv.org/abs/2010.03525
14. Tashakkori, A., Creswell, J.W.: The new era of mixed methods **01**, 3–7 (01 2007)
15. Teixeira, J., Robles, G., González-Barahona, J.M.: Lessons learned from applying social network analysis on an industrial free/libre/open source software ecosystem. Journal of Internet Services and Applications **6**(1), 1–27 (2015)
16. Van Aken, J.E.: Management research as a design science: Articulating the research products of mode 2 knowledge production in management **16**, 19–36 (03 2005)
17. Wareham, J., Fox, P.B., Cano Giner, J.L.: Technology ecosystem governance. Organization science **25**(4), 1195–1215 (2014)
18. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The fair guiding principles for scientific data management and stewardship. Scientific data **3**(1), 1–9 (2016)
19. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
20. Zaggl, M.A., Schweisfurth, T.G., Herstatt, C.: The dynamics of openness and the role of user communities: A case study in the ecosystem of open source gaming handhelds. IEEE transactions on engineering management **67**(3), 712–723 (2019)
21. Zelkowitz, M., Wallace, D.: Experimental models for validating technology. **31**, 23–31 (05 1998)