

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Free and open source software sustainability: an exploratory analysis in the cloud observability ecosystem

Gabriel Silva Fontes

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Gabriel Silva Fontes

Free and open source software sustainability: an exploratory analysis in the cloud observability ecosystem

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Elisa Yumi Nakagawa

USP – São Carlos
August 2026

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

F683f Fontes, Gabriel Silva
Free and open source software sustainability: an
exploratory analysis in the cloud observability
ecosystem / Gabriel Silva Fontes; orientadora Elisa
Yumi Nakagawa. -- São Carlos, 2026.
137 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2026.

1. Software livre. 2. Sustentabilidade de
software. 3. Computação em nuvem. I. Nakagawa, Elisa
Yumi, orient. II. Título.

Gabriel Silva Fontes

**Sustentabilidade de software livre e open source: uma
análise exploratória no ecossistema de observabilidade de
cloud**

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação
e Matemática Computacional. *EXEMPLAR DE
DEFESA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientadora: Prof. Dr. Elisa Yumi Nakagawa

USP – São Carlos
Agosto de 2026

Em memória da Kiara, a melhor cachorrinha do mundo.

Acknowledgements

À Layla, cujo amor e cuidado são, para mim, o significado de lar; é graças a ela que eu tive forças para chegar até aqui, obrigado por tudo. À Laiane, que me amou e me apoiou incontáveis vezes, arrancando risadas de mim nos momentos em que mais precisei. À Dani, Murilo, Lucky, Deby, Gui, Marafelli, Luís, Flávio, Carlos, Rádio, Ani, Ana, aos amigos do Magic, aos Peres, aos Profetas, aos Márcios, aos Loiros, e a tantos outros círculos que fizeram todos os momentos valerem a pena.

Ao meu pai, Marcos, minha mãe, Sonia, minha irmã, Veronica, e minha avó, Marisa; por me estimularem a buscar meus sonhos, por me amarem tanto, e por tornarem possível tudo o que vivi. Muitas saudades de vocês.

Aos amigos do GELOS, do Grupy, do Sanca Hackerspace, por me inspirarem tanto neste trabalho, que tem tudo a ver com a nossa comunidade. Às tantas comunidades de Open Source e Linux pelas quais passei, por me fazerem me apaixonar pela computação.

A todos os amigos que fiz no Magalu Cloud, entre colegas e mentores, que me ensinaram e me inspiraram tanto. Aos amigos do LabES e do ICMC, que tornaram os momentos que tive no laboratório e em eventos verdadeiramente maravilhosos. To my friends at SEARCH and RUG, who helped make my stay in the Netherlands something truly unforgettable. To Professor Vasilios, for giving me that opportunity, for believing in my work, and for being a truly empathetic, kind, and brilliant co-advisor.

À Professora Elisa, a orientadora mais atenciosa, dedicada, compreensiva, e competente que alguém poderia querer. Obrigado por me ensinar tanto, por ser tão presente, e por sempre me ajudar a melhorar. Tenho muita sorte de tê-la como orientadora e amiga.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O estudo de mapeamento sistemático apresentado nesta dissertação também recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (313245/2021-5) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (2023/00488-5 e 2025/06946-0). O estudo sobre o ecossistema de observabilidade também recebeu apoio do CNPq (313245/2021-5), da FAPESP (2023/00488-5), e da ITEA4 e RVO, sob o contrato de financiamento n.º 22035 do projeto MAST (<https://itea4.org/project/mast.html>).

“A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it.”

“Não se pode entender um processo interrompendo-o. É preciso acompanhar o fluxo do processo. Devemos nos juntar a ele e fluir junto com ele.”

Frank Herbert, *Dune* (1965)

Abstract

FONTES, G. S. **Free and open source software sustainability: an exploratory analysis in the cloud observability ecosystem.** 2026. 137 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2026.

Free and Open Source Software (hereafter OSS) supports much of modern digital infrastructure, but the availability of source code and continued repository activity do not guarantee that a project can keep fulfilling its role. Sustainability depends on contributor renewal, governance, funding, licensing, evolution, and the ecosystem in which a project operates. In this dissertation, we investigate how OSS sustainability can be characterized and how it relates to a project's position in a software ecosystem, using cloud observability as our setting. We conducted three cumulative studies. First, a systematic mapping study of 111 publications synthesized OSS sustainability into 19 aspects across five dimensions. Second, we identified 45 OSS cloud observability tools from the literature and derived a relation graph from source-code references among the tools. Third, we selected the 14 most central tools and evaluated each against the 19 aspects. For each aspect, we assigned a qualitative rating and confidence level, and organized the resulting profiles into four groups according to their stewardship. The mapping showed that OSS research primarily addresses social and technical sustainability, while individual, economic, and environmental concerns remain less explored. The ecosystem study revealed recurring observability stacks and uneven centrality, and Prometheus occupies the most central position. In the project assessment, technical activity, functionality, and maintainability were the most consistent strengths, while risks appeared in governance, knowledge concentration, licensing, and economic support. Foundation-hosted projects had the most uniform social and technical profiles. Vendor-controlled projects generally combined substantial technical and economic support with concentrated governance, licensing, and product-boundary risks. Independent and research-led projects exposed funding and succession constraints. These results show that project sustainability profiles and ecosystem position describe distinct but complementary properties. Reading ecosystem position together with multidimensional profiles provides a structured exploratory way to identify project conditions and ecosystem exposure.

Keywords: Open source software, Software sustainability, Cloud computing.

Resumo

FONTES, G. S. Sustentabilidade de software livre e open source: uma análise exploratória no ecossistema de observabilidade de cloud. 2026. 137 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2026.

Software Livre e Open Source (doravante OSS) sustenta grande parte da infraestrutura digital moderna, mas a disponibilidade do código-fonte e a atividade nos repositórios não garantem que um projeto continue cumprindo seu papel. A sustentabilidade depende da renovação de contribuidores, governança, financiamento, licenciamento, evolução, e do ecossistema no qual um projeto opera. Nesta dissertação, investigamos como caracterizar a sustentabilidade de OSS e relacioná-la à posição de um projeto em um ecossistema, usando a observabilidade de cloud como nosso contexto. Conduzimos três estudos cumulativos. Primeiro, um mapeamento sistemático de 111 publicações sintetizou a sustentabilidade de OSS em 19 aspectos distribuídos entre cinco dimensões. Segundo, identificamos 45 ferramentas OSS de observabilidade de cloud na literatura e derivamos um grafo de relações a partir de referências entre as ferramentas em seus códigos-fonte. Terceiro, selecionamos as 14 ferramentas mais centrais e avaliamos cada uma nos 19 aspectos. Para cada aspecto, atribuímos uma classificação qualitativa e um nível de confiança, organizando os perfis resultantes em quatro grupos segundo o modelo de gestão dos projetos. O mapeamento mostrou que a pesquisa em OSS trata principalmente da sustentabilidade social e técnica, enquanto as dimensões individual, econômica, e ambiental permanecem menos exploradas. O estudo do ecossistema revelou combinações recorrentes de ferramentas e uma distribuição desigual de centralidade, com o Prometheus ocupando a posição mais central. Na avaliação dos projetos, a atividade técnica, a funcionalidade, e a manutenibilidade foram os pontos fortes mais consistentes, e identificamos riscos relacionados à governança, concentração de conhecimento, licenciamento, e suporte econômico. Projetos hospedados por fundações apresentaram os perfis sociais e técnicos mais uniformes. Projetos controlados por empresas geralmente combinaram forte suporte técnico e econômico com riscos de concentração de governança, licenciamento, e divisão entre projeto e produto. Projetos independentes e liderados por pesquisadores apresentaram limitações de financiamento e sucessão. Esses resultados mostram que os perfis de sustentabilidade dos projetos e suas posições no ecossistema descrevem propriedades distintas, mas complementares. Analisar a posição no ecossistema em conjunto com perfis multidimensionais oferece uma forma exploratória e estruturada de identificar condições dos projetos e nível de exposição no ecossistema.

Palavras-chave: Software livre, Sustentabilidade de software, Computação em nuvem.

List of Figures

Figure 1 – XKCD #2347: Dependency	24
Figure 2 – Relationship among the three studies and their contributions to the combined analysis	27
Figure 3 – SMS research method workflow	32
Figure 4 – Selected studies per publication year and venue type	36
Figure 5 – Term frequencies in the sustainability keywords extracted from selected study fragments	37
Figure 6 – Overview of research method	48
Figure 7 – Number of studies mentioning each tool (Tools appearing on a single study were omitted)	52
Figure 8 – Number of tools per role (Some tools have multiple)	52
Figure 9 – Yearly study distribution	53
Figure 10 – Relations between tools, clustered and weighted by relative code occurrence	54
Figure 11 – Overview of the research method and the inputs adopted from our prior studies	63

List of Tables

Table 1 – Set of studies selected	34
Table 2 – OSS sustainability aspects	39
Table 3 – Selected Tools	51
Table 4 – OSS sustainability aspects derived from our systematic mapping study (FONTES <i>et al.</i> , 2026b)	64
Table 5 – Tools selected for sustainability evaluation, ordered by ecosystem centrality. The number of studies mentioning each tool in their abstracts is also shown for reference; data are from our observability study (FONTES <i>et al.</i> , 2026a).	66
Table 6 – Qualitative scoring scale for sustainability aspects	67
Table 7 – Analysis criteria: evaluation question and observable evidence per sustainability aspect	67
Table 8 – Sustainability profiles of the 14 tools, grouped post hoc by current stewardship model.	72
Table 9 – Sustainability fragments and keywords extracted from the selected studies	107
Table 10 – Sustainability profile: Prometheus	124
Table 11 – Sustainability profile: OpenTelemetry	125
Table 12 – Sustainability profile: Apache Kafka	126
Table 13 – Sustainability profile: Jaeger	127
Table 14 – Sustainability profile: Kepler	128
Table 15 – Sustainability profile: Elasticsearch	129
Table 16 – Sustainability profile: Kibana	130
Table 17 – Sustainability profile: Grafana	131
Table 18 – Sustainability profile: InfluxDB	132
Table 19 – Sustainability profile: Nagios	133
Table 20 – Sustainability profile: Net-SNMP	134
Table 21 – Sustainability profile: Ganglia	135
Table 22 – Sustainability profile: PowerJoular	136
Table 23 – Sustainability profile: JoularJX	137

List of abbreviations and acronyms

AGPL	GNU Affero General Public License	NER	Named-Entity Recognition
API	Application Programming Interface	OSI	Open Source Initiative
ASF	Apache Software Foundation	OSS	Open Source Software
BSD	Berkeley Software Distribution	OTel	OpenTelemetry
CHAOSS	Community Health Analytics in Open Source Software	OTLP	OpenTelemetry Protocol
CI	Continuous Integration	PMC	Project Management Committee
CLA	Contributor License Agreement	PR	Pull Request
CNCF	Cloud Native Computing Foundation	RAPL	Running Average Power Limit
CoC	Code of Conduct	RFC	Request for Comments
CVE	Common Vulnerabilities and Exposures	RQ	Research Question
DCO	Developer Certificate of Origin	SDK	Software Development Kit
ELK	Elasticsearch, Logstash, and Kibana	SE	Software Engineering
FSF	Free Software Foundation	SIG	Special Interest Group
GPL	GNU General Public License	SMS	Systematic Mapping Study
HPC	High-Performance Computing	SNMP	Simple Network Management Protocol
IoT	Internet of Things	SSPL	Server Side Public License
KIP	Kafka Improvement Proposal	TICK	Telegraf, InfluxDB, Chronograf, and Kapacitor
LLM	Large Language Model	TSDB	Time-Series Database

Contents

1	Introduction	23
1.1	Context	23
1.2	Motivation and Research Problem	25
1.3	Objectives and Research Questions	26
1.4	Research Method	26
1.5	Contributions	27
1.6	Organization	28
2	A Systematic Mapping on Open Source Software Sustainability	29
2.1	Introduction	30
2.2	Research Method	32
2.2.1	Planning	33
2.2.2	Conducting	33
2.3	Results	35
2.3.1	Overview of Studies	36
2.3.2	Sustainability Aspects	36
2.3.3	OSS sustainability understandings	41
2.4	Discussion	42
2.4.1	Main Findings	42
2.4.2	Future Work	43
2.4.3	Threats to Validity	43
2.5	Final Remarks	44
3	An Overview of the Open Source Software Ecosystem for Cloud Observability	45
3.1	Introduction	46
3.2	Research Method	47
3.2.1	Search Phase	47
3.2.2	Extraction Phase	48
3.2.3	Selection Phase	49
3.2.4	Analysis Phase	50
3.3	Results	51
3.3.1	OSS tools in Cloud Observability Stacks	51
3.3.2	Combinations of OSS Tools in Cloud Observability Stacks	53
3.4	Discussion	54
3.4.1	Main Findings	54

3.4.2	Threats to Validity	56
3.4.3	Future Work	56
3.5	Conclusions	57
4	A Sustainability Analysis of Open Source Tools for Cloud Observability . .	59
4.1	Introduction	60
4.2	Background and Related Work	61
4.2.1	Software sustainability and OSS health	61
4.2.2	Project-health metrics	62
4.2.3	OSS governance and economic models	62
4.3	Research Method	63
4.3.1	OSS Sustainability Aspects	64
4.3.2	Identification of Open-source Tools for Observability	65
4.3.3	Evaluation Planning	66
4.3.4	Evaluation Execution	70
4.4	Results	72
4.4.1	Comparative overview	72
4.4.2	Foundation-hosted tools	73
4.4.3	Vendor-controlled tools	77
4.4.4	Independent projects	79
4.4.5	Research-led software	80
4.4.6	Synthesis	81
4.5	Discussion	82
4.5.1	What we learned about the tools	82
4.5.2	What we learned about the 19 aspects framework	84
4.5.3	What we learned about the ecosystem centrality metric	85
4.5.4	Implications	85
4.5.5	Threats to validity	86
4.6	Conclusion	87
5	Conclusions	89
5.1	Answers to the Research Questions	89
5.2	Contributions and Implications	90
5.3	Limitations	92
5.4	Future Work	93
5.5	Final Remarks	94
	References	95
	APPENDIX A OSS Sustainability Fragments and Keywords	107
	APPENDIX B Detailed Sustainability Profiles	123

Introduction

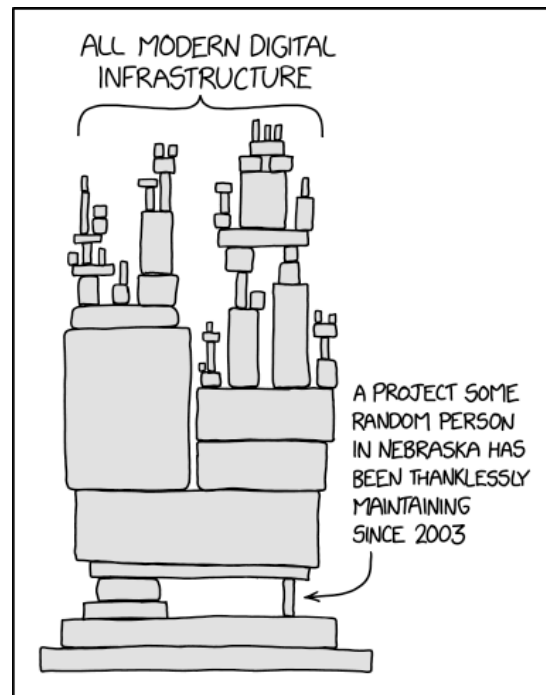
1.1 Context

Over the past few decades, Open Source Software (OSS)¹ has conquered the software world (ROBLES *et al.*, 2019). Its low barriers to adoption, opportunities for reuse, flexibility, and interoperability have made OSS critical to modern software systems (LINDEN *et al.*, 2009; NAGLE *et al.*, 2022). Across many sectors, the leading software product or one of its principal competitors is OSS (ROBLES *et al.*, 2019). Organizations frequently adopt OSS from the bottom up, through choices made by engineers rather than through explicit strategic decisions (HAUGE *et al.*, 2010).

This widespread use has made OSS a form of digital infrastructure. Eghbal (2016) compares widely relied-upon projects to “roads and bridges”, foundational infrastructure that enables a wide variety of activities, yet requires costly constant maintenance. Greenstein and Nagle (2014) use the expression “digital dark matter” to describe the substantial economic value OSS provides, despite being difficult to observe. They both identify a mismatch: a project may be critical to many organizations, while the responsibility for its governance, maintenance, and funding remains concentrated in a small community, a single company, or even a few individuals.

This is famously illustrated by XKCD comic #2347, “Dependency”, reproduced in Figure 1. The comic depicts a familiar case: the entirety of modern digital infrastructure quietly and transitively depending on a small project maintained by one little-known person. The comic has become one of the most recognizable depictions of how widely used dependencies may remain unnoticed until they fail.

¹ Throughout this dissertation, we use OSS as an umbrella term for Free Software and Open Source Software, covering software under licenses approved by both the Open Source Initiative (OSI) and the Free Software Foundation (FSF).



“Someday ImageMagick will finally break for good and we’ll have a long period of scrambling as we try to reassemble civilization from the rubble.”

<https://xkcd.com/2347>

Figure 1 – XKCD #2347: Dependency

Real incidents illustrate the possible scale: the 2016 removal of the left-pad package disrupted many thousands of projects ([npm, 2016](#)), while the Heartbleed vulnerability in OpenSSL exposed an estimated 24–55% of popular HTTPS sites ([DURUMERIC *et al.*, 2014](#)).

Adopting a project available under an open source licence does not guarantee continued maintenance, contributor succession, financial resources, or technical evolution. OSS projects also differ considerably in their origins and organization. Some are volunteer efforts maintained by one person; some are governed by foundations and developed by employees from competing companies; and others are controlled by a vendor whose business depends on the project. Forks may preserve the code when interests diverge, but they do not necessarily preserve knowledge, community, funding, or compatibility. Understanding whether an OSS project can continue fulfilling its role, therefore, requires more than simply verifying that its source code remains available.

Software sustainability provides researchers and practitioners with a vocabulary to understand and address this broader problem. The Karlskrona Manifesto describes sustainability through technical, social, economic, individual, and environmental dimensions ([BECKER *et al.*, 2015](#)). In OSS, the technical capacity to maintain and evolve software interacts with contributor communities, governance, personal knowledge and motivations, funding, monetization, and environmental concerns. Research on project health provides related indicators and practices ([LINÅKER *et al.*, 2022](#)), but sustainability has a wider scope and a longer-term perspective.

1.2 Motivation and Research Problem

OSS sustainability is difficult to study because both sustainability and OSS projects are multidimensional. Repository activity, release cadence, contributor count, and vulnerability handling provide useful evidence, but none of these indicators by themselves explain who can make decisions, whether project knowledge is shared, how development is funded, or what happens when a company changes its strategy. These concerns also interact. Commercial backing may fund a large engineering team while concentrating governance and relicensing power. Neutral governance may distribute authority while maintenance work remains concentrated in a single employer. A stable licence may coexist with an unfunded maintainer base, while a highly active project may repeatedly impose migration costs on its users.

The first problem addressed by this dissertation is thus conceptual: knowledge regarding OSS sustainability is dispersed across different terms and research traditions. Studies on project health, community participation, governance, technical maintenance, funding, contributor motivation, and environmental impact may all address sustainability without naming it as such. Without a synthesis, it becomes difficult to relate these findings or to evaluate a project without reducing sustainability to the indicators that are easiest to collect.

The second problem concerns the boundaries of an OSS project. OSS projects do not operate in isolation: they implement standards, exchange data, provide extensions, and occupy different roles in software stacks. A peripheral tool and a widely implemented interface may expose their ecosystems to different risks even when the projects have similar sustainability profiles. In the same way, a tool may appear central because its name or interface is frequently referenced, while the project behind it has concentrated knowledge, weak funding, or declining activity. Project-level indicators do not reveal which risks may propagate through an ecosystem, while ecosystem position does not describe the conditions of the projects occupying that position. Sustainability may, therefore, concern not only the project maintaining an implementation, but also the reach, continuity, and interoperability of its interfaces and the resilience of the ecosystem that depends on them.

Cloud observability provides a useful context in which to examine this relation. Cloud systems combine heterogeneous infrastructure, services, and software, so their systematic measurement through logs, metrics, and traces is necessary for operation (BEYER *et al.*, 2016; NIEDERMAIER *et al.*, 2019; PICORETI *et al.*, 2018). The OSS observability landscape includes collectors, databases, tracing systems, dashboards, exporters, alerting systems, and standards that are routinely combined into monitoring stacks. It also contains projects under different stewardship arrangements, including foundation-hosted, vendor-controlled, independent, and research-led projects. This ecosystem is, therefore, both practically relevant and diverse enough for us to compare project sustainability with ecosystem position.

In this dissertation, we investigate the thesis that OSS project sustainability cannot be

adequately characterized through project-level activity indicators or ecosystem position alone. Multidimensional profiles distinguish governance, funding, licensing, knowledge, and continuity conditions that similar activity levels may conceal. Ecosystem position, in turn, indicates the exposure associated with the projects or interfaces represented in the ecosystem graph. By reading both together, we can identify project conditions and ecosystem exposure that neither captures in isolation.

1.3 Objectives and Research Questions

The general objective of this dissertation is to characterize, through multidimensional profiles, the sustainability of the most central OSS projects in the cloud observability ecosystem and to examine how those profiles relate to ecosystem position. To that end, we organize this dissertation around three specific objectives:

1. Synthesize how sustainability is understood in OSS research and identify the aspects used to characterize it;
2. Identify the OSS tools that form the cloud observability ecosystem, their roles, and their relations; and
3. Apply the sustainability aspects to the most central OSS cloud observability tools and examine what their profiles reveal when considered together with ecosystem position.

These objectives correspond to the following research questions (RQs):

RQ1: How is sustainability understood and characterized in the OSS context?

RQ2: Which OSS tools form the cloud observability ecosystem, and how are they related?

RQ3: What sustainability profiles emerge among the most central OSS cloud observability tools, and what does comparing these profiles with ecosystem position reveal?

1.4 Research Method

We investigated the three RQs through three cumulative studies. Each study has its own research method, reported in detail in its corresponding chapter, while the sequence summarized here explains how their outputs informed the combined analysis, as illustrated by Figure 2.

First, to answer RQ1, we conducted a systematic mapping study of how sustainability is understood in OSS research. The study selected 111 studies and synthesized their findings into 19 aspects across the social, technical, individual, economic, and environmental dimensions. These aspects provided the multidimensional framework used in the third study.

Second, to answer RQ2, we studied the OSS cloud observability ecosystem. We identified 45 tools from the literature, classified their functional roles, and examined source-code occurrences to characterize relations among them. We then calculated a centrality score from normalized incoming reference weights to identify the tools occupying the most prominent positions in the resulting ecosystem graph. This study provided both the empirical setting and the selection criterion used in the third study.

Third, to answer RQ3, we selected the 14 tools in the top 30% centrality tier and assessed each of them against the 19 sustainability aspects. For every aspect, we recorded a rating, its supporting public evidence, and our confidence in that evidence. During the cross-case comparison, we organized the resulting profiles into four descriptive stewardship groups and examined them together with ecosystem position.

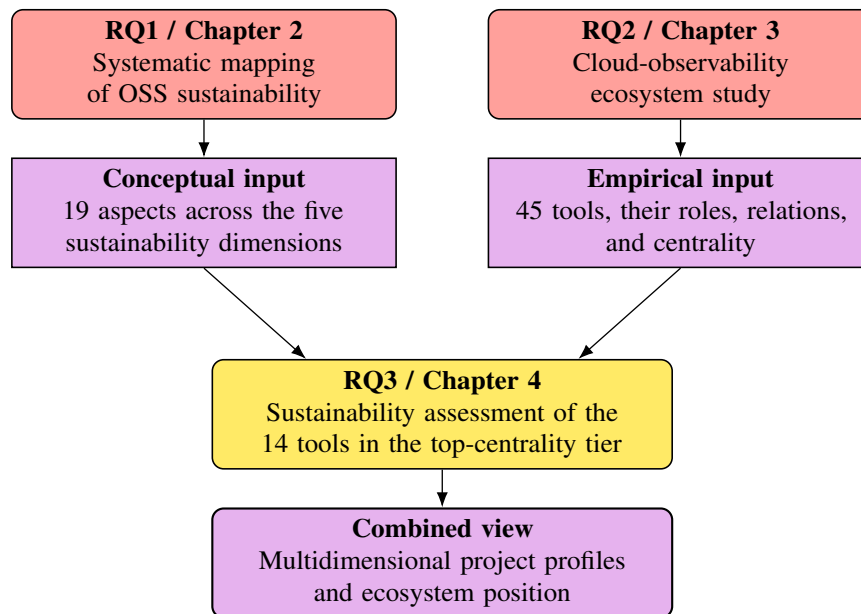


Figure 2 – Relationship among the three studies and their contributions to the combined analysis

1.5 Contributions

This dissertation makes three primary contributions:

1. A synthesis of OSS sustainability research into 19 aspects organized across the social, technical, individual, economic, and environmental dimensions. This synthesis consolidates concerns that appear under different terminology and provides an explicit structure for multidimensional assessment;
2. An overview of the OSS cloud observability ecosystem containing 45 tools, their functional roles, source-code relations, and centrality. It describes how observability stacks are assembled and identifies interfaces and projects that occupy prominent positions in the ecosystem; and

3. An evidence-based sustainability assessment of all 14 tools in the top 30% centrality tier, including evidence and confidence for each aspect. We organize the cases into four descriptive stewardship groups and show that ecosystem centrality mainly captures interface reach rather than project health. By considering centrality together with the project profiles, we identify governance, funding, knowledge, licensing, and continuity risks not represented by ecosystem position alone.

Together, these contributions build an exploratory approach that relates multidimensional project profiles to ecosystem position. The study does not replace repository metrics, ecosystem analysis, or practitioner judgment, but relates these forms of evidence, avoiding the assumptions that a technically active project is sustainable across every dimension or that a tool is healthy simply because many other tools integrate with it.

1.6 Organization

The remainder of this dissertation is organized as follows. Chapter 2 presents our systematic mapping study of OSS sustainability. It describes the study protocol, synthesizes 111 selected studies, and derives the sustainability aspects used later in the dissertation. This study was published at the 3rd International Workshop on Designing Software (Designing 2026), held at the 48th IEEE/ACM International Conference on Software Engineering (ICSE 2026) (FONTES *et al.*, 2026b).

Chapter 3 presents the OSS cloud observability ecosystem. We identify tools from the literature, classify their roles, analyze their source-code relations, and discuss the resulting centrality and integration patterns. This study was published at the 14th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS 2026), held at ICSE 2026 (FONTES *et al.*, 2026a).

Chapter 4 applies the aspects to the tools in the ecosystem's top-centrality tier. We report their comparative profiles, examine recurrent patterns, and discuss what this joint application reveals. The corresponding manuscript is being prepared for submission to the Journal of Systems and Software.

Finally, Chapter 5 summarizes the findings across the three studies, discusses the limitations of our combined approach, and identifies directions for future research.

A Systematic Mapping on Open Source Software Sustainability

This chapter is based on the paper “Is my Dependency Sustainable? A Systematic Mapping on Open Source Software Sustainability”, authored by Gabriel Silva Fontes, Vinicius dos Santos, and Elisa Yumi Nakagawa, published at the 3rd International Workshop on Designing Software (Designing 2026) at the 48th IEEE/ACM International Conference on Software Engineering (ICSE 2026).

Abstract

Context: With growing software reuse and interoperability, Open Source Software (OSS) is more ubiquitous than ever. Often sourced for free from volunteer-driven work, many developers lack awareness of the sustainability of OSS they depend on. Adopting OSS components while building software systems has long-lasting consequences, not only bringing benefits, but also risks. Understanding the sustainability of OSS projects thus becomes a critical activity during software design and development. However, OSS sustainability is not a very well-defined concept, which makes research and decisions regarding OSS adoption more difficult. **Objective:** Synthesize how the wider concept of OSS sustainability is currently understood "in the wild", explore how it can be characterized, and discuss the impact a better understanding can have on more responsible OSS adoption. **Method:** We conducted a systematic mapping study (SMS) across 111 studies from the literature to extract sustainability understandings and synthesize them into aspects of sustainability in OSS projects. **Results:** Social and technical concerns are well-explored and reaffirm the importance of the OSS community. Individual, economic, and environmental definitions remain relatively underexplored. There is a growing usage of the term "sustainability" in OSS research. We propose 19 aspects that can be utilized to understand OSS sustainability. **Conclusion:** Those aspects can be utilized by practitioners who rely on OSS

to better understand the projects they depend upon, as well as by researchers exploring this ecosystem.

2.1 Introduction

Open Source Software (OSS) is widely used in a variety of activities in the modern economy (WACHS *et al.*, 2022). It is present in most business domains, powers critical software infrastructure (EGHBAL, 2016), and prompts deep changes to quality through commodification pressure (LINDEN *et al.*, 2009), even in ecosystems dominated by proprietary software (ALAMI, 2020; YIN *et al.*, 2022). OSS has become a sort of "ethereal" resource: ever-present, deeply involved with everything it touches, and unnoticed by untrained eyes (EGHBAL, 2016; GREENSTEIN; NAGLE, 2014).

Software Engineering (SE) researchers and practitioners foresee an ever-present risk to the long-term sustainability of OSS, as more than 80% of OSS projects are eventually abandoned (SCHWEIK; ENGLISH, 2012). It is a challenge to assure the continued existence and quality of a resource that is, by its nature, ever-flowing, changing, and almost always distributed without warranties. OSS adoption is thus a risky decision, often made in a non-systematic manner. Since most programming languages, frameworks, and deployment/orchestration tools are OSS, adopting OSS is frequently an architectural decision, thus hard to make and costly to change. As with all such decisions, the cheapest moment to make them is during the initial phases of software design, so evaluating the sustainability of OSS components becomes a critical design activity for practitioners.

Software sustainability is a relatively recent concept that encompasses multiple aspects in software development/maintenance (PENZENSTADLER *et al.*, 2012). Sustainable software aims to not only meet present needs but enable future generations to continue creating and maintaining this software (BECKER *et al.*, 2015). Sustainability is described in the literature as a systemic concept that cannot be comprehended without systemic thinking (VOULVOULIS *et al.*, 2022). To help researchers and software engineers understand, software sustainability dimensions are often used. The Karlskrona Manifesto defines five main dimensions for sustainability: economic, social, technical, individual, and environmental (BECKER *et al.*, 2015).

From a social perspective, sustainable software aims to ensure equitable access to resources and opportunities for all individuals and communities (LAGO *et al.*, 2015). Economically, it involves preserving long-term value and capital for stakeholders while minimizing risks and maximizing returns (LAGO *et al.*, 2015; RAZAVIAN *et al.*, 2014). Environmentally, it focuses on minimizing the consumption of natural resources and mitigating environmental impacts associated with software activities (LAGO *et al.*, 2015). Individually, it prioritizes the well-being and development of human beings, encompassing aspects such as mental and physical health, education, and mobility (BECKER *et al.*, 2015; NAZIR *et al.*, 2020). Finally, from a technical

standpoint, sustainable software systems are designed for longevity, adaptability, and evolution in changing environments (LAGO *et al.*, 2015).

Achieving sustainability in software development involves addressing various challenges through the same five dimensions, including energy efficiency, social equity, economic viability, and environmental impact (HILTY *et al.*, 2011). It requires integrating sustainability considerations into all phases of the software life-cycle, from requirements gathering and design to deployment and maintenance (VENTERS *et al.*, 2014). Overall, being sustainable in the software development context means creating products that not only fulfill their intended purpose but also minimize negative impacts on society, economy, and the environment, thus continuing their existence for a longer period.

Open Source Software (OSS) aligns closely with the social dimension of sustainability, as it helps create a more equitable software development landscape. The social dimension emphasizes the preservation of communities that maintain and improve OSS, so that users have access to high-quality, efficient, and freely available software (BECKER *et al.*, 2015). OSS is typically sustained by self-organized communities (LIU *et al.*, 2020) whose goal is to safeguard the ability of future developers to contribute and evolve the software. However, sustainability is a multifaceted concept that is not well-defined in the context of OSS, and adopters therefore find it harder to evaluate the sustainability of their OSS dependencies while designing software systems. The existing body of knowledge on sustainability in OSS remains fragmented across many studies.

The research question (RQ) that has motivated this mapping is: “*How is sustainability understood in the OSS context?*” Several other secondary studies were published about software sustainability (PENZENSTADLER *et al.*, 2012; MOURÃO *et al.*, 2018; BERNTSEN *et al.*, 2017). Linåker *et al.* (LINÅKER *et al.*, 2022) published a secondary study on OSS health, which is related to sustainability, but not quite the same. Other authors (e.g., (CHENGALUR-SMITH *et al.*, 2010; LIAO *et al.*, 2018)) conduct longitudinal studies on OSS project sustainability that examine the factors influencing the long-term sustainability of OSS, but do not map the literature insights about sustainability in this domain. Therefore, developing a clearer understanding of what software sustainability means in the OSS context is essential to tighten the relations between the OSS community, sustainability researchers, and practitioners trying to better understand their potential OSS dependencies.

This chapter maps the understandings and aspects that define an OSS project as more sustainable. To do this, we performed a Systematic Mapping Study (SMS) that puts together 111 studies to extract (i) the current understanding of sustainability in OSS projects; (ii) aspects that comprise or contribute to OSS sustainability; and (iii) future directions and insights from the literature. The main contribution of this chapter is to present a broader view of which aspects make up OSS sustainability. We propose 19 sustainability aspects of OSS projects that were classified into the five sustainability dimensions. In addition, those aspects revealed attention

spots, such as the environmental concerns that are underexplored in the literature. We note that OSS culture influences the investigation, and it becomes a great opportunity to raise awareness on the overall sustainability problem in OSS, given how widely adopted OSS projects are.

The remainder of this chapter is structured as follows: Section 2.2 presents the research method. Section 2.3 describes the results of the SMS. Section 2.4 discusses the SMS results, its threats to validity, and actions performed to mitigate them. Section 2.5 provides the final considerations and future research perspectives.

2.2 Research Method

The SMS follows the guidelines of Petersen et al. (PETERSEN *et al.*, 2015). We follow the three-step process that encompasses planning, conduct, and results reporting, as shown in Figure 3. This section presents the first two steps (planning and conduct), outlining the most important elements of the SMS protocol. Section 2.3 presents data synthesis results.

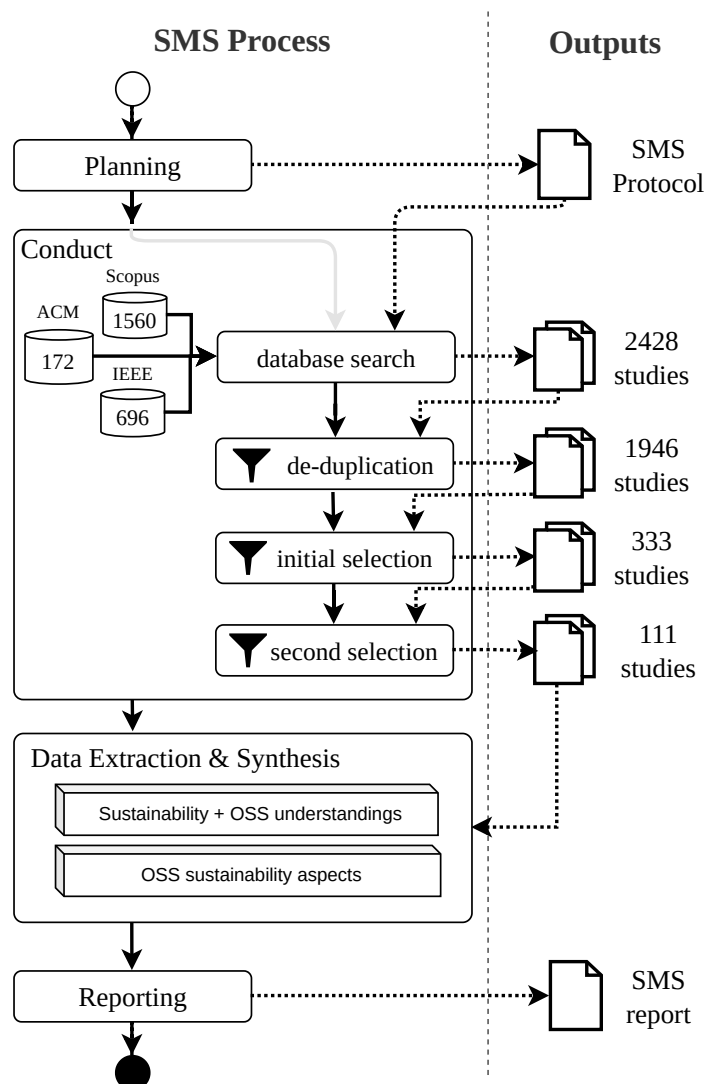


Figure 3 – SMS research method workflow

2.2.1 Planning

This section reports important elements of our SMS protocol that enable further audits and reproduction/updates. First, we defined our goal using the GQM (Goal Question Metric) (KOZIOLEK, 2008) approach as: analyzing primary studies *for the purpose of* identifying how sustainability has been understood/addressed in Open Source Software (OSS) *from the point of view of* SE researchers *in the context of* scientific literature. Based on this goal, we defined our research questions (RQs):

RQ1: How is sustainability understood in the OSS domain?

RQ2: Which aspects make an OSS project more sustainable?

Our search string defines sustainability and OSS as major domains. As we intended to locate studies that address "sustainability" within "OSS", we joined the two with the 'AND' operator. In addition, as OSS frequently appears as one of its synonyms, we joined those with the 'OR' operator. Our final string was: ("sustainability") AND ("open source" OR "free software" OR "FLOSS" OR "OSS"). This string was developed iteratively and pilot-tested to ensure sensitivity (capturing relevant studies) and precision (excluding irrelevant studies).

We adopted three databases (Scopus¹, ACM Digital Library², and IEEE Digital Library³), considering that they are well-known and widely used publication databases in the computing area. For the selection, one inclusion criterion (IC) and five exclusion criteria (EC) were defined to ensure a transparent and replicable process: **IC1:** The study addresses sustainability and OSS. **EC1:** Full text of the study is not available. **EC2:** Study is a shorter version of another. **EC3:** Study is a summary of a conference/workshop. **EC4:** Study is not peer-reviewed. **EC5:** Study is written in a language other than English.

2.2.2 Conducting

Study collection in electronic databases started by adapting the generic search string to each database. Scopus permits searching three metadata fields (title, abstract, and keywords) simultaneously. ACM Digital Library and IEEE require executing the search separately for each metadata field and combining them into a single studies dataset. We did not limit our search to publication period or field (like Computer Science), aiming to reach as many studies as possible.

Figure 3 illustrates the search process and scrutiny. Our systematic search initially retrieved a total of 2,428 studies (1,560 from Scopus, 172 from ACM Digital Library, and 696 from IEEE Digital Library). Following deduplication based on DOI and, when necessary, title and year, 1,946 unique studies remained. A first selection was conducted by screening

¹ <https://scopus.com>

² <https://dl.acm.org>

³ <https://ieeexplore.ieee.org>

titles, abstracts, and keywords against the inclusion and exclusion criteria, resulting in 333 studies for full-text assessment. During the second selection, all 333 studies were fully read and re-evaluated according to the same criteria, yielding 111 unique studies that are presented in Table 1. The selection process relied on a main researcher who was responsible for performing the first selection. In addition, an experienced researcher in sustainability and the SMS process double-checked the selections; finally, conflicts were resolved in consensus meetings.

Table 1 – Set of studies selected

ID	Authors	Year	ID	Authors	Year
S001	Mora-Cantallops et al.	2020	S044	Zhang et al.	2023
S002	Suleimenov et al.	2020	S045	Barcomb et al.	2020
S003	Mahaux et al.	2015	S046	Barcomb et al.	2022
S004	Wiggins et al.	2010	S047	Li et al.	2022
S005	Arantes et al.	2011	S048	Yue et al.	2023
S006	Robles et al.	2012	S049	Tan et al.	2023
S007	Petrinja et al.	2012	S050	Macho et al.	2013
S008	Barham et al.	2012	S051	Steinmacher et al.	2014
S009	Riehle et al.	2012	S052	Zhou et al.	2017
S010	Nyman et al.	2012	S053	Valiev et al.	2018
S011	Calefato et al.	2022	S054	Coelho et al.	2018
S012	Nov et al.	2008	S055	Gasparini et al.	2019
S013	Poba-Nzaou et al.	2019	S056	Qiu et al.	2019
S014	Liu et al.	2020	S057	Tan et al.	2020
S015	Ghapanchi et al.	2015	S058	Alami et al.	2020
S016	Santos et al.	2013	S059	Alami et al.	2020
S017	Gamalielsson et al.	2014	S060	Trinkenreich et al.	2020
S018	Butler et al.	2020	S061	Yin et al.	2021
S019	Hata et al.	2015	S062	Zhang et al.	2022
S020	Wang et al.	2022	S063	Shimada et al.	2022
S021	Jensen et al.	2011	S064	Fang et al.	2022
S022	Carillo et al.	2014	S065	Xiao et al.	2022
S023	Sajadi et al.	2023	S066	Gray et al.	2022
S024	He et al.	2023	S067	Zhang et al.	2022
S025	Ye et al.	2003	S068	Ramchandran et al.	2022
S026	Zanetti et al.	2012	S069	Zhou et al.	2022
S027	Izquierdo et al.	2015	S070	Zhang et al.	2022
S028	Barcomb et al.	2019	S071	Stanciulescu et al.	2022
S029	Dias et al.	2021	S072	Dam et al.	2023
S030	Trinkenreich et al.	2023	S073	Yin et al.	2022
S031	Yin et al.	2023	S074	Nakakoji et al.	2002
S032	Guizani et al.	2023	S075	Onoue et al.	2016
S033	Hasselbring et al.	2020	S076	Chengalur-Smith et al.	2010
S034	Zhang et al.	2022	S077	Papamichail et al.	2021
S035	Bird et al.	2007	S078	Liao et al.	2018
S036	Robles et al.	2009	S079	Perens et al.	2005
S037	Gupta et al.	2017	S080	Ju Long et al.	2005

Table 1 – continued

ID	Authors	Year	ID	Authors	Year
S038	Rastogi et al.	2016	S081	Carillo et al.	2013
S039	Horiguchi et al.	2021	S082	Carillo et al.	2013
S040	Chouchen et al.	2021	S083	Atiq et al.	2016
S041	Xia et al.	2023	S084	Chua et al.	2017
S042	Terceiro et al.	2010	S085	Vainio et al.	2006
S043	Goggins et al.	2021	S086	Sethanandha et al.	2010
S087	Kritikos et al.	2024	S088	Schwartz et al.	2024
S089	O’Neil et al.	2024	S090	Chang et al.	2024
S091	Xiao et al.	2023	S092	Sonabend et al.	2024
S093	Han et al.	2024	S094	Wattanakriengkrai et al.	2023
S095	Nguyen et al.	2023	S096	Sokyina et al.	2024
S097	Santos, Italo	2024	S098	Khan et al.	2024
S099	Zhang et al.	2024	S100	Jahn et al.	2024
S101	Fang et al.	2024	S102	Feng et al.	2024
S103	Han et al.	2024	S104	Linåker et al.	2024
S105	Wang et al.	2023	S106	Wilson, Katrina	2023
S107	Fang et al.	2023	S108	Hovhannisyan et al.	2024
S109	Adejumo et al.	2024	S110	Wattanakriengkrai et al.	2024
S111	Steinmacher et al.	2014			

The synthesis process for identifying OSS sustainability aspects involved a systematic three-step approach. First, fragment extraction was performed, where relevant portions of each study that discussed sustainability were carefully located. To support this process, we used an extraction form that recorded paper metadata (DOI, title, authors, publication year, venue, and venue type), along with the sustainability-related fragments. The extracted data was synthesized using thematic analysis (CRUZES; DYBA, 2011), applying both open and axial coding. Based on this, we developed a coding scheme for qualitative synthesis, classifying sustainability-related concepts in OSS.

Our first step in the analysis was to select fragments and extract terms from all the studies obtained through the SMS. The terms were deduplicated and factored into a set of what we called sustainability aspects. These aspects were further grouped according to their sustainability dimension (social, technical, individual, economic, environmental). We then show how the studies our SMS selected were classified according to these aspects in Section 2.3.2.

The full replication kit, containing a complete list of studies, selection decisions, raw data extracted, and synthesized data, is available in the external material.

2.3 Results

This section presents the data synthesis from the selected studies, followed by the answers to RQs.

2.3.1 Overview of Studies

Figure 4 shows the distribution of selected studies according to their publication year and venue type. The studies range from 2002 to 2024, with the majority of publications concentrated in the last few years.

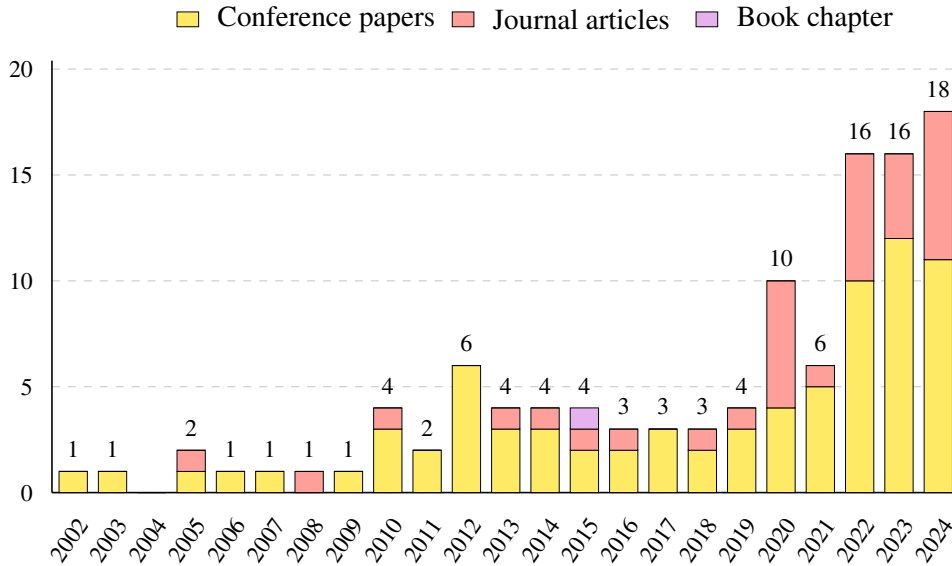


Figure 4 – Selected studies per publication year and venue type

The growing understanding of Software Sustainability as a multidisciplinary and systemic concept could explain the increasing number of publications. Many of the individual concerns have been previously researched without an explicit mention of sustainability itself. Concerning publication venues, around two-thirds of studies were published in event proceedings (i.e., 77), while one-third (i.e., 33) were published in journals, and a single book chapter. The increasing number of journal papers may indicate that the topic is becoming more mature over time.

2.3.2 Sustainability Aspects

Figure 5 provides an overview of the vocabulary from which the sustainability aspects were derived, visualizing individual term frequencies within the keywords extracted from the 111 selected studies. Larger terms occur more frequently in the extracted keywords. The complete fragments and their highlighted keywords are provided in Appendix A.

Our sustainability aspects were derived from the set of OSS sustainability terms that our SMS extracted. Their intent is to be able to fully express every sustainability concern our selected studies address, while minimizing overlap. The aspects are grouped within established software sustainability dimensions (social, technical, individual, economic, and environmental). These are defined as follows:



Figure 5 – Term frequencies in the sustainability keywords extracted from selected study fragments

Social Dimension

This dimension covers the human and community aspects of open-source software (OSS) projects. It emphasizes the interactions, governance, and social structures that sustain participation and collaboration.

1. **Contributor Retention:** Also called "stickiness"(HATA *et al.*, 2015). The probability of continued contributions over time. It reflects the project's ability to keep its contributors engaged and motivated.
2. **Contributor Attraction:** Also called "magnetism"(HATA *et al.*, 2015). The ability to attract new contributors is often linked to the project's popularity, niche, and functionality, as most contributors are also users/stakeholders.
3. **Internal Communication:** The effectiveness of communication among peers fosters a sense of community and belonging. It is related to governance transparency.
4. **Governance Structure:** The structure, transparency, legitimacy, and consistency of leadership regarding decision-making and project management.
5. **Openness/Onboarding:** The onboarding process and the project's attitude toward new contributors. This is highly related to governance and heavily affects contributor retention.
6. **Project Legitimacy:** The perception of the project, which can sometimes be enhanced through marketing. It is composed of smaller characteristics, such as popularity, niche, and age. Niche size interacts with other aspects (such as quality and feature set), increasing popularity. A popular project attracts users who may become contributors, forming a virtuous cycle (SANTOS *et al.*, 2013). Age, when combined with other aspects (e.g.,

sustained activity and maintainability), is a frequently used and accurate measure of sustainability.

7. **External Communication/Ecosystem:** The project's relationship with outside projects and its cooperation with external entities, which can enhance features and quality (e.g. through interoperability).

Technical Dimension

This dimension highlights the software engineering aspects of OSS projects. It focuses on the practices, architecture, and development processes that determine the project's long-term sustainability and adaptability.

1. **Sustained Activity:** One of the most frequently used measures. The ongoing software development activity, including the production of artifacts (release rate, issues, patches) and response times. Heavily influenced by contributor retention and governance; contributes to retention and attraction.
2. **Functionality, Scope:** The relevance of functional requirements to stakeholders and the project's ability to evolve and incorporate new features over time. Having exclusive functionality in a niche that usually lacks it frequently leads to increased project popularity and legitimacy.
3. **Quality Assurance:** The processes for bug resolution, adherence to quality standards, and testing, which contribute to the software's maintainability and increase its capability of incorporating new features without regressions, which otherwise often cost legitimacy.
4. **Methodologies, Practices:** The development methodologies and practices adopted by the project, which influence its maintainability, contributor retention, and quality.
5. **Architecture, Maintainability:** The design of the software architecture, including modularity and code complexity, affects contributor retention, leads to increased quality and is the limiting factor for development scalability (thus impacting feature set).
6. **Licensing:** Licensing policies, usually static over the project lifetime, impacting how the software can be used and contributed to. Similarly to governance, it interacts in a very complex way with other aspects. However, there is no conclusive research linking specific license types (e.g., permissive vs. copyleft) to higher sustainability.

Individual Dimension

This dimension captures the characteristics of individual contributors. It considers their motivations, knowledge, and skills, which directly affect the quality and direction of the project.

1. **Personal Motivations:** The individual motivations of contributors, which are often influenced by the sense of community, project niche, and feature set. It influences attraction and retention.
2. **Personal Knowledge:** The technical expertise and skills of individual contributors. It is affected by project niche/size and affects the quality and feature set of the project.

Economic Dimension

This dimension deals with the financial aspects of sustainability in OSS projects. It highlights the availability, management, and monetization of resources that enable long-term development.

1. **Financial Resources:** The attraction and management of financial resources, including foundations, crowdfunding, and sponsorships. This influences and is influenced by internal monetization and governance.
2. **External Monetization:** The ways external actors monetize their involvement with the project and their actions towards it. It includes paid contributors. It is highly related to external communication and openness.
3. **Internal Monetization:** The use of project funds to pay contributors, often seen in projects with corporate governance. The backing organization is, therefore, a heavy factor in the project's continued development.

Environmental Dimension

This dimension focuses on the environmental implications of OSS projects.

1. **Environmental Requirements:** The concerns related to the project's environmental impact. It is sometimes seen in the form of regulatory compliance (thus relates to governance).

Table 2 – OSS sustainability aspects

Dimension	ID	Aspect	Studies
Social	A1	Contributor Retention	S012, S013, S016, S017, S019, S025, S028, S030, S036, S037, S040, S045, S046, S047, S048, S049, S050, S051, S053, S056, S058, S059, S065, S066, S072, S074, S075, S076, S081, S082, S086, S087, S090, S092, S097, S098, S099, S102, S107, S111
	A2	Contributor Attraction	S005, S013, S015, S016, S019, S021, S024, S030, S031, S037, S039, S043, S047, S051, S053, S056, S059, S064, S065, S071, S073, S076, S099, S102, S104, S107

Table 2 – continued

Dimension	ID	Aspect	Studies
	A3	Internal Communication	S002, S003, S012, S013, S014, S017, S022, S023, S027, S029, S003, S030, S031, S043, S061, S068, S069, S071, S073, S076, S081, S083, S084, S085, S087, S092, S096, S097, S100, S102, S104, S106, S111
	A4	Governance Structure	S012, S013, S014, S017, S027, S029, S031, S043, S057, S058, S060, S061, S065, S071, S072, S073, S080, S084, S085, S009, S087, S089, S097, S102, S104, S105
	A5	Openness/Onboarding	S003, S035, S041, S043, S055, S056, S057, S058, S059, S060, S067, S070, S078, S081, S082, S087, S097, S099, S102, S104, S109, S110, S111
	A6	Legitimacy	S006, S007, S015, S043, S053, S064, S071, S076, S078, S080, S088, S095, S098, S101, S107
	A7	External Communication/ Ecosystem	S012, S013, S014, S043, S045, S006, S069, S071, S087, S104, S107, S110
Technical	A8	Sustained Activity	S004, S007, S010, S011, S013, S015, S016, S018, S033, S034, S038, S041, S043, S054, S068, S071, S072, S073, S076, S078, S088, S091, S092, S093, S095, S098, S103
	A9	Functionality/Scope	S003, S010, S013, S015, S017, S018, S029, S032, S044, S053, S071, S077, S078, S079, S087, S088, S092, S095
	A10	Quality Assurance	S008, S010, S013, S015, S018, S029, S040, S041, S043, S058, S059, S064, S071, S087, S092
	A11	Methodologies/ Practices	S001, S004, S005, S008, S019, S040, S043, S052, S058, S065, S071, S086, S087, S090, S092, S094, S108, S111
	A12	Architecture/ Maintainability	S001, S005, S015, S026, S042, S052, S053, S077, S087, S094, S111
	A13	Licensing	S017, S013, S015, S087, S092
Individual	A14	Personal Motivations	S012, S018, S056, S058, S069, S083, S110
	A15	Personal Knowledge	S015, S018, S002, S085, S100, S102, S106, S111
Economic	A16	Financial Resources	S013, S015, S016, S020, S032, S043, S005, S062, S063, S064, S076, S085, S087, S089, S092, S104, S110
	A17	External Monetization	S012, S014, S015, S034, S052, S067, S070, S079, S087, S092, S099, S104, S106
	A18	Internal Monetization	S019, S020, S079, S083, S087, S092
Environmental	A19	Environmental Requirements	S003

Table 2 shows an overview of our aspects and which studies address each of them. We can see that concerns overwhelmingly fit the technical and/or social sustainability dimensions, especially concerns that are frequently used as a measure of sustainability (e.g., sustained activity and contributor retention ([CHENGALUR-SMITH et al., 2010](#))). The technical and social concerns are also closely related, as they form a virtuous cycle ([SANTOS et al., 2013](#)) of project legitimacy, attraction, and activity ([CHENGALUR-SMITH et al., 2010](#)). Corporate involvement is also a relevant concern, as it bridges the economic (internal/external monetization) and social

(governance ([NAKAKOJI et al., 2002](#))) dimensions.

2.3.3 OSS sustainability understandings

Sustainability in the OSS domain encompasses a project's long-term viability and ability to continue delivering value (S010, S033, S098). It refers to the software's potential to maintain the functionality of the software system and support its future evolution (S044). A core characteristic of sustainable software is its ability to exhibit software development and maintenance activity over the long term (S013, S076), so that it remains active for a longer period of time (S076) and can evolve to incorporate new demands over time (S029). Sustainability also means creating the necessary conditions for future generations to use and evolve present artifacts (S077).

From a technical perspective, sustainability is closely tied to the software's inherent qualities and structure. Projects need to maintain high-quality standards and implement rigorous methodologies (S008). Being technically sustainable means overcoming inherent challenges of OSS like software complexity and its non-uniform evolution, which can pose threats (S001, S013). Long-lived projects usually adopt modular structures and less complex code, which incentivize developers and contribute significantly to maintainability (S005, S026, S042, S053). Other factors also contribute to sustaining OSS. For instance, a project's ability to reduce bugs and ensure stability (S013), along with a high probability that new revisions are published and that maintenance is consistent (S007, S071, S095), is important for technical sustainability.

A cornerstone of OSS sustainability is the underlying community of contributors (S002, S017, S025, S074). This involves the continuous attraction of new and retention of existing contributors (S013, S021, S024, S030, S037, S039, S099, S102, S107), and the literature recognizes that contributor disengagement threatens sustainability (S066, S090). Essential social aspects include the knowledge and skills contributors bring (S002, S012), fostering community growth (S005, S043, S102), building interpersonal trust (S023), ensuring fairness and acknowledging contributors' motivations (S012, S083, S096, S106). Effective governance rules (S027, S031, S055, S058, S070, S072, S089) with clear leadership (S017), a participative development process (S003), and transparent communication (S013, S029, S074, S100, S104) are vital. Mentoring and support to newcomers (S048, S049, S058, S060, S071, S102) along with promoting an open and inclusive atmosphere and diversity (S061, S092, S097, S099, S102, S104) are also critical for retaining members and promoting long-term success.

Despite OSS having a facilitated distribution due to its free licenses, financial resources (S005, S043, S062, S063, S084, S092), sponsorships, and fundraising (S013, S019, S020, S064, S095, S104) are important for project progress and sustainability. The increasing commercial participation and firm involvement in OSS projects have significant consequences for sustainability (S014, S034, S052, S053, S067, S070, S078, S082, S089, S099), and bring both benefits and new challenges to OSS communities. Sustainability plays an important role in the overall open-source ecosystem (S069, S094, S107), which means that OSS is part of a long and complex

chain of software dependencies that depends on financial support directly or indirectly.

At the individual level, sustainability is heavily influenced by the contributors themselves, particularly their knowledge and skills (S002) and their intrinsic motivations to participate (S012, S083, S096, S106). Factors such as fairness and perceptions of justice within the project community significantly affect contributors' willingness to engage (S012, S083). A developer's feelings of identity and belonging are also crucial for their retention and the project's long-term survival (S030), while the presence of toxic behaviors among developers can critically jeopardize community sustainability (S096, S104). From an environmental perspective, sustainability is understood to include broader ecological and social requirements (S003), which is not incorrect, but represents a disconnection between OSS development and other environmental aspects.

2.4 Discussion

This section presents our main findings, outlines some future work, and discusses the threats to the validity of the mapping, along with the countermeasures we have applied to address them.

2.4.1 Main Findings

The main findings resulting from our investigation are:

- **A way to evaluate OSS sustainability while designing software:** Our proposed aspects' main intended usage is as a starting point for more specific evaluation frameworks (e.g., with actual metrics). With that said, we believe that these are already useful to practitioners as high-level guidance on key concerns they should be on the lookout for while adopting OSS. We propose that further discussion and research about how OSS sustainability could be evaluated in a more systematic way during software design phases should follow.
- **Sustainability of OSS is uncontrollable, but can be understood:** The first step for organizations when designing their applications is to recognize that they have no significant control (in most cases) over OSS projects. Considering the wide range of factors affecting sustainability across various dimensions, only systemic and high-level actions have a direct impact on the sustainability of OSS. However, designers can and should understand all aspects that affect the sustainability of projects used as dependencies.
- **Sustainability aligns with OSS culture and values:** OSS is deeply rooted in a culture of collaboration and knowledge sharing, rather than profit-seeking. This culture seems to directly shape the factors that contribute to project sustainability. Introducing sustainability concerns to the OSS development community could strengthen OSS, as economic and environmental dimensions, which often conflict in proprietary software development, tend to be more aligned within OSS.

- **Privileged and underexplored dimensions:** Current research tends to prioritize technical and social aspects, with limited attention to individual, economic, and particularly environmental dimensions. The scarcity of studies addressing environmental issues within OSS hints that the relation between the OSS communities and the environmental impacts of their software has not yet been well explored.
- **Awareness of sustainability's systemic effects:** Adopting a sustainability-aware vision aligns OSS projects with long-term goals. While most research emphasizes sustainability in software, i.e., enhancing maintainability and system endurance, domain-specific studies highlight sustainability by software, showing how software can support social, environmental, and organizational goals. This approach encourages system designers and architects to consider the broader systemic effects of their decisions and to integrate sustainability into every stage of software engineering.

2.4.2 Future Work

The future is bright for further research within OSS sustainability. OSS sustainability is different enough from general software sustainability to warrant its own specificities, and the research community is still maturing them towards widely accepted understandings. Thus, research into any of the aspects will help further our current understanding of them.

The diversity of sustainability understandings might warrant surveys to gauge how researchers and practitioners from different segments understand the concept. Such surveys would help us see whether a wide consensus is possible and, if so, what it would look like depending on the demographic.

Our proposed aspects are theoretical; thus, they mostly provide a conceptual framework for practitioners to think about OSS sustainability, whether they are adopters or contributors. Researchers can build upon these aspects to create more specific guidance and/or metrics to complement them, working as tools that can be directly used by practitioners to evaluate OSS sustainability.

The SMS results can also be useful for future mappings and/or reviews in the OSS sustainability area. By including all of the aspects as synonyms of sustainability, it is possible to locate studies that are sustainability adjacent but do not address it by name directly. This includes important, older, sustainability-related studies from periods when the sustainability terminology was very rarely used.

2.4.3 Threats to Validity

Errors could occur during SMS execution, especially during study selection. This could compromise the SMS coverage. To address this possible bias, we developed a detailed protocol based on (PETERSEN *et al.*, 2015), validated by a senior SMS expert, and resolved disagree-

ments through consensus meetings. Similarly, the construction of categories, which requires the interpretation of fragments, may introduce mistakes. This was mitigated by multiple iterations and additional consensus meetings.

Another threat concerns the generalizability of our results, as most analyzed studies stem from academic initiatives, with limited representation from industry or OSS communities. This restricts the scope of our conclusions, which we deliberately confine to the academic context, while validation in industrial or community settings is left for future work. Additionally, our search strategy might not have retrieved all relevant studies. To mitigate this, we refined the search string through pilot studies and targeted well-known SE databases (Scopus, IEEE Xplore, ACM Digital Library). Although synonyms for “sustainability” could not be exhaustively captured, this limitation aligns with the SMS objective of clarifying the term’s meaning.

Finally, data interpretation poses a potential threat, as many findings rely on the classification of text passages. To reduce subjectivity, we held brainstorming sessions to ensure consistent interpretations of sustainability in OSS, and all extracted data was reviewed by all authors to avoid hasty or biased conclusions.

2.5 Final Remarks

OSS has become a critical foundation of today’s digital infrastructure, yet its sustainability remains an open and multifaceted challenge. In this mapping, our objective was to clarify how sustainability is understood in the OSS domain, what dimensions and aspects influence it, how these aspects are addressed in research, and the impact on the design of software systems. Our SMS put together studies from two decades of OSS and sustainability literature, revealing that while the social and technical dimensions are well-explored, the economic and especially environmental dimensions remain significantly underrepresented. The main contribution of this mapping is to provide an overview of sustainability aspects and their interrelations, which supports decisions regarding project direction as well as adoption concerns. By identifying 19 sustainability aspects across five dimensions, our results offer both a conceptual foundation and a practical reference for researchers, maintainers, and adopters who want OSS projects to be more resilient in the long term. This mapping raises awareness about the importance of introducing a systemic vision to sustain the software ecosystem upon which much of society now depends.

An Overview of the Open Source Software Ecosystem for Cloud Observability

This chapter is based on the paper “Open Source Software Ecosystem for Cloud Observability: An Overview and Trends”, authored by Gabriel Silva Fontes, Vasilios Andrikopoulos, and Elisa Yumi Nakagawa, published at the 14th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS 2026) at the 48th IEEE/ACM International Conference on Software Engineering (ICSE 2026).

Abstract

Context: With the adoption of cloud computing and the evolution of IT and software engineering practices, observability, the ability to systematically measure and monitor software systems, becomes a growing concern. Cloud computing is very heterogeneous, and thus creating, maintaining, and observing systems built on it requires integration of multiple tools. Open source software (OSS) tooling emerged as a de-facto standard in multiple areas, including cloud observability. The ecosystem formed by OSS observability tooling is large and often difficult to navigate. **Objective:** The main goal of this chapter is to present the OSS Ecosystem formed by cloud computing observability tooling, by providing an overview of the main tools and identifying trends on their integration. **Method:** We searched the literature for studies on cloud observability and used an automated named-entity recognition (NER) method to extract candidate observability tools from abstracts. We selected the OSS observability tools according to well-defined criteria. Following this, we used keyword matching on the tools’ source codes, revealing integration code, documentation, and other types of relations. **Results:** As a result, we found 45 OSS observability tools and derived quantitative measures on their relations to one another. The tools serve different roles in observability stacks, and our results serve as a proxy measure on how strongly they relate to one another. **Conclusion:** We present an overview of different tools, how frequently they appear, their functionality, and a visualization of their relations. Some trends are identified, such

as the centrality of some tools (e.g. *Prometheus*), common stacks (e.g. *ELK*, *TICK*), and outliers (*Thingspeak*). We discuss topics such as measuring relations, standardization, proprietary lock-in, among others.

3.1 Introduction

Catalyzed by the revolution brought by the public cloud and the advance of modern IT practices, such as the Site Reliability Engineering concept (BEYER *et al.*, 2016) and the DevOps movement, systems administration is shifting from operational into an engineering domain (BEYER *et al.*, 2016). With this shift, there is a need to systematically measure and monitor software systems, which is known as *observability*. Observability is a concept originating from control theory, meant to measure the degree to which a system's internal state can be inferred from its output at any point in time (NIEDERMAIER *et al.*, 2019). Cloud observability, in turn, is the degree to which (cloud) infrastructure, applications deployed on this infrastructure, and their interactions can be monitored using data such as logs, metrics, and traces (PICORETI *et al.*, 2018).

The cloud computing ecosystem is, by its very nature, a heterogeneous combination of different platforms, software, and systems. Ranging from on-premises infrastructure and traditional workloads, to a multitude of different cloud providers and cloud-native container orchestration, plus dozens of different configuration management tools (with varying degrees of compatibility), the list goes on and on. In this scenario, it is challenging to create, maintain, and observe systems that rely on such a diverse ecosystem. This diversity makes the interoperability requirement critical (LOUTAS *et al.*, 2011) (LEWIS, 2013). Open Source Software (OSS) is a natural fit for this, due its effect on standardization and commodification (LINDEN *et al.*, 2009). There are many cases where an OSS' potential for interoperability helped it become a dominant force in software engineering tooling, including Docker/OCI, Kubernetes, Linux, Git, LSP, among others. Cloud observability is no different, as we will discuss further in this chapter.

Just as the cloud systems that it aims to observe, the OSS ecosystem for cloud observability is similarly diverse and heterogeneous. This ecosystem includes tools for tracing, benchmarking, logging, aggregation, storage, visualizations, alerting (BEYER *et al.*, 2016), with many different tools used in different contexts and business domains. Understanding this ecosystem becomes crucial for both researchers in the cloud computing domain, as well as practitioners looking to build more reliable and observable cloud-based systems. This understanding, however, is not just about which tools there are out there, but how they integrate and which role they play in an observability stack.

While there is some non-academic effort to better understand subsets of this ecosystem (Cloud Native Computing Foundation, 2025b), current literature lacks a wider study on the OSS ecosystem for cloud observability. With that in mind, the main goal of this chapter is to present

an overall view of the OSS tooling ecosystem used for cloud observability. To achieve our goal and identify trends in this field, we defined two research questions (RQs):

- **RQ1:** What are the most relevant OSS tools in cloud observability stacks?; and
- **RQ2:** How are OSS tools combined to form cloud observability stacks?

Following this, we searched the scientific literature and located 3068 studies, from which, through an automated extraction followed by manual selection, we identified 45 OSS observability tools. As a main result, we observed the centrality of some tools, and tools that cluster together. We intend that the overview presented in this chapter can bring implications for both practitioners and researchers.

The remainder of this chapter is structured as follows: Section 3.2 outlines the research method; Section 3.3 reports the main results; Section 3.4 presents our main findings, threats to validity, and future work; Section 3.5 concludes the chapter.

3.2 Research Method

Figure 6 provides an overview of the research method and the reproduction package files that correspond to each step. The reproduction package is made available both in the supplementary material and online¹ to replicate the research method and results reported in this chapter.

3.2.1 Search Phase

To build our initial set of tools, we used a large population of research studies as basis. For that, we searched Scopus² with the following query:

```
1: TITLE-ABS-KEY (  
2:   cloud AND tool AND (  
3:     observability OR logging OR monitoring  
4:   )  
5: ) AND LIMIT-TO (SUBJAREA , "COMP")
```

Our goal includes, but is not limited to, finding more niche tools and thus benefits from a large and diverse sample size. To avoid missing relevant tools, this search query intentionally

¹ <<https://github.com/Misterio77/OSS-Cloud-Observability>>

² <<https://scopus.com>>

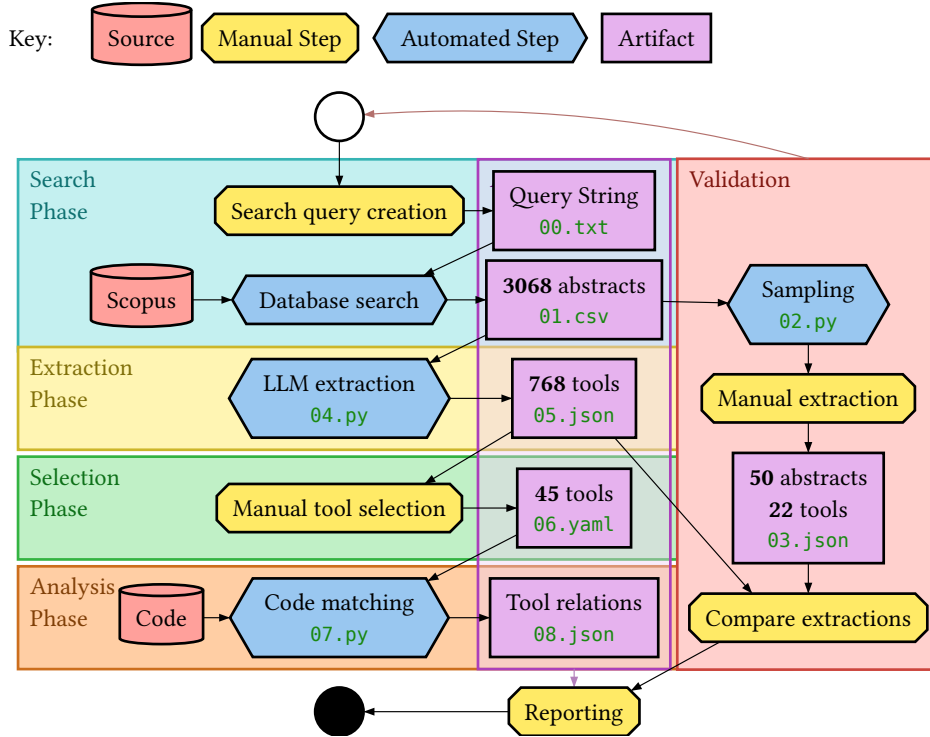


Figure 6 – Overview of research method

did not try to exclude research software nor proprietary software, thus leaving this filtering for manual selection, as described in Section 3.2.3.

The search resulted in a set of **3068** studies, which were exported into a CSV file in the following format: "Title", "Year", "DOI", "Link", "Abstract". This data is available in the reproduction package as `01-scopus-search-results.csv`.

3.2.2 Extraction Phase

This is a problem in the class of Named-entity Recognition (NER) (PAKHALE, 2023), albeit a single-class one. As the intent is to extract tool names mentioned in the studies, a method that can be aware of context, such as BERT or an LLM, is ideal. For this extraction, due to ease of access and previous experience, we decided to use an LLM to extract the tools from abstracts.

The reason for using the abstracts alone, besides ease of automation and availability, is that an abstract can fit together with the LLM's prompt in most models' context windows, drastically reducing hallucinations and costs when compared to the full texts. The main risk from this strategy is missing tools that only appear in the full text; hence, we mitigated it in two ways: (i) by using a large amount of studies; and (ii) by calibrating the prompt using a validation set, as discussed next.

3.2.2.1 Validation sampling and extraction

A validation set of **50** manually extracted abstracts was created. This set was used to estimate precision and recall of the overall dataset and to calibrate the LLM prompt used for extraction. The first attempt was by randomly sampling abstracts. After analysis, it was evident that the negatives (i.e., abstracts not containing any observability tool names) made up around 80-90% of the corpus, making the data very sparse. To get a set with a higher share of positives, we decided to use a combination of keyword spotting (by matching the text with a few known observability tools³) and random sampling. The set is composed of **42** studies originating from keyword match and **8** from random sampling. The script that reproduces this step is available in the reproduction package as `02-validation-set-script.py`. The authors then manually extracted tools from these abstracts, inserting the annotations into the dataset, which is available in the reproduction package as `03-validation-set-results.json`. The validation set was used to evaluate and refine the study, including the search query and LLM prompt, as discussed next.

3.2.2.2 LLM extraction

The extraction method consists of feeding each study's abstract to an LLM (in this case, DeepSeek-V3 (DEEPSEEK-AI, 2024)), with the following prompt:

“ You will receive a paper abstract that relates to cloud observability/monitoring/logging/tracing tools. Enumerate all observability/monitoring/logging/tracing tools mentioned in it. Respond with the names separated by comma: tool1, tool2, tool3. If there are no relevant tools, reply with an empty string. ”

This prompt was the result of several rounds of evaluation and calibration. By using the LLM to extract tools from the validation set, we achieved recall **97%**, precision **69%**, and F_1 **80%** (66 true positives, 30 false positives, and only 2 false negatives). The key metric here is the recall, as this LLM extraction is then used for a manual selection, so it is crucial that false negatives are minimized, even if this leads to more false positives.

From the full set, the LLM extracted a total of **768** candidate tools, originating from **774** studies (**25%** of **3068** total studies). This step can be fully reproduced via the script `04-llm-extraction-script.py`, and its results are available in `05-llm-extraction-results.json`. These tools were then manually selected, as detailed in Section 3.2.3.

3.2.3 Selection Phase

We manually examined the 768 candidate tools and the studies they appeared on and selected the tools according to the following selection criteria:

³ grafana, prometheus, graphite, opentelemetry, elasticsearch, fluentd, kibana, logstash, jaeger, influxdb, ceilometer

- Inclusion Criteria (IC):
 - **IC1:** The tool is/was a piece of Free/Open Source Software, meaning licensed under one of the OSI⁴'s or FSF⁵'s approved software licenses; and
 - **IC2:** The tool is used/discussed as part of a cloud observability solution.
- Exclusion Criterion (EC):
 - **EC1:** The tool is a piece of research software and not otherwise available as an actual FOSS project on the web or mentioned anywhere else on the web besides the studies that introduce it.

By applying this criteria, we selected a total of **45** tools. Besides the tool name and studies it originates from, we also annotated each tool with:

1. A list of its relevant software repositories (e.g., Git, SVN), located via web searches.
2. Its main functions/roles, also located via web searches, and organized roughly based on (KOSIŃSKA *et al.*, 2023):
 - a) instrumentation (Instrumentation): allows systems to export observability data (e.g. libraries, exporters).
 - b) collection/processing (Data Collector): aggregates, processes, and collects data.
 - c) storage (Data Backends): long-term storage, querying.
 - d) visualization/alerting/analysis (Analysis/Visualization): understanding of the system, root cause analysis, discovering “unknowns”.

This data is available in the reproduction package as `06-tool-selection-manual.yaml`. This is part of RQ1 and further explored in Section 3.3.1.

3.2.4 Analysis Phase

Our main goal in this phase is to find the relations between the tools, to answer the RQ2. For that, we downloaded their source code, and, for each tool, programmatically matched occurrences of every other tools' names in its codebase. This step can be reproduced using `07-tool-code-occurrences-script.py` and its results are available as `08-tool-code-occurrences-results.json`.

These numbers represent how much a tool's codebase is aware of and/or coupled to each other tool, and includes integration features, documentation, tests, dependencies, etc. This is part

⁴ <<https://opensource.org/licenses>>

⁵ <<https://gnu.org/licenses/license-list.html>>

of RQ2 and is further explained in Section 3.3.2. A deeper study on the nature of each individual relation is out of scope for this chapter, but we briefly discuss this in Section 3.4.3.

3.3 Results

This section focuses on answering the two RQs defined in Section 3.1.

3.3.1 OSS tools in Cloud Observability Stacks

We selected **45** tools, which can be tracked back to the abstracts of **111** studies in our set (4% of **3068** from the initial search). Table 3 contains the full set of selected tools, the amount of abstracts they appear on, and our annotations on the main functions each of them provide in an observability stack (as per (KOSIŃSKA *et al.*, 2023)). As it can be seen from the table, there is a long tail of tools with presence in a single study each, while a few tools such as *Thingspeak* and *Prometheus* dominate the discourse.

Table 3 – Selected Tools

Tool	Studies	Roles	Tool	Studies	Roles
thingspeak	21	collection, storage	collectd	1	instrumentation, collection
prometheus	18	instrumentation, collection, alerting	collectl	1	collection
grafana	15	visualization, alerting	elastic beats	1	instrumentation
nagios	10	collection, alerting	flume	1	collection, processing
elasticsearch	7	storage	jaeger	1	collection, visualization
kafka	6	processing	joularjx	1	instrumentation, collection
zabbix	6	collection, alerting, visualization	kapacitor	1	processing, alerting
ceilometer	4	collection	logstash	1	collection, processing
ganglia	4	collection, processing, visualization	monasca	1	storage, processing
kepler	4	instrumentation	netdata	1	collection, alerting
kibana	4	visualization	net-snmp	1	collection
snort	4	collection	ntopng	1	instrumentation, collection
thingsboard	4	collection, processing, visualization, analysis	opentsdb	1	storage
influxdb	3	storage, processing	powerjoular	1	collection
wazuh	3	instrumentation, alerting, visualization, collection, analysis	sysdig	1	instrumentation
xdmod	3	instrumentation, collection, analysis	syslog-ng	1	collection
fluentd	2	collection	systemtap	1	instrumentation
opentelemetry	2	instrumentation, collection	telegraf	1	processing, collection
scaphandre	2	instrumentation, collection	tetragon	1	collection, analysis
skydive	2	collection	tracecompass	1	visualization, analysis
cacti	1	collection, visualization	zenoss	1	collection, visualization
cadvisor	1	collection	zipkin	1	storage, visualization, analysis
chronograf	1	visualization			

Figure 7 visualizes how frequently each tool appears, in number of studies. Notoriously, *Prometheus* and *Grafana* are very frequently mentioned in the studies, which correspond to our expectations and industry experience, where the two are frequently combined for a basic metric+visualization stack. *Thingspeak* being highly mentioned, while unexpected, is possibly an indicator that IoT research frequently intersects with cloud computing.

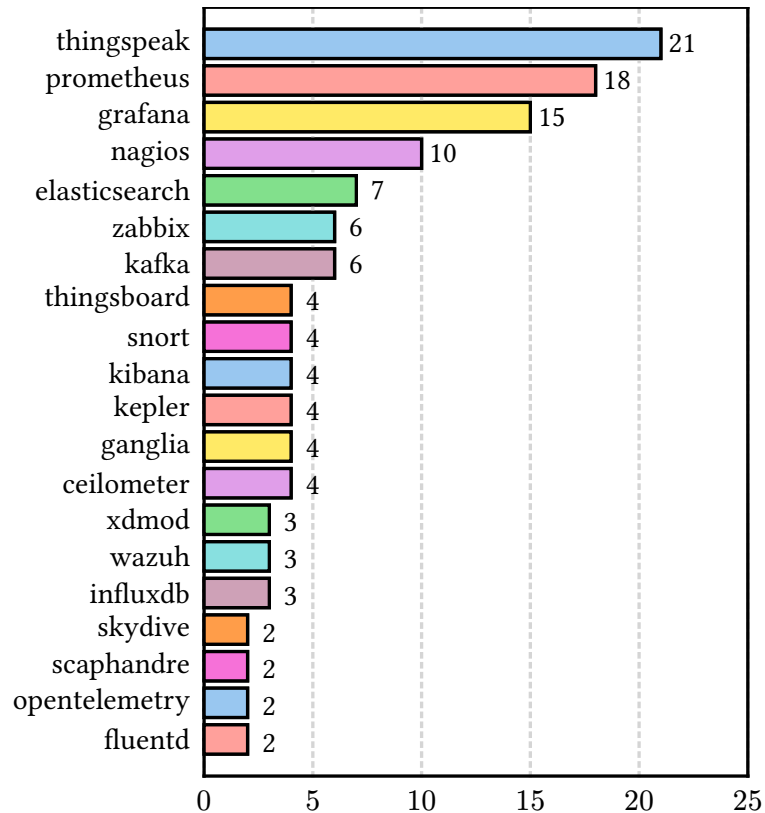


Figure 7 – Number of studies mentioning each tool (Tools appearing on a single study were omitted)

Figure 8 shows how frequent each role (as defined in Section 3.2.3) is. Some tools have multiple roles (e.g. *Grafana* has both visualization and alerting). We can see a very high frequency in the collection role; besides dedicated collectors, most instrumentation and processing tools seem to have some sort of collection/aggregation mechanism built-in.

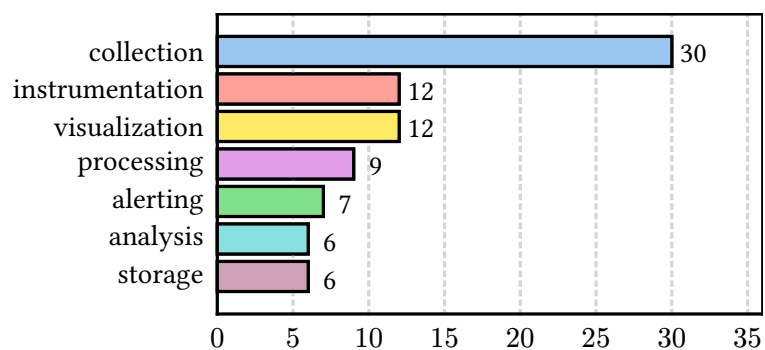


Figure 8 – Number of tools per role (Some tools have multiple)

Figure 9 shows the yearly distribution of studies with at least one selected tool. We can see that there is a growing trend, possibly indicating the increased interest in the area. Our data contains studies from up until October 2025, thus 2024 studies are slightly more present than 2025 in the graph.

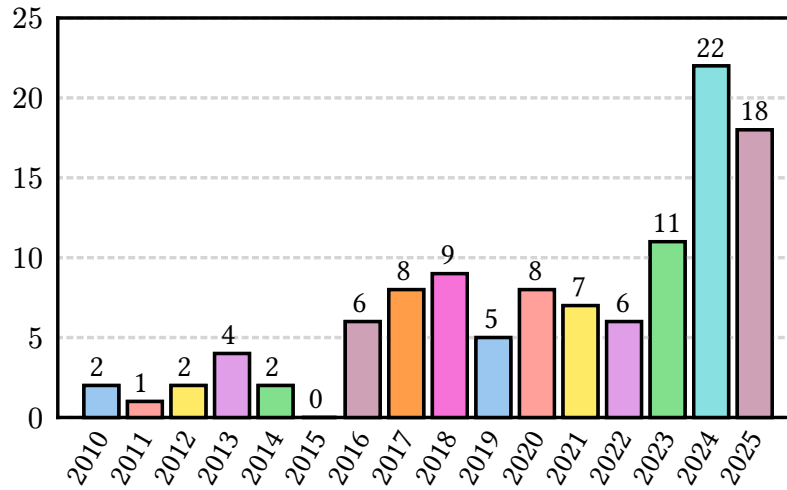


Figure 9 – Yearly study distribution

Answering RQ1: Prometheus is ubiquitous in the cloud observability ecosystem. Nagios, Grafana and ElasticSearch are also frequent appearances. Thingspeak is an interesting outlier, representing the IoT sub-ecosystem. Collection functionality is the most frequent, followed by instrumentation and visualization.

3.3.2 Combinations of OSS Tools in Cloud Observability Stacks

To better visualize the relation data, we conducted a network analysis in it, with the results shown in Figure 10. For this figure, we represent each tool as a node.

The graph edges are directed and weighted, serving as visualization of how much a tool X relates to another tool Y. This is obtained, as described by Section 3.2.4, through matching Y's name on X's codebase, and represents relations such as integration features, comments mentioning a borrowed snippet, documentation (usually comparing an alternative tool or documenting integration), tests, and usage as libraries. As the codebase sizes vary, the edge weights are normalized with the other edges originating from the same node. Edges whose weights were lower than 0.05 (i.e. 5% of all keyword matches that codebase has) were hidden from the visualization, for improved visibility.

The node size is derived from the sum of the weights of connections with that node as destination, interpreted as how much important that the tool is to the ecosystem. The node colors are a visualization of community clustering, that groups tightly connected nodes together.

We can see in the figure how many tools relate to *Prometheus*; due to its format being the dominant one for metrics, most other tools export metrics that are compatible with *Prometheus*.

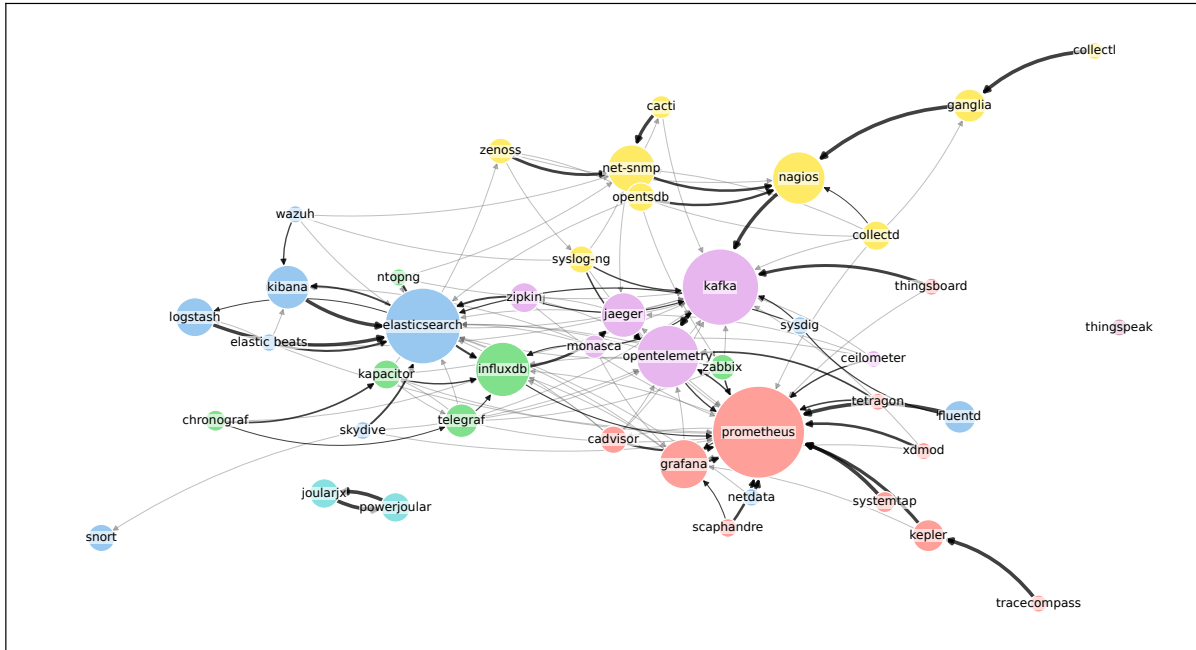


Figure 10 – Relations between tools, clustered and weighted by relative code occurrence

Thingspeak is very disconnected from the other tools, which hints that IoT tooling tends to be more standalone. *JoularJX* and *Powerjoular*, a pair of tools built together to, respectively, collect and aggregate power consumption data, also appear isolated yet tightly coupled together. We can clearly see some ecosystems forming in the figure; the *ElasticSearch+Logstash+Kibana* ELK stack and the *Telegraf+InfluxDB+Chronograf+Kapacitor* TICK stack are some obvious ones. The directionality of the relations is also representative of how the tools behave, and *Grafana* relates heavily to *Prometheus* (it has a *Prometheus* datasource built-in ([Grafana Labs, 2025b](#))), but not the other way around (i.e. *Prometheus* knows almost nothing about *Grafana*).

Answering RQ2: some common stacks appear in this analysis, such as ELK and TICK. Some tools are highly integrated in the ecosystem, and Prometheus is a central piece. Thingspeak is an outlier and shows that IoT tends toward standalone solutions.

3.4 Discussion

This section discusses our main findings, the threats to the validity of the analysis, and future work.

3.4.1 Main Findings

The main findings resulting from our investigation of OSS tools for cloud observability are:

- **Quantitative code matching can be a good proxy to find relations between codebases.** Our approach of matching the names of each tool on other tools' codebases worked. Figure 10 clearly shows clustering between well-known combinations: ElasticSearch+Logstash+Kibana (ELK stack), *Telegraf+InfluxDB+Chronograf+Kapacitor* (TICK stack). We can also see library usage (e.g., *OpenTelemetry*), how heavily the most ubiquitous tools (*Prometheus*) are referenced, as well as some disconnected clusters (e.g., *Powerjoular+JoularJX*, *Thingspeak*). Notably, our approach is very accurate in modeling directed relations, such as *Grafana* integrating with *Prometheus* (Grafana Labs, 2025b), but not the other way around. Finally, our approach is heavily automated, and thus scales much better than the alternatives that involve manual labeling or more qualitative analysis. It should be interesting to apply these techniques to different Software Ecosystems.
- **Some tools are de-facto standards, bringing interoperability but risking over-reliance.** It is interesting to see how *Prometheus* has become so ubiquitous that there are references to it in the code of basically every single cloud observability tool that exists. If a software exports metrics, it is probably in the *Prometheus* exposition format. Although *Prometheus*'s community-based governance brings some safety, over-reliance on a single implementation can be problematic, thus bringing the need of standardization. *OpenTelemetry* has recently emerged as a more standardized framework for different aspects of observability (including compatibility with the *Prometheus* exposition format) (OpenTelemetry, 2024), and it is interesting to see adoption grow. The connection graph we built can be a helpful metric to determine which OSS tools are becoming critical, thus should be prioritized for standardization and/or alternative implementations by OSS communities and funding agencies.
- **OSS and standardization can challenge proprietary lock-in.** Our data extraction step yielded some proprietary software (e.g. *CloudWatch*, *AzureWatch*), most of them specific to a single cloud vendor. These were, however, often accompanied by OSS tools, a reminder that some of the OSS tools are so ubiquitous that vendors are effectively forced to implement support for them, further showing the strength OSS has in standardization and interoperability. Practitioners should consider prioritizing OSS when adopting observability tooling that needs to interoperate, for example, in multi-cloud scenarios.
- **The line between an observability tool and a tool that can be used for observability can be thin.** During our manual selection step, some tools were difficult to classify. For example, *Kafka* is not necessarily built for observability solutions, but its usage is so frequent that it becomes clear that it should be labeled as such. But what about other message queues? Or storage systems? Further research is required to effectively draw this line.

3.4.2 Threats to Validity

As with any secondary studies, threats to validity must be considered as well as mitigation actions, as discussed below:

- **Not all tools are mentioned in scientific literature.** There are a few tools the authors know about from their exposure to cloud observability solutions that did not appear in any of the abstracts our search yielded. This includes *Mimir*, *Loki*, *Graphite*, and *Thanos*. This means that including other datasources and analysis techniques could provide a more complete set of tools. This is an acknowledged limitation of our analysis.
- **Sparsity of dataset.** The abstract dataset is very sparse, making validation harder. Only 4% of the total corpus includes at least one tool from the final selection. We mitigated this by building a validation set that included both known tools as well as randomly sampled studies.
- **LLM bias.** Although we calibrated the prompt to minimize false negatives, the LLM can be biased towards more well-known tools. We mitigated this by inspecting the abstracts of all 774 extracted studies (i.e. studies from which the LLM extracted at least one tool), this same inspection is, however, not possible in the negative population, as it is very large. The mitigation provided by the validation set (Section 3.2.2.1), with a high recall value, is considered sufficient.
- **Abstracts may not contain every tool relevant to the work.** Using only the abstracts (as opposed to the fulltexts) can leave out relevant tools. This is somewhat mitigated by the large corpus of studies we used, and is a tradeoff we accepted to build a more automated research method.
- **Code matches can contain false positives.** Some codebases, such as ElasticSearch's, contain files such as wordlists. This is mitigated by making the weights relative and discarding less relevant edges. Future research could involve additional heuristics.
- **Code matches can contain transitive relations.** Transitive dependencies/integrations are also frequently matched, which might be an issue if the intention is to only model direct ones. This can be partially mitigated by excluding lockfiles. We decided to not qualitatively differentiate the nature of each relation, so this is considered out of scope for this chapter.
- **Human error during selection.** As with any classification based on human decisions, our selection process could contain errors or be biased. This was mitigated by consensus between the authors and by carefully researching each candidate tool.

3.4.3 Future Work

Based on the results of our investigation, we can suggest the following future work:

- **Qualitative analysis on the relations.** Our approach with the relations between tools was very quantitative, with no regard for the nature of the code nor any attempt to filter false positive matches. By conducting a more qualitative analysis of each individual relation identified, one could gain a better understanding of the relation between tools, and how good of a proxy the quantitative analysis is.
- **Other datasources besides academic studies.** Although useful for casting a wide net, academic studies might not be enough to locate all tooling, specially emerging ones. Other methods such a multivocal literature reviews to find the set of tools might be interesting to explore.
- **Applying our method to other ecosystems.** The code keyword matching approach is simple, very scalable, and seems to provide good results. One could apply this technique to other ecosystems.
- **A better definition of observability tooling is needed.** Some tools are used in observability but do not define themselves as observability tools. Some better heuristic to make this classification could be explored.
- **Other heuristics for finding relations.** Research on more heuristics for the relations can be interesting. For example, by analyzing the common contributors between two projects, one could measure integration and/or cross-pollination between different software projects.

3.5 Conclusions

As IT operations shift into an Engineering activity, Observability also grows as an important topic for SE research. The objective of this chapter was to provide an accurate starting point to researchers exploring the OSS Observability ecosystem. Our analysis identifies a selection of 45 OSS tools, a categorization on their functionality, and a quantitative measure of the relations between one other. A few tools are shown to have a central role in the ecosystem, such as *Prometheus*, others appear clustered in stacks, such as *ELK* (*ElasticSearch*, *Logstash*, *Kibana*) and *TICK* (*Telegraf*), *InfluxDB*, *Chronograf*, *Kapacitor*). We also showcase some outliers, such as *Thingspeak* which does not relate to any other tool we have researched, as well as tools that appear isolated as a pair, such as *JoularJX* and *PowerJoular*.

Our methodology is very automated and can be helpful for any exploratory study of a Software Ecosystem where source code is available for analysis. This sort of overview can be helpful for practitioners to locate additional components their observability stack might be missing, as well as to discover alternatives for tools they are using.

A Sustainability Analysis of Open Source Tools for Cloud Observability

This chapter is based on the manuscript “How Sustainable Is Open Source Cloud Observability? A Multi-Aspect Analysis of the Ecosystem”, authored by Gabriel Silva Fontes, Vasilios Andrikopoulos, and Elisa Yumi Nakagawa, currently in preparation for journal submission.

Abstract

Context: Open Source Software (OSS) projects heavily support cloud infrastructure, but repository activity and ecosystem prominence do not show whether the projects behind these components can continue fulfilling their roles. **Objective:** We investigate sustainability strengths, risks, and evidence gaps in OSS cloud observability projects, recurring patterns among them, and what comparing their profiles with ecosystem centrality reveals. **Method:** We combined 19 sustainability aspects from our previous systematic mapping study with tool relations from our study of the OSS cloud observability ecosystem. We selected the 14 tools in the graph’s top 30% by centrality and evaluated each aspect using public evidence collected in June–July 2026. We assigned qualitative ratings and confidence and grouped the resulting profiles by their current stewardship. **Results:** Technical activity, functionality, and maintainability were the most consistent strengths, while risks appeared in governance, knowledge concentration, licensing, and economic support. Foundation-hosted projects had the most uniform social and technical profiles. Vendor-controlled projects generally combined substantial support with governance, licensing, and product-boundary risks, while independent and research-led projects showed funding and succession constraints. Personal Motivations remained unassessed for every tool. Centrality represented incoming reference exposure, both of implementations and interfaces. **Conclusion:** Ecosystem position and project sustainability describe different properties. Reading centrality together with multidimensional profiles exposes project-level conditions and ecosystem exposure

that either view misses in isolation, while preserving the qualitative evidence needed to interpret them.

4.1 Introduction

Open Source Software (OSS) has become a critical part of the infrastructure on which modern software systems are built. The same code can be reused by individuals, companies, and public institutions, so projects can support systems far beyond the organizations that maintain them. [Eghbal \(2016\)](#) compares this shared digital infrastructure to roads and bridges, whose continued value depends on regular maintenance. [Greenstein and Nagle \(2014\)](#) estimate that Apache accounts for substantial unmeasured software capital, while [Berlinguer \(2023\)](#) describes OSS as central to cloud computing and to the standardization and collaboration that take place around it.

Source-code simply being available does not, however, guarantee that a project will continue fulfilling its role. A project may depend on very few contributors, concentrate governance and relicensing authority in one company, or generate widespread commercial value without ever receiving contributions in return. Repository activity and release cadence can provide useful proxies for project health, but do not show who can make decisions, whether project knowledge can be transferred to new contributors, how work is financed, or what happens when the interests of a sponsoring organization and the community clash. OSS health research similarly describes project viability through a broad set of socio-technical characteristics and notes that projects cannot always be understood separately from their dependency networks ([LINÅKER *et al.*, 2022](#)).

Software sustainability offers a multidimensional view of these conditions. In the Karlskrona Manifesto, [Becker *et al.* \(2015\)](#) divide sustainability into social, technical, individual, economic, and environmental dimensions. In our previous systematic mapping study, presented in Chapter 2, we examined 111 studies from the OSS literature and derived their project sustainability concerns into 19 aspects across these five dimensions. These aspects cover contributor attraction and retention, governance, openness, development activity, maintainability, licensing, personal knowledge and motivations, financial resources, economic activity, reinvestment, and environmental requirements. They describe interconnected conditions and states, rather than independent variables. For example, governance can affect contributor renewal and licensing, while financial support can sustain technical work without creating a community with distributed authority.

Cloud observability provides a concrete ecosystem in which to apply this view. Observability stacks combine OSS tools for instrumentation, collection, processing, storage, visualization, and alerting to observe cloud systems through metrics, logs, and traces. In our previous study of this ecosystem, presented in Chapter 3, we identified 45 OSS tools from the literature

and measured the relations between them through references to one another found in their source-code repositories. The resulting graph includes widely referenced interfaces, recognizable stacks, and a few small isolated components. We derived a centrality measure, which represents prominence within this ecosystem, but does not address if the projects themselves are sustainable.

In this chapter, we combine these two methodological inputs in an exploratory multiple-case assessment. We select the 14 tools in the top 30% of the observability graph by centrality and evaluate them across the 19 sustainability aspects using public evidence. For each aspect, we report a qualitative rating, its confidence, and an evidence summary. Our main objective is to investigate the sustainability strengths, risks, and evidence gaps that become visible when project-level conditions are read together with their ecosystem position.

Based on this objective, we define the following research questions (RQs):

- **RQ1:** What strengths, risks, and evidence gaps appear in selected projects' sustainability profiles?
- **RQ2:** Which recurring sustainability patterns appear across the selected projects?
- **RQ3:** What does comparing sustainability profiles with ecosystem centrality reveal?

As a result, we organize the tools into four descriptive groups according to their current stewardship: foundation-hosted, vendor-controlled, independent, and research-led software. We observe that projects with similar levels of activity can rely on different governance, contributor, licensing, knowledge, and funding conditions. We also notice that centrality sometimes represents not just the reach of reference implementations, but of their interfaces (and thus compatible alternatives) as well.

The remainder of this chapter is structured as follows. Section 4.2 presents external concepts and related work on software sustainability, OSS health, and project economics. Section 4.3 presents our two previous studies as methodological inputs and describes tool selection, evaluation criteria, evidence collection, and scoring. Section 4.4 reports the comparative profiles and presents individual cases. Section 4.5 discusses the tools, sustainability framework, centrality metric, implications, and threats to validity. Section 4.6 concludes the chapter.

4.2 Background and Related Work

4.2.1 Software sustainability and OSS health

Software sustainability is broader than the maintainability of a codebase. In the Karlskrona Manifesto, [Becker et al. \(2015\)](#) define sustainability as the capacity of a system to endure and describe five dimensions: social, technical, individual, economic, and environmental. The technical dimension concerns the longevity and evolution of software and infrastructure; the

individual and social dimensions concern people, communities, and organizations; the economic dimension concerns capital and value; and the environmental dimension concerns effects on nature. The dimensions provide different views of the same system and are useful for separating concerns without treating them as interchangeable.

In OSS research, sustainability overlaps with the general idea of project health. [Linåker et al. \(2022\)](#) define OSS health as a project's capability to remain viable and maintained over time. Their framework, derived from a review of 146 papers, identifies 107 health characteristics under 15 themes covering actors, software, and project orchestration. The characteristics include contributor retention, financial support, knowledge concentration, documentation, quality assurance, licensing, and governance. They also distinguish characteristics at the project level from those at the level of the surrounding network, since OSS projects are connected through dependencies and cannot always be understood in isolation. This breadth makes OSS health related to our understanding of sustainability.

4.2.2 Project-health metrics

Several initiatives use repository data to make subsets of OSS health measurable. The Community Health Analytics in Open Source Software (CHAOSS) project develops common metrics and tools for examining OSS communities. In their account of the project's first four years, [Goggins et al. \(2021\)](#) argue that activity measures are useful but insufficient when read without project context. The same decline in activity can represent contributor loss in one project and stabilization in another. They identify comparison, transparency, trajectory, and visualization as principles for interpreting metrics, and emphasize that project-health narratives require both data and knowledge of the project.

The OpenSSF Scorecard provides a more specific automated assessment that focuses on security. It evaluates repository evidence such as code review, branch protection, automated dependency updates, vulnerability handling, CI tests, and security policies. In an ecosystem-level evaluation, [Zahan et al. \(2023\)](#) found that several checks required ecosystem-specific interpretation, could return inconclusive results, or did not recognize alternative practices. Scorecard is therefore useful evidence about visible security and related development practices, but its aggregate score does not capture the social, individual, economic, and environmental conditions of OSS sustainability. In our assessment, we use repository metrics and the OpenSSF Scorecard as evidence for specific aspects rather than as overall project-health scores.

4.2.3 OSS governance and economic models

The economic organization of OSS also cannot be inferred simply from whether its source code is available. In his evolutionary account, [Berlinguer \(2023\)](#) distinguishes an earlier phase centered on self-organized communities from more recent arrangements that combine

commons and markets, while noting that communities and companies coexist throughout this history. These hybrid arrangements include shared infrastructure collectively sustained by its users, commercial activity, and ecosystems created around an open platform. Consequently, substantial economic value may be generated around a project without necessarily becoming resources controlled by the project itself. We address this distinction in our assessment.

Commercial OSS can follow different models. Companies may sell hosting, maintenance, support, consulting, or proprietary features; these may be combined with dual-licensing or open-core strategies (POPP, 2023; RIEHLE, 2012). In the single-vendor model described by Riehle (2012), one company retains control of the project and the intellectual property needed to offer the software under different licences. Copyright assignment or relicensing rights granted through Contributor License Agreements can preserve this asymmetric ability even when external contributions are accepted (RIEHLE, 2012). Monaco (2023) describes models in which companies fund a shared innovation because it creates opportunities for products and services rather than serving as their exclusive competitive advantage. With this, open-core and dual-licensing arrangements create different relations between community contributions, company control, and the value returned to the project. These distinctions motivate our separate analysis of governance, licensing, financial resources, external value capture, and reinvestment.

Prior work thus offers detailed views of wider software sustainability, OSS health characteristics, repository metrics, security practices, and OSS economic organization. We apply these complementary perspectives in a structured exploratory assessment of projects selected from our cloud observability ecosystem graph. Rather than replacing existing health metrics, we treat them as evidence alongside qualitative public sources and examine how project-level sustainability conditions relate to ecosystem position.

4.3 Research Method

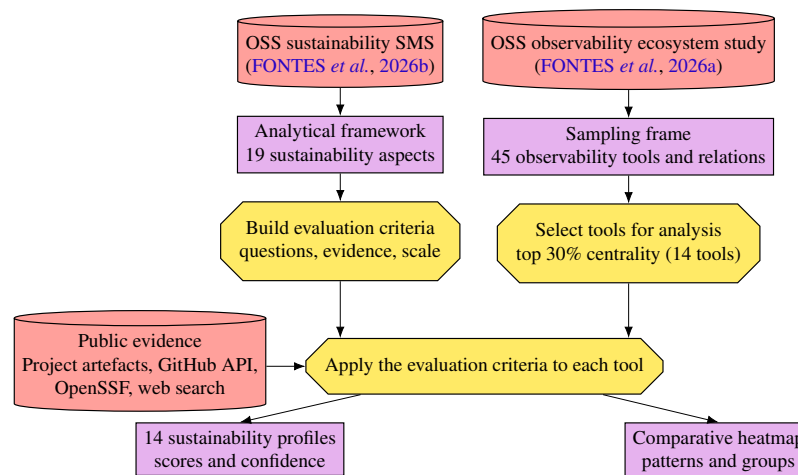


Figure 11 – Overview of the research method and the inputs adopted from our prior studies

4.3.1 OSS Sustainability Aspects

Our first methodological input is the systematic mapping study presented in Chapter 2, in which we investigated how sustainability is understood and addressed in OSS research. The mapping followed the planning, conduct, and reporting process proposed by [Petersen *et al.* \(2015\)](#) for systematic mappings. Its search covered Scopus, ACM Digital Library, and IEEE Xplore. The search yielded 2,428 studies, of which 1,946 unique studies remained after deduplication. The first selection, by title, abstract, and keyword, resulted in 333 studies of which, after full-text selection, 111 were included. A second researcher checked the selections, and disagreements were resolved by consensus.

The mapping extracted fragments in which the selected studies discussed OSS sustainability and applied inductive open coding to identify sustainability-related terms. We then merged duplicate and overlapping codes and factored them into a set of minimally overlapping categories, producing the 19 aspects shown in Table 4 under the social, technical, individual, economic, and environmental sustainability dimensions from [Becker *et al.* \(2015\)](#). The aspects cover project participation and governance, development activity and maintainability, contributors' motivations and knowledge, financial support and monetization, and environmental requirements.

Table 4 – OSS sustainability aspects derived from our systematic mapping study ([FONTES *et al.*, 2026b](#))

Dimension	ID	Aspect
Social	A1	Contributor Retention
	A2	Contributor Attraction
	A3	Internal Communication
	A4	Governance Structure
	A5	Openness/Onboarding
	A6	Legitimacy
	A7	External Communication/Ecosystem
Technical	A8	Sustained Activity
	A9	Functionality/Scope
	A10	Quality Assurance
	A11	Methodologies/Practices
	A12	Architecture/Maintainability
	A13	Licensing
Individual	A14	Personal Motivations
	A15	Personal Knowledge
Economic	A16	Financial Resources
	A17	External Value Capture (renamed from External Monetization)
	A18	Reinvestment (renamed from Internal Monetization)
Environmental	A19	Environmental Requirements

For this assessment, we refine part of the economic aspect names and boundaries for

clarity. **Financial resources** is unchanged, and represents the quality (recurrence, diversification, quantity) of funding managed directly by the project’s governing body. External Monetization was renamed to **External Value Capture**, with its meaning unchanged: how actors external to the project governance monetize their involvement with the project, which may or may not be reinvested into the project. Internal Monetization was refined into **Reinvestment**, being slightly broader than the previous work’s definition: the overall financial or material support directed back into the project, including development, infrastructure, and community support. This refinement better describes different project monetization and governance shapes.

In this chapter, we thus use these 19 aspects as an analytical framework. We operationalize each aspect as an evaluation question and a set of publicly observable evidence items in Table 7. This preserves the mapping study’s multidimensional view while making it possible to apply the framework in a relatively consistent manner to the projects identified in the next subsection.

4.3.2 Identification of Open-source Tools for Observability

Our second methodological input is the study presented in Chapter 3, in which we identified the main open source tools that are part of the cloud observability ecosystem, as well as their relations. Its method is detailed in Section 3.2 and Figure 6.

The study queried Scopus for papers containing keywords, yielding 3,068 studies. The abstracts from these studies were processed through an LLM (DeepSeek V3) to obtain the names of cloud observability tools they discuss, adopt, or otherwise mention. The 768 candidate tools were analyzed according to well-defined criteria: a tool had to be open source, be used or discussed as part of a cloud observability solution, and not be exclusively research software; this yielded 45 tools. The complete list of tools, together with their functional roles and the number of studies mentioning them, is shown in Table 3 of Chapter 3.

For each selected tool, the study matched strings in its codebase to find references to the other 44 tools; the number of matches was then used to build a weighted graph of the ecosystem, shown in Figure 10 of Chapter 3. Nodes represent tools, and their sizes indicate how much incoming reference weight they receive. Directed edges indicate that the source tool’s codebase references the target tool, and their thickness is the relative amount of references. Edges with weight less than 0.05 are omitted for visualization clarity. Node colors represent community clusters identified by community detection on the graph structure.

Formally, centrality (node size) and weight (arrow thickness) are defined as follows: for each referee tool x and referenced tool y , let m_{xy} be the number of matches for y ’s name in the analyzed codebase of x . Self-matches were not counted ($m_{xx} = 0$). The study normalized each outgoing relation as $w_{xy} = \frac{m_{xy}}{\sum_z m_{xz}}$ and calculated the centrality of y as $C_y = \sum_x w_{xy}$. Thus, each source tool with at least one match contributes a total weight of 1.0 across the other tools it references, and the resulting score is a normalized weighted in-degree. The score therefore

represents prominence within the observed code-reference graph.

In this chapter, we use these 45 tools as our sampling frame and their ecosystem relations and centrality as context when interpreting the sustainability profiles.

4.3.3 Evaluation Planning

From the 45 tools presented in Chapter 3, we selected the top 30% (14 tools, rounded up) by their centrality score, as shown in Table 5. This defines a manageable census of the graph's highest-centrality tier.

Table 5 – Tools selected for sustainability evaluation, ordered by ecosystem centrality. The number of studies mentioning each tool in their abstracts is also shown for reference; data are from our observability study (FONTES *et al.*, 2026a).

Tool	Centrality Score	Number of Studies	Role
Prometheus	9.68	18	Instrumentation, collection, alerting
Elasticsearch	5.48	7	Storage
Kafka	4.85	6	Processing
OpenTelemetry	3.45	2	Instrumentation, collection
Nagios	2.85	10	Collection, alerting
InfluxDB	2.23	3	Storage, processing
Net-SNMP	2.14	1	Collection
Jaeger	1.27	1	Collection, visualization
Kibana	1.24	4	Visualization
Grafana	1.14	15	Visualization, alerting
Ganglia	1.07	4	Collection, processing, visualization
Kepler	1.02	4	Instrumentation
JoularJX	1.00	1	Instrumentation, collection
PowerJoular	1.00	1	Collection

To evaluate the sustainability of each selected tool, we adopt the 19 sustainability aspects derived in Chapter 2, organized across the five sustainability dimensions defined by Becker *et al.* (2015). For each aspect we define a general evaluation question and look for publicly observable evidence that helps answer it, as summarized in Table 7. The criteria are the same for every tool, but we interpret the evidence according to each tool's characteristics, so that a metrics collector and a visualization frontend are judged on the same aspects without being held to the same indicators.

Each aspect is rated on the scale presented in Table 6. A confidence level (High, Medium, or Low) records how reliably the available public evidence supports the rating. When public evidence is insufficient to distinguish strength from risk, we report the aspect as not assessed (N/A).

We deliberately avoid producing a total score or ranking: the aspects are not equally weighted; instead, we report per-tool sustainability profiles and compare them across the ecosystem.

Table 6 – Qualitative scoring scale for sustainability aspects

Rating	Meaning
Strong (2)	The evaluation question is answered positively by evidence of a sustained capability, practice, condition, or outcome.
Partial (1)	The question is answered only partly: the evidence exists but the observed condition is constrained, immature, intermittent, incomplete, or significantly vulnerable to concentration.
Weak/risk (0)	The question is answered negatively by direct evidence that the relevant capability or condition is absent, has broken down, or presents a current risk.
Not assessed (N/A)	The available public evidence does not support a rating; lack of public evidence by itself is not treated as weakness.

We assigned confidence independently from the rating. High confidence indicates that current, direct, and authoritative evidence supports the decision, with corroboration across artifacts or sources where relevant. Medium confidence indicates direct but incomplete evidence, reliance mainly on one source type, or uncertainty about scope or continuity. Low confidence indicates sparse, indirect, old, or conflicting evidence that is still sufficient to distinguish a rating from N/A.

Table 7 – Analysis criteria: evaluation question and observable evidence per sustainability aspect

Aspect	Evaluation question	Evidence
<i>Social</i>		
Contributor Retention	Does the project retain contributors over time?	Active maintainers, recurring contributors, contributor concentration, recent maintainer activity, continuity of core contributors.
Contributor Attraction	Does the project attract new contributors?	New contributors, first-time PRs, good-first-issue labels, contributor growth, community activity.
Internal Communication	Is project communication active and transparent?	Issue and PR discussions, mailing lists, forums, public design discussions, meeting notes.
Governance Structure	Is governance explicit, legitimate, and credible?	Governance docs, maintainer roles, decision process, foundation membership, steering committees, code of conduct.

Table 7 – continued

Aspect	Evaluation question	Evidence
Openness/ Onboarding	Can newcomers realistically contribute?	CONTRIBUTING file, developer docs, issue templates, good-first-issue labels, review responsiveness, onboarding docs.
Legitimacy	Is the project perceived as legitimate and important by its ecosystem?	Age, adoption, stars, downloads, citations, foundation/vendor use, presence in major stacks.
External Communication	Does the project cooperate with or integrate into the wider ecosystem?	Plugin systems, integrations, standards support, interoperability docs, ecosystem references, compatible exporters/connectors.
<i>Technical</i>		
Sustained Activity	Is the project actively maintained over time?	Recent releases, commits, issue/PR response, changelog, active branches.
Functionality/ Scope	Does the project continue to fill a relevant ecosystem role?	Documentation, roadmap, feature evolution, use cases, compatibility with current platforms; interpreted per tool role.
Quality Assurance	Are software quality practices visible?	Tests, CI, release checks, security policy, bug-handling process, regression practices.
Methodologies/ Practices	Are development practices organized and sustainable?	Release process, semantic versioning, RFCs/design docs, code review practices, issue templates.
Architecture/ Maintainability	Does the architecture support long-term evolution?	Modular design, plugin/exporter/datasource architecture, developer docs, code ownership.
Licensing	Is the licensing clear, stable, and compatible with adoption?	OSI/FSF-approved license, mandatory copyright attribution (i.e. CLA), relicensing history.
<i>Individual</i>		
Personal Motivations	Are contributors' personal motivations supported?	Maintainer interviews or surveys; public recognition practices and community norms provide context but do not establish contributors' motivations.

Table 7 – continued

Aspect	Evaluation question	Evidence
Personal Knowledge	Does the project support knowledge transfer and reduce dependence on lone individuals?	Documentation, mentoring, design docs, multiple maintainers, bus-factor, review practices.
<i>Economic</i>		
Financial Resources	How much discretionary and durable financial capacity does the project itself possess?	Project-controlled funds or reserves, recurring and diversified income, allocation authority, restrictions, concentration, and runway.
External Value Capture	How much economic value do external actors generate or capture around the project?	Products, hosted services, commercial distributions, consulting, training, support, and paid participation.
Reinvestment	How much financial or material support is directed back into sustaining the project?	Paid development, infrastructure, and community support.
<i>Environ.</i>		
Environmental Requirements	Does the project explicitly address environmental or resource concerns?	Energy-efficiency docs, resource-usage goals, performance efficiency, green-software concerns, resource-cost discussions.

For **Licensing**, the rating reflects the observed licence history and relicensing authority. Concentrated relicensing authority, usually through Contributor License Agreements (CLAs), scores partial, while exercising that authority to remove an open-source option scores weak, even if an open-source option is later restored. Trademarks and separately licensed commercial products do not lower the Licensing rating unless they alter the rights within the evaluated tree. We do not treat permissive licences as inherently stronger or weaker than copyleft licences.

For **Financial Resources**, a strong rating requires substantial, recurring resources, and meaningful control over allocation. Diversification strengthens it, but concentration does not necessarily prevent a strong rating when the project is a durable flagship product with strategic value to its sponsor. Resources are partial when control is constrained, temporary, opaque, or materially vulnerable to concentration; little or no accessible funding or allocation influence is

weak. **External Value Capture** is strong when a significant ecosystem generates and/or captures economic value, partial when this activity is limited, indirect, immature, or dominated by one actor, and weak when little activity is visible. **Reinvestment** is strong when substantial recurring financial or material support covers several critical project needs, even when one steward supplies it; it is partial when support is temporary, narrow, fragile, or leaves important needs unfunded, and weak when beneficiaries return little or no support. Ordinary volunteer labour does not establish Reinvestment.

4.3.4 Evaluation Execution

We scored each tool against the criteria using only publicly observable evidence, drawn from four kinds of sources. The first is the projects' artefacts: the repository's files (LICENSE, GOVERNANCE, MAINTAINERS, CODEOWNERS, CONTRIBUTING, etc), changelogs, developer documentation, and public communication channels. The second is the GitHub API, from which we read each creation date, star count, release cadence, contributor count, and merged pull-request counts. The third is the OpenSSF Scorecard API, whose security and quality score contributes to the Quality Assurance aspect. The fourth is targeted web searches, that are used to establish project history, foundation graduation, relicensing events, and significant forks. Ecosystem centrality and the per-tool study counts are taken from Chapter 3, as described in Section 4.3.2. Together these comprise the "Evidence" column of Table 7. Conflicting or limited evidence, as explained, lowers confidence.

Peer-reviewed studies provide the conceptual and methodological basis of this assessment, while repositories, governance documents, licence records, foundation pages, and vendor publications are primary sources for the project characteristics being evaluated. We prioritized official project artefacts over secondary accounts and used independent reporting for historical events. We treated organizational claims, such as revenue or product strategy, as self-reported unless they appeared in regulatory records.

The observability study provides a list of repositories for each project (FONTES *et al.*, 2026a). The first GitHub repository listed for each tool was its primary repository and supplied the more automated metrics. When a tool was distributed across several official repositories, as with Prometheus and OpenTelemetry, we also read the listed component, specification, documentation, and governance repositories. We do not include unaffiliated forks or third-party plugins in project-level activity and contributor measures; we only used them when they document an ecosystem concern, such as a significant fork.

LLM tools (GPT 5.6) supported the discovery and organization of candidate evidence sources, and assisted with profile consistency checks. The first author independently inspected, collected, and assessed the evidence, and made all rating and interpretation decisions. Unless otherwise noted, all evidence reported here was collected in June–July 2026. The deterministic inputs to the tool set (ecosystem centrality and identified tools) and the collected metrics will be

included in the replication package accompanying the eventual journal submission. Centrality and the tool list originate from the study presented in Chapter 3; the scripts and their July 2026 output will also be included in that package.

For each aspect with sufficient public evidence, we assigned a rating on the scale of Table 6 and a confidence level. We used N/A where the available sources did not support a defensible rating; in particular, public recognition mechanisms do not reveal whether contributors' personal motivations are supported.

After completing the ratings, we grouped the tools by current stewardship: foundation-hosted, vendor-controlled, independent, and research-led software. We assigned the groups from each tool's current institutional home and decision authority: neutral foundation hosting, control by one company over project strategy or the open/commercial boundary, no visible current institutional controller, or investigator-led stewardship tied to an active research programme, respectively. When these overlapped, the documented decision process took precedence over paid-contributor concentration or historical origin. The groups organize the comparison and did not affect the ratings.

The full per-aspect ratings, confidence levels, and evidence behind each are given per tool in the Appendix B and will also be included in the replication package accompanying the eventual journal submission. The consolidated information is shown in the comparative heatmap (Table 8) in the next section.

4.4 Results

4.4.1 Comparative overview

Table 8 – Sustainability profiles of the 14 tools, grouped post hoc by current stewardship model.

Tool	A. Foundation-hosted					B. Vendor-controlled					C. Independent		D. Research-led	
	Prometheus	OpenTelemetry	Kafka	Jaeger	Kepler	Elasticsearch	Kibana	Grafana	InfluxDB	Nagios	Net-SNMP	Ganglia	PowerFoular	JoularIX
Centrality Appendix table	9.68 10	3.45 11	4.85 12	1.27 13	1.02 14	5.48 15	1.24 16	1.14 17	2.23 18	2.85 19	2.14 20	1.07 21	1.00 22	1.00 23
<i>Social</i>														
Contributor Retention	2	2	2	2	1	2	2	2	2	1	1	0	1	1
Contributor Attraction	2	2	2	2	1	1	1	2	1	1	1	0	1	1
Internal Communication	2	2	2	2	2	2	2	2	2	1	1	1	1	1
Governance Structure	2	2	2	2	2	1	1	1	1	0	1	0	0	0
Openness/Onboarding	2	2	2	2	2	1	1	2	1	1	1	1	0	1
Legitimacy	2	2	2	2	1	2	2	2	2	1	2	1	1	1
External Communication	2	2	2	2	2	2	2	2	2	1	2	1	1	1
<i>Technical</i>														
Sustained Activity	2	2	2	2	2	2	2	2	2	1	2	0	1	2
Functionality/Scope	2	2	2	2	1	2	2	2	2	1	2	1	2	2
Quality Assurance	2	2	2	2	2	2	2	2	2	1	1	0	0	1
Methodologies/Practices	2	2	2	2	2	2	2	2	2	1	1	0	1	1
Architecture/Maint.	2	2	2	2	2	2	2	2	2	1	1	1	2	2
Licensing	2	2	2	2	2	0	0	1	1	2	2	2	2	2
<i>Individual</i>														
Personal Motivations	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Personal Knowledge	2	2	2	1	1	2	2	2	2	1	0	1	0	1
<i>Economic</i>														
Financial Resources	2	2	1	1	1	2	2	2	1	1	0	0	0	0
External Value Capture	2	2	2	1	1	2	2	2	1	1	2	0	0	0
Reinvestment	2	2	1	2	1	2	2	2	2	1	0	0	1	1
<i>Environmental</i>														
Environmental Req.	1	1	1	1	2	1	1	1	1	N/A	N/A	1	2	2

Legend: 2 strong 1 partial 0 weak/risk N/A not assessed

We evaluated every tool in the top 30% by centrality against the 19 aspects, with Table 8 serving as our overview. The full per-tool aspect ratings, confidence levels, and evidence behind each are given in the Appendix B.

During the cross-case comparison, we observed that several recurring patterns in the profiles aligned with the projects' current stewardship models. We therefore organized the tools into four descriptive groups (A–D): foundation-hosted, vendor-controlled, independent, and research-led software.

The foundation-hosted tools (A) have the most uniformly strong social and technical profiles. Their differences concern the degree of financial control, the concentration and continuity of employer-backed work, and the maturity of their contributor bases.

The vendor-controlled tools (B) combine considerable technical and economic support with dependence on corporate governance and product strategy. Their profiles vary despite this shared structure, particularly in contributor renewal, licensing, financial runway, and the boundary between the open project and the commercial product.

The independent projects (C) show that the absence of an institutional controller does not imply project health or financial autonomy. The research-led tools (D) combine no dedicated project funding and small communities with a clear technical and environmental purpose.

Across all four groups, Personal Motivations remains unassessed because the available sources are not sufficiently conclusive. Environmental Requirements is strongest only among projects that explicitly make environmental measurement part of their purpose. The following subsections examine the individual profiles and the evidence behind these differences.

4.4.2 Foundation-hosted tools

The five tools in group A are hosted by a neutral foundation and currently use stable OSI/FSF-approved licences, and they share the strongest and most uniform profiles in the census. We treat Prometheus in full, both because it is the most central tool in the ecosystem and because its near-uniform profile provides a reference for the remaining cases.

Prometheus

Prometheus is a metrics-based monitoring system built on a pull model: it scrapes numerical time series from instrumented targets, stores them in a local time-series database (TSDB), and exposes them through the PromQL query language for dashboards and alerting. It originated at SoundCloud in 2012 and was accepted by the Cloud Native Computing Foundation (CNCF) in 2016 as its second hosted project after Kubernetes, graduating in 2018 ([Cloud Native Computing Foundation, 2018](#)). Its exposition format has since become the de facto standard for metrics, likely the reason why it is the most central tool in the ecosystem graph, with a centrality of 9.68, far ahead of any other tool (Table 5). It is also highly interoperable: beyond its own format it ingests Graphite, StatsD, and Collectd through exporters, accepts OpenTelemetry (OTLP) data, and both reads and writes through a documented remote protocol that several long-term-storage systems implement.

Its profile is strong in 17 of the 19 aspects. Socially, development is fully in the open: more than 1400 contributors over the project's history and a governance body of around 50 members from different organizations ([Prometheus, 2026d](#)), communicating across public mailing lists, the CNCF Slack, a Discourse forum, and the PromCon conference ([Prometheus, 2026a](#)). Governance is explicit, team-member status follows from sustained contribution and a supermajority vote. Maintainers own specific parts of the tree, and the project adopts the CNCF Code of Conduct ([Prometheus, 2026c](#)). Technically it is under continuous, well-organized development: a six-week release cycle with shepherds, semantic versioning, and long-term-support. Quality

practices are visible, including around 270 test files, CI, fuzzing, CodeQL, govulncheck, and an OpenSSF Scorecard of 8.1/10 ([Open Source Security Foundation, 2026e](#)). Licensing is Apache-2.0 from the first licence-bearing public commit in 2012 through the current repository history, with no substantive relicensing found; contributions use DCO sign-off and the Prometheus mark is held by the Linux Foundation rather than a single vendor ([Prometheus, 2012](#); [Prometheus, 2026d](#); [Prometheus, 2026b](#); [The Linux Foundation, 2026](#)). Economically, a substantial ecosystem generates value around Prometheus, while the CNCF provides project infrastructure. Maintainers affiliated with several employers contribute to it: the self-funded-contributor pattern in which corporate peers finance shared infrastructure they build products on, rather than selling the software itself as described by [Monaco \(2023\)](#). This produces strong External Value Capture and Reinvestment. Financial Resources is also strong: diversified CNCF backing is durable, while Prometheus's strategic importance and mature governance give it meaningful influence over resource requests and priorities ([Cloud Native Computing Foundation, 2025a](#); [Cloud Native Computing Foundation, 2026b](#); [Cloud Native Computing Foundation, 2018](#); [Prometheus, 2026c](#)).

The two aspects without a strong rating are instructive. Same as all other projects analyzed, we report Personal Motivations as N/A: the project offers recognition through its team-member progression ladder, mentorship, and conference visibility, but whether these satisfy contributors' underlying motivations cannot be responsibly inferred from public data. Environmental Requirements is rated partial: resource efficiency is a first-class engineering concern (the TSDB is heavily optimized) but efficiency is designed for cost and scale, and is never framed as a sustainability goal. The motivation and environmental patterns recur across the census, while the economic separation distinguishes project autonomy from external support.

Prometheus is close to a best case for OSS sustainability, and that is what makes it the reference point: on the healthiest projects the evaluation largely restates signals that are already trivial to read (foundation graduation, release activity, star counts), so its discriminating value lies in the harder cases discussed below. Two mild risks are still worth recording. The first is maintainer concentration: a substantial share of maintainer activity comes from a handful of companies, Grafana Labs most of all, so a strategic shift at one employer could remove a meaningful share of it. The second is the ecosystem's dependence on Prometheus itself: because its exposition format is the de facto standard, the health of much of the observability ecosystem is coupled to this one project, not a formalized neutral standard, which creates an ecosystem-level risk despite the project's strong profile.

OpenTelemetry

OpenTelemetry represents convergence through standardization, being a standard for traces, metrics, logs, and profiles. Originating in 2019 as a merger of OpenTracing and OpenCensus ([MCLEAN; SIGELMAN, 2019](#)), it provides vendor-neutral APIs, SDKs, a wire protocol (OTLP), and a Collector reference implementation. It intentionally excludes storage, querying, and visualization, instead standardizing how telemetry is generated, collected, and exported. At

the instrumentation layer, OpenTelemetry and Prometheus provide overlapping client-library approaches to recording metrics, and official compatibility guidance describes their model differences and equivalent patterns ([OpenTelemetry, 2026c](#)). They interoperate in both directions: Prometheus can ingest OTLP data and OTel can ingest Prometheus metrics. However, this interoperability is imperfect and potentially lossy because the data models differ ([Prometheus, 2026e](#)).

This convergence is reflected in OpenTelemetry's sustainability profile. Its graduation from the CNCF in 2026, elected governance, public SIG structure, and leadership distributed across competing vendors support its social legitimacy while reducing dependence on any single company ([OpenTelemetry, 2026b](#); [Cloud Native Computing Foundation, 2026a](#)). Its canonical specification is Apache-2.0 since inception; a CLA is in place, but is assigned to the CNCF/Linux Foundation rather than to a single vendor ([OpenTelemetry, 2019](#); [OpenTelemetry, 2026a](#)). The same convergence produces strong External Value Capture and Reinvestment: employer-backed maintainers from several companies contribute to the project because their products depend on OTLP ingestion, distributions, and managed services, without one vendor controlling it. OpenTelemetry also has strong Financial Resources: the CNCF provides diversified and durable backing, while the project's Governance Committee controls requests and allocation priorities across the project ([Cloud Native Computing Foundation, 2025a](#); [Cloud Native Computing Foundation, 2026b](#); [OpenTelemetry, 2025a](#)). Its main qualification comes from the breadth that makes the standard useful. Maintaining clients across around twelve languages, together with the specification, OTLP, and the Collector, creates a substantial coordination and consistency cost. The project's status states that tracing, logging, and their protocols are stable ([OpenTelemetry, 2025b](#)). Similarly to Prometheus, Environmental Requirements is partial because performance and telemetry-volume reduction are engineering concerns but are not framed as environmental goals; Personal Motivations is not reliably assessed.

Kafka

Kafka has a profile similar to Prometheus and OTel. Its Apache Software Foundation governance, Apache-2.0 licence, more than 1,700 lifetime contributors, public developer mailing list, and Kafka Improvement Proposal process support strong scores across the Social and Technical dimensions ([Apache Software Foundation, 2026b](#); [Apache Software Foundation, 2026d](#)). The same process also preserves design decisions for a large codebase, while the removal of ZooKeeper in Kafka 4.0 shows that the project can complete architecturally significant refactors ([Apache Software Foundation, 2025](#)). Its economic support is broad, including managed service offerings and maintainers employed by several companies. This gives Kafka strong External Value Capture, while the concentration of employer-backed work around Confluent makes Reinvestment partial and ASF support gives the project partial discretionary Financial Resources because evidence of meaningful control over foundation resources is limited. This work concentration does not give Confluent control over the ASF project, but it moves part

of the licensing risk to the surrounding stack: Schema Registry, ksqlDB, REST Proxy, and some connectors use the source-available Confluent Community License, while Kafka, Kafka Connect, and Kafka Streams have remained Apache-2.0 from the initial 2011 ASF import through the current tree ([Apache Software Foundation, 2011](#); [Apache Software Foundation, 2026b](#); [CONFLUENT, 2018](#)). Thus, Kafka's strong Licensing score describes the core project, not every component that users may associate with a Kafka deployment.

Jaeger

Jaeger is also active and CNCF-graduated, but it follows a different path from the previous three. The project changed from MIT to Apache-2.0, which are the same class (permissive OSS), through a public proposal in 2017 and has remained Apache-2.0 since ([Jaeger, 2017](#); [Jaeger, 2026](#)). It has six active maintainers from six employers, approximately 490 lifetime contributors, and explicit governance and release practices ([Jaeger, 2026](#); [Cloud Native Computing Foundation, 2019](#)). However, this maintainer core is small enough that we score Personal Knowledge as partial, and the commercial ecosystem around Jaeger itself is more limited than those around Prometheus or Kafka. Jaeger therefore has partial External Value Capture, strong Reinvestment from the CNCF and six employers, and partial Financial Resources because CNCF support is available but evidence of meaningful project-level allocation control is limited. The project responded to the convergence around OpenTelemetry by deprecating its own instrumentation libraries and rebuilding Jaeger v2 on the OpenTelemetry Collector ([Jaeger, 2024](#)). This decision kept Jaeger interoperable and reduced duplicated instrumentation work, but also narrowed its distinct scope to tracing storage, query, and visualization. Its centrality of 1.27 is consistent with this change: the instrumentation interface moved to OpenTelemetry even though Jaeger remains an active and technically strong project.

Kepler

Kepler is an outlier among the foundation-hosted tools. It has formal CNCF governance, public enhancement proposals, frequent pre-1.0 releases, and an Apache-2.0 licence present in both its earliest public release and current tree, but these mechanisms sit over a young and concentrated community ([Kepler, 2022](#); [Kepler, 2026](#)). In 2025, twelve maintainers moved to emeritus status, including the original creator and the complete Intel contingent, while three Red Hat engineers produced approximately 84% of the rewritten code. A controlled evaluation of Kepler 0.7.2 found less than 1% error in total energy over the experiment, but unreliable container-level attribution, including power assigned to containers that were no longer running ([PIJNACKER *et al.*, 2025](#)). Kepler's maintainers subsequently replaced the eBPF-based implementation with a design using read-only host interfaces and validation against IPMI measurements ([MANOLEDAKI; CHOOCHOTKAEW, 2026](#)). We therefore score Functionality, Contributor Retention, Contributor Attraction, Personal Knowledge, External Value Capture, Reinvestment, and Financial Resources as partial. CNCF hosting provides real resources, but day-to-day governance remains with Kepler's maintainers and evidence of meaningful project-

level allocation control is limited. At the same time, Kepler scores strongly on Environmental Requirements because measuring workload energy use is its explicit purpose. This score records environmental orientation; measurement accuracy remains under Functionality rather than Legitimacy.

4.4.3 Vendor-controlled tools

The five tools in group B share control by one company over project strategy or the commercial boundary around the project. Elasticsearch and Grafana provide the closest comparison: both vendors relicensed their projects, but Elastic removed an open-source option in response to AWS competition and the project split into Elasticsearch and OpenSearch, while Grafana adopted AGPLv3, kept its plugin-facing libraries permissive, and continued its AWS partnership (BANON, 2021; The Linux Foundation, 2024; Grafana Labs, 2021; Grafana Labs, 2020; Grafana Labs, 2026a). Grafana consequently retains an OSI/FSF-approved open-source core after relicensing, while Elasticsearch retains a structural licensing risk despite later adding AGPLv3 as a response to the backlash. InfluxDB, Kibana, and Nagios show other ways in which vendor control shapes the boundary and trajectory of an OSS project.

Elasticsearch

Elasticsearch is the clearest case because Elastic has exercised its control over the licence. In 2021, the company relicensed the Apache-2.0 portions of Elasticsearch and Kibana to the unfree SSPL and Elastic License 2.0 in response to AWS's managed service (BANON, 2021; Open Source Initiative, 2021). AWS then forked OpenSearch from the last Apache-2.0 release, and the project moved to the Linux Foundation in 2024 (BANON, 2021; The Linux Foundation, 2024). Elastic added AGPLv3 as an open-source option later that year, but the x-pack tree remains under ELv2, while the project's CLA and all-Elastic code ownership concentrate licensing control and preserve a structural risk of another unilateral relicensing (BANON, 2024; ELASTIC, 2026a; RIEHLE, 2012).

We therefore score Governance as partial and Licensing as weak. At the same time, Elastic directly funds a large engineering organization and its flagship product: Elastic's reports fiscal-year revenue of \$1.739 billion, 94% of it from subscriptions. Development and project operations are directly visible in the repositories. This supports strong Financial Resources, External Value Capture, and Reinvestment, although the concentration keeps confidence in the first and third ratings at Medium (Elastic N.V., 2026; ELASTIC, 2026b). The support and governance risk have the same source: direct commercial control both sustains development and gives one vendor the ability (and incentive) to change its terms.

Grafana

Grafana also made licensing changes with a similar objective to Elastic's, but with a different method and outcome. Grafana Labs moved Grafana from Apache-2.0 to AGPLv3

in 2021, with the goal of restricting providers integrating the software into proprietary SaaS offerings without contributing back or making commercial agreements. The plugin-facing libraries remained Apache-2.0, and AWS continued as a managed-service partner ([Grafana Labs, 2021](#); [Grafana Labs, 2020](#); [Grafana Labs, 2026a](#)). Grafana documents public voting and contributor roles, although Grafana Labs retains authority over governance changes and supplies most team members. Grafana Labs reports more than \$400 million in annual revenue and 7,000 customers, and its repository makes the company's development support directly visible ([Grafana Labs, 2025a](#); [Grafana Labs, 2026b](#)). The evidence supports strong Financial Resources and Reinvestment, with Medium confidence because both remain company-dependent.

Elasticsearch and Grafana consequently receive similar profile scores, but the history shows different degrees of risk: while both depend on one vendor and currently employ CLAs combined with dual-licensing, only Elasticsearch exercised its authority by removing the open-source option and splitting the project.

InfluxDB

InfluxDB is a typical open-core project. Its earliest visible history and v1/v2 line use MIT, while v3 added Apache-2.0 as an additional license, likely to enforce the patent grant on contributions; Licensing nevertheless scores partial ([INFLUXDATA, 2013](#); [INFLUXDATA, 2026b](#)). The current terms remain open source, but InfluxData's individual CLA grants the company a perpetual right to sublicense contributions and explicitly presents future relicensing as an option ([INFLUXDATA, 2018](#)). InfluxData restricts capabilities through the open-core model instead: InfluxDB 3 Core is single-node and limits queries to approximately 72 hours, while the proprietary Enterprise edition supplies high availability, multiple nodes, and unrestricted historical queries ([INFLUXDATA, 2026b](#); [INFLUXDATA, 2026c](#); [INFLUXDATA, 2025](#)). The open-edition licensing has therefore remained OSI/FSF-approved even though production use is steered toward the paid product. Successive migrations from v1 to v2 to v3, from Go to Rust, and from Flux back to SQL and InfluxQL add a separate continuity cost for users ([INFLUXDATA, 2026a](#)).

Kibana

Kibana reproduces Elasticsearch's rating profile aspect for aspect. It shares Elastic's governance, CLA, licensing history, OpenSearch fork, high recurring revenue, and reinvestment, despite having much lower centrality (1.24 against 5.48) ([BANON, 2021](#); [BANON, 2024](#); [ELASTIC, 2026a](#); [The Linux Foundation, 2024](#); [Elastic N.V., 2026](#)). This indicates that the methodology is capturing the vendor's governance structure rather than only the technical properties or ecosystem position of one product.

Nagios

Nagios presents a different outcome within vendor control. Its governance problems and control produced observable exits. Nagios Enterprises controls the project, its trademark, and

the boundary between Nagios Core and the commercial Nagios XI product (Nagios Enterprises, 2026a). The core itself has used GPLv2 from the initial 2002 public import through the present date, and no CLA requirement was found in its contribution guide or pull-requests. We therefore score Licensing as strong; Nagios Enterprises' trademarks and proprietary Nagios XI product do not alter the rights granted over the evaluated Core tree (Nagios Core Development Team, 2002; Nagios Enterprises, 2026b; Nagios Core Development Team, 2026; Nagios Enterprises, 2026a). The project accepts fixes, but has no public roadmap or external decision body, so major feature decisions remain restricted to the company (Nagios Enterprises, 2026b). In Icinga's retrospective account, Nagios community members and advisory-board participants created the fork in 2009 to build an easier and more scalable alternative (Icinga, 2023). The Monitoring Plugins project states that the plugin community separated again in 2014 after a domain and trademark dispute (Monitoring Plugins Development Team, 2026). These events support the only weak Governance score among commercially funded tools in the census. Nagios still receives security and maintenance releases, and Nagios Enterprises maintains a commercial product and company-led development around the core, but project-level funding and maintenance expenditure are not publicly disclosed; Financial Resources, External Value Capture, and Reinvestment therefore remain partial. This support has sustained the project without repairing the governance conditions that drove contributors elsewhere.

4.4.4 Independent projects

Group C contains Net-SNMP and Ganglia, two older projects with no visible current institutional controller. This organizational independence does not indicate health: Net-SNMP remains active without visible funding, while Ganglia has largely lost both maintainers and activity. Ganglia began in the UC Berkeley Millennium Project, but later evolved into broadly deployed operational infrastructure and is no longer organized around an active research program.

Net-SNMP

Net-SNMP is the active but unsupported case in this group. It is the reference implementation of the SNMP standards, remains widely distributed, and has maintained approximately 300 commits per year from 2021 through 2026 (Net-SNMP Development Team, 2026b). Its BSD-style licensing is from the earliest available 1996 history through the current tree; later changes accumulated additional notices with the same terms rather than restricting the project (UCD-SNMP Development Team, 1996; Net-SNMP Development Team, 2026b). This supports strong Legitimacy, External Communication, Sustained Activity, Functionality, and Licensing scores. However, the large C codebase depends heavily on one or two maintainers, has no formal governance or contribution guide, and has no visible company, foundation, sponsorship, or paid stewardship. We therefore score Personal Knowledge, Financial Resources, and Reinvestment as weak. In contrast, External Value Capture is strong because the project has long been embedded in commercial and community distributions, including Red Hat Enterprise Linux, without visible

resources returning to the project ([Net-SNMP Development Team, 2026a](#); [Red Hat, 2026](#)). Net-SNMP is not dormant, but its continuing infrastructure role depends on a maintainer base and funding model more vulnerable than its downstream adoption would suggest.

Ganglia

Ganglia shows a different outcome of thin stewardship: activity has already faded. Its low-overhead architecture remains suitable for the HPC-cluster role for which it was designed, and the project has genuine historical and academic legitimacy ([MASSIE *et al.*, 2004](#)). Its licensing has remained stable: an initial University of California notice was clarified to BSD-3-Clause in 2010, with no later substantive licence change found ([Ganglia Development Team, 2010](#); [Ganglia Development Team, 2026b](#)). The core repository has had no commits since 2021, the web component receives only sporadic fixes, and no release has been cut for around a decade ([Ganglia Development Team, 2026b](#); [Ganglia Development Team, 2026a](#)). There is no current governance structure, maintainer list, sponsor, or visible commercial ecosystem. Contributor Retention, Contributor Attraction, Governance, Sustained Activity, Quality Assurance, Methodologies/Practices, and all Economic aspects consequently score weak. Ganglia differs from Net-SNMP because the human and financial capacity to continue maintaining it has largely disappeared.

4.4.5 Research-led software

PowerJoular and JoularJX form the research-led software group. Both are led by the same researcher, have used GPLv3 since their first public releases in 2021 without a substantive licence-text change in the examined histories, and have explicit environmental goals ([NOUREDDINE, 2021b](#); [NOUREDDINE, 2021a](#); [NOUREDDINE, 2026d](#); [NOUREDDINE, 2026c](#); [NOUREDDINE, 2022](#)). The project reports no dedicated funding, though limited equipment and activities received support in 2020-2021 ([NOUREDDINE, 2026b](#)). The profiles are therefore similar: weak Governance, Financial Resources, and External Value Capture; partial Reinvestment and several Social aspects; and strong Functionality, Architecture/Maintainability, Licensing, and Environmental Requirements. Their scopes establish relevant research roles, although the available evidence does not independently validate measurement effectiveness. The pair's centrality of 1.00 each should not be read as broad ecosystem reach because it results from the two tools referencing one another in a small connected component, an artifact of the centrality metric.

PowerJoular

PowerJoular is written in Ada and attributes approximately 77% of its commits to one author ([NOUREDDINE, 2026d](#)). It has been maintained for five years and documents support for CPUs, GPUs, virtual machines, and Raspberry Pi models, but activity fell from 103 commits in 2024 to 23 in 2025 and one by the middle of 2026. Its documented scope supports strong Functionality, with Medium confidence because measurement effectiveness has not been independently validated. The repository has release and build automation but no

automated tests, contribution guide, nor succession structure. Personal Knowledge, Governance, Openness/Onboarding, Financial Resources, and Quality Assurance thus score as weak, while Contributor Retention and Contributor Attraction score partial. Its GPLv3 licence since the first public beta and explicit purpose of measuring software energy consumption support strong Licensing and Environmental Requirements scores.

JoularJX

JoularJX has the same academic origin and environmental goal as PowerJoular, but has different implementation details. It is written in Java, includes a JUnit test suite, and has a repeated external-contributor tail, with its main author responsible for around 60% of commits (NOUREDDINE, 2026c). These differences support strong Sustained Activity and partial Personal Knowledge for JoularJX rather than PowerJoular's corresponding partial and weak scores. The otherwise close match provides an internal consistency check: the analysis gives sibling projects similar profiles while distinguishing the language, tests, and contributor distribution that differ.

4.4.6 Synthesis

RQ1: What strengths, risks, and evidence gaps appear in selected projects' sustainability profiles?

Our profiles show strengths, risks, and evidence gaps across different dimensions rather than one degree of project health. The most frequent strong ratings are technical: 11 of the 14 tools receive strong Sustained Activity, Functionality, and Architecture/Maintainability ratings, and 10 receive strong Licensing ratings. The most frequent weaknesses are distributed across Governance, Personal Knowledge, Quality Assurance, Licensing, and the three Economic aspects. Governance and Financial Resources each have four weak ratings, External Value Capture has three, and Licensing and Reinvestment have two each. Personal Motivations is N/A for every tool because the public evidence does not reveal whether contributors' motivations are actually supported. Environmental Requirements is strong only for Kepler, PowerJoular, and JoularJX, whose stated purpose includes measuring energy use; nine tools receive partial ratings based on documented resource-efficiency concerns, while Nagios and Net-SNMP are N/A because no public evidence was available.

RQ2: Which recurring sustainability patterns appear across the selected projects?

The clearest recurring patterns observed in our sample correspond to current stewardship. Foundation-hosted tools have the most uniform Social and Technical strengths, even though Kepler has a younger and more concentrated contributor base. Vendor-controlled tools combine strong technical and economic support with partial or weak Governance and, in several cases, partial or weak Licensing; Nagios shows that company support does not necessarily lead to strong governance or technical practices. The two independent projects differ: Net-SNMP

remains active despite weak Financial Resources, Reinvestment, and Personal Knowledge, while Ganglia is weak in Sustained Activity, Contributor Attraction and Retention, Governance, Quality Assurance, Practices, and all three Economic aspects. The research-led projects, PowerJoular and JoularJX, share weak Governance, Financial Resources, and External Value Capture with partial Reinvestment, while differing in Sustained Activity, Quality Assurance, and Personal Knowledge.

RQ3: What does comparing sustainability profiles with ecosystem centrality reveal?

Tools with different centrality can have similar profiles, and tools with comparable ecosystem positions can have different project-level profiles. Prometheus combines the highest centrality (9.68) with 17 strong aspect ratings. Elasticsearch and Kibana share their rating profiles despite centrality scores of 5.48 and 1.24, because both inherit the same vendor structure. Net-SNMP remains relatively central (2.14) and active while receiving weak Personal Knowledge, Financial Resources, and Reinvestment ratings. Jaeger is technically and socially strong despite a lower centrality of 1.27 after instrumentation work moved to OpenTelemetry. PowerJoular and JoularJX each receive 1.00 from references within their small two-tool component rather than broad interface reach, an artifact of the centrality metric. These comparisons show that centrality identifies incoming reference exposure in the observed graph, while the profiles describe the condition of the projects behind that exposure.

4.5 Discussion

We combined the 19 sustainability aspects identified in our systematic mapping study with the ecosystem position measured in our observability study (FONTES *et al.*, 2026b; FONTES *et al.*, 2026a). The aspect profiles distinguish projects that occupy similarly central ecosystem positions but have different governance and funding. In the same way, ecosystem position indicates which of the per-project profiles may matter beyond the project itself, in the form of interfaces and compatibility. The resulting analysis is exploratory and does not try to predict project failure, but makes different forms of sustainability risk comparable and visible.

4.5.1 What we learned about the tools

The four groups show that sustained activity is compatible with different underlying conditions. The foundation-hosted projects have the most uniform social and technical profiles. Their strongest cases combine neutral institutional homes, public decision processes, stable OSI/FSF-approved licences, and maintainers from several different employers. Their Economic profiles also distinguish access and allocation control. Prometheus and OpenTelemetry share the CNCF's diversified backing and have enough strategic value and governance maturity to exercise influence over resource requests and priorities, so we rate their Financial Resources as strong; the other foundation projects may have real resources but less evidence of comparable project-

level control. External actors also generate value and reinvest it through paid development and sponsorships. Prometheus and OpenTelemetry are the strongest examples, while Kafka shows that commercial concentration can coexist with neutral governance. Jaeger retains a broadly strong profile after ceding scope to OpenTelemetry, and Kepler shows that formal governance can be established before a wide contributor base. Foundation affiliation is not sufficient to guarantee sustainability, but our results show that it provides a clear defence against unilateral control and organizes investments.

The vendor-controlled group exposes different strengths and risks. Elasticsearch, Grafana, and Kibana receive strong Financial Resources and Reinvestment ratings due to broad recurring company revenue durably supporting these flagship projects. These ratings have Medium confidence because the support remains vendor-dependent. InfluxDB instead receives partial Financial Resources and strong Reinvestment, both with High confidence: InfluxData's recurring support across development and project operations is directly visible, while control remains concentrated and evidence of durable financial runway is limited. External Value Capture is strong where a wider market exists, while InfluxDB and Nagios remain partial because economic activity is dominated by their controlling companies. The same commercial control produces partial or weak Governance and partial Openness scores because product strategy and project strategy are decided together. This agrees with the distinction between business models that build a shared market around a project and models that reserve capabilities to one vendor (MONACO, 2023; POPP, 2023). Contributor licence agreements and concentrated distribution rights are particularly relevant because they provide the legal capacity for unilateral relicensing (RIEHLE, 2012). In Elasticsearch, that capacity was exercised and followed by the OpenSearch fork, a pattern also observed in an empirical study on restrictive relicensing (FOSTER, 2024). InfluxDB shows the limit of looking only at licence: its open-edition options remain OSI/FSF-approved (although with a CLA), but the production boundary is enforced through proprietary features. Nagios is the within-group counterexample: Nagios Enterprises provides maintenance, although project-level funding is not publicly disclosed, and this has not produced open governance, contributor renewal, or comparable technical evolution. Company support can therefore sustain activity without producing a uniformly strong sustainability profile.

The independent group contrasts two projects without a visible current controlling vendor, foundation, or research programme. Net-SNMP remains active and is widely depended upon, but has no visible funding and relies on very few maintainers. Ganglia has a stable licence and a sound historical architecture, while the capacity to maintain it has largely disappeared. Both resemble digital infrastructure whose use is dispersed while responsibility for upkeep remains concentrated (BERLINGUER, 2023), but their current sustainability profiles differ substantially. PowerJoular and JoularJX remain research-led software: one principal researcher, no dedicated project funding, and no visible commercial market define their current academic setting (NOUREDDINE, 2026b). Their profiles reveal succession and knowledge risks that matter if other work comes to depend on them.

4.5.2 What we learned about the 19 aspects framework

The profiles discriminate more clearly by project shape than by a single degree of health. Prometheus and OpenTelemetry are strong on nearly every social and technical aspect, while the refined economic aspects also capture how their strategic importance and mature governance convert foundation resources into control over financial resources. The framework's value becomes clearer for projects whose strengths and risks coexist. Elasticsearch is technically active with strong capacity and reinvestment while remaining dependent on one vendor for both; Net-SNMP is technically active and legitimately embedded while commercial actors capture value without returning visible support. Simply producing a scalar number by combining the scores would partly cancel these opposing facts. Reading the profile preserves them and the evidence explains why the same numeric score can represent different risks.

The application also shows that the aspects are strongly related rather than independent. Governance can affect Licensing directly, which in turn influences Contributor Attraction, Openness, or External Communication. The economic aspects also form distinct paths: External Value Capture may convert to project-controlled Financial Resources or as direct Reinvestment, while grants and donations may enter either path without commercial activity. Net-SNMP shows strong External Value Capture with little return; the foundation-hosted projects show strong or partial Reinvestment alongside different degrees of control over their parent foundation's resources; and the flagship vendor projects show that strong capacity and reinvestment can coexist with concentration in one actor. Licensing distinguishes three conditions within this group. Elasticsearch and Kibana are weak because Elastic exercised its authority to remove their open-source option, while Grafana and InfluxDB are partial because vendor-held CLAs preserve unilateral authority but their current cores have never been converted to a proprietary license (BANON, 2021; BANON, 2024; Grafana Labs, 2021; INFLUXDATA, 2018). Nagios is strong because Core's GPLv2 history is stable and no CLA requirement was found in its public contribution process, even though the company controls the trademarks and separate Nagios XI product (Nagios Core Development Team, 2002; Nagios Core Development Team, 2026; Nagios Enterprises, 2026a). The profile must still be read with its evidence: Grafana exercised control from permissive to a copyleft open-source licence (technically possible even without a CLA), while InfluxDB has not used its authority.

Environmental Requirements is partial for general-purpose tools with documented resource-efficiency work framed around cost and performance, and strong for Kepler, PowerJoular, and JoularJX because environmental measurement is their stated purpose. It remains N/A for Nagios and Net-SNMP, where only inferred footprint benefits were available. Personal Knowledge is strong or partial for projects with several maintainers and falls to weak for the low bus-factor cases. These boundaries delimit what each aspect measures and where the framework needs evidence from more than one aspect.

4.5.3 What we learned about the ecosystem centrality metric

The comparisons reported in Results delimit what the centrality metric can support. Incoming reference weight can represent either interface reach, an implementation of a protocol, or strong coupling. It does not encode governance, contributor renewal, maintainability, funding, or project health. Centrality therefore identifies ecosystem exposure within the graph, while the profiles characterize the condition of the projects behind that exposure. Reading both prevents a prominent node from being interpreted automatically as broadly adopted or as evidence that it is sustainable.

Our analysis also shows limitations of the centrality graph. The tool set, which is derived from literature, lags behind more recent ecosystem changes. References to interfaces also remain in literature long after the projects initially implementing them have declined. Small interconnected components can produce incoming weight without broad ecosystem reach, which we consider an unintended artifact of the centrality methodology. These limitations do not make the metric unusable, but require each centrality score to be read as a property of the observed graph rather than as a general measure of real-world importance.

Future applications could compare the current normalized weighted in-degree with PageRank or other global centrality measures that also consider the relevance of the referring nodes. Component-aware normalization or a requirement for support from outside a tool's immediate component could also help distinguish ecosystem exposure from mutual references inside a small group.

4.5.4 Implications

For ecosystem analysis, our results support a two-stage reading. Firstly, ecosystem relations can identify projects or interfaces whose condition may have effects outside their own repositories. Secondly, multidimensional profiles distinguish the source of risk, such as governance capture, unfunded maintenance, concentrated knowledge, restricted product boundaries, or fading activity. This is more informative than applying a simple activity threshold to every project because the appropriate response differs by shape. A governance fork, funding intervention, succession plan, or migration strategy addresses a different problem even when several projects receive the same partial score.

For application of the 19-aspect framework from our systematic mapping study ([FONTES *et al.*, 2026b](#)), the analysis suggests that future usage should retain the per-aspect evidence and confidence fields, distinguish insufficient evidence from evidence of weakness and avoid aggregating sustainability scores. For the ecosystem centrality method from our observability study ([FONTES *et al.*, 2026a](#)), centrality should be accompanied by checks for isolated components, historical lag, and the difference between an implementation and its interface. The two instruments applied together provide an exploratory way to identify project-level and ecosystem

risks that neither can capture alone. Evaluating whether this information can help improve dependency selection or support maintenance decisions requires a separate study with practitioners.

4.5.5 Threats to validity

Construct validity is affected by the breadth of sustainability and by the interpretation required to map public evidence to ratings. Several aspects overlap in their causes, and the encoding into weak, partial, and strong is by definition lossy. We mitigated these threats by defining the refined boundaries explicitly, applying the same criteria to every tool, recording confidence separately, retaining an evidence summary for every rating, and using N/A when the public evidence does not support a rating. We did not aggregate the ratings because doing so would imply independence and equal distance between the values. Personal Motivations remains unassessed: public artefacts can show recognition mechanisms, but not whether they satisfy contributors' motivations.

The centrality measure introduces a second construct threat. It is based on tool-name occurrences in source code, so it can capture interfaces, implementations, or mutually connected small components differently. The top-30% threshold is useful for defining a census within the graph, but it does not reflect real-world importance or adoption. We therefore report centrality alongside study and adoption evidence and use it to define ecosystem position rather than project health. The PowerJoular/JoularJX component and OpenTelemetry's recent growth make these known limitations visible.

Bias is possible because one researcher collected the qualitative evidence and assigned the scores. We did not perform independent coding or calculate inter-rater agreement, and prior familiarity with several tools can influence both evidence selection and interpretation. The explicit criteria, evidence tables, and confidence markings make the decisions more auditable. The close profiles obtained for Elasticsearch/Kibana and PowerJoular/JoularJX are useful internal consistency checks because each pair shares structural conditions and differs where the evidence actually differs. The four groups are post-hoc descriptive groups; the small sample does not establish a taxonomy or imply that the same groups will recur in other OSS ecosystems.

Finally, the evidence is a snapshot collected in June–July 2026, while sustainability is longitudinal. We used project histories, forks, release cadence, and governance changes where available, but recent activity cannot predict the future and past decline does not prove abandonment. The analysis is descriptive and contains no statistical inference. Our conclusions should therefore be read as an evidence-based comparison of current project shapes and ecosystem exposure, not as predictions of failure or to be used directly as recommendations for adoption.

4.6 Conclusion

We combined multidimensional sustainability profiles with ecosystem centrality to examine 14 OSS cloud observability projects. We found that sustained technical activity does not imply a uniformly strong sustainability profile. Active projects may still have concentrated knowledge, limited funding, concentrated governance, or company-held relicensing authority. Public evidence also has limits, most visibly for contributors' Personal Motivations.

The recurring patterns in our sample were related to current stewardship. Foundation-hosted projects had the most consistent social and technical strengths. Vendor-controlled projects generally combined substantial support with different governance, licensing, and product-boundary risks. Independent and research-led projects showed that the absence of company control does not imply financial autonomy, contributor renewal, or succession capacity.

Centrality provided a different view. It represented incoming reference exposure within the observed graph, which may reflect references to implementations or interfaces, but did not measure project health, adoption, or future continuity.

We therefore treat centrality as an indication of where ecosystem exposure lies and sustainability profiles as descriptions of the conditions behind it. Our findings are descriptive rather than predictive and remain bounded to the selected cases, observed graph, and public evidence collected in June–July 2026.

Conclusions

The three studies in this dissertation answer complementary questions. The first characterizes the project conditions associated with OSS sustainability, the second maps relations and prominent positions among tools in a concrete ecosystem, and the third examines what becomes visible when both views are considered together. We answer each dissertation research question below while emphasizing how the later studies refined the initial result.

5.1 Answers to the Research Questions

RQ1: Understanding and Characterizing OSS Sustainability

OSS sustainability can be characterized through 19 aspects across the social, technical, individual, economic, and environmental dimensions, as described in Chapter 2. Together, these aspects cover the conditions under which a project and its community can continue delivering value, including contributor participation, governance, technical evolution, knowledge, funding, value capture, reinvestment, licensing, and environmental concerns.

Applying the aspects in Chapter 4 refined this answer. The aspects do not form an independent checklist in which strengths can offset weaknesses, but appear in related configurations. Sustainability is therefore better represented as a multidimensional profile, accompanied by its evidence and confidence, than as one aggregate score.

RQ2: OSS Cloud Observability Ecosystem

The ecosystem study in Chapter 3 identified 45 OSS tools used for instrumentation, collection, processing, storage, visualization, and alerting in cloud computing solutions. Source-code occurrences among these tools produced a weighted relation graph with uneven centrality, in which we can see both recurring combinations of tools and interfaces that are referenced across different observability stacks.

The sustainability assessment refined how this graph should be interpreted. Centrality represents incoming reference exposure within the observed graph, which may arise from an implementation, a protocol, a data format, or strong coupling between components. It does not by itself measure adoption, project health, or future continuity. The graph therefore describes ecosystem position within the study boundaries, while the project profiles provide the evidence needed to interpret the conditions behind that position.

RQ3: Sustainability Profiles and Ecosystem Position

The 14 projects in the top-centrality tier exhibit different combinations of sustainability conditions. Similar profiles may occur at different centrality levels when projects share governance and economic structures, while projects with comparable ecosystem positions may differ in funding, knowledge concentration, licensing, or capacity for succession. The four stewardship groups provide a descriptive way to organize these recurring patterns.

Comparing the profiles with ecosystem position shows that project sustainability and ecosystem exposure are distinct properties. A profile characterizes the conditions under which a project can continue, while ecosystem position indicates how widely consequences associated with those conditions may extend within the observed graph. Each view changes how the other is interpreted.

5.2 Contributions and Implications

The integrated contribution of this dissertation is a structured exploratory way to relate multidimensional OSS sustainability profiles to ecosystem position. The approach combines a literature-derived framework, an ecosystem graph, and evidence-based project assessments.

Project Conditions and Ecosystem Exposure

The combined analysis distinguishes two questions that are often treated as one. The first concerns project condition: whether governance, contributor knowledge, technical practices, licensing, and economic support provide the capacity to continue. The second concerns exposure: how far the consequences of changes in that project or its interfaces may propagate. The underlying weakness does not change merely because a project is central, but its effects may reach more of the ecosystem.

This distinction explains why activity and centrality cannot serve as interchangeable sustainability indicators. For instance, Net-SNMP remains active and occupies a relevant ecosystem position while depending on a small maintainer base and receiving little visible returned support. Elasticsearch combines high centrality and strong technical and economic capacity with

concentrated governance and a history of unilateral relicensing. These are different configurations and call attention to different mechanisms.

Projects, Interfaces, and Ecosystems

Our studies also surface the difference between the continuity of a project, the continuity of an interface, and the resilience of the ecosystem that depends on it. Source-code references can point to formats, protocols, APIs, or plugin conventions rather than only to one implementation. Prometheus's exposition format, OpenTelemetry's protocols, SNMP as implemented by Net-SNMP, and the Elasticsearch and OpenSearch lineages illustrate how an interface can remain relevant across changes in project scope, governance, or implementation.

These three objects can evolve differently. A healthy project may maintain an interface used by many others, but an interface may also outlive its original implementation, move to another project, or be preserved through a fork. Ecosystem resilience then depends not only on whether one repository remains active, but also on whether compatible alternatives, neutral standards, knowledge, and governance arrangements can preserve the functions on which other projects rely. The sustainability of a project and the continuity of its interface should therefore be examined together.

Sustainability as Adaptation

The cases suggest that continuity does not always mean preserving the same repository, organization, and technical scope unchanged. In particular, OpenTracing and OpenCensus converged into OpenTelemetry; Jaeger ceded instrumentation responsibilities to OpenTelemetry and rebuilt its newer architecture around the OpenTelemetry Collector; and OpenSearch preserved an Elasticsearch lineage under different governance after the 2021 relicensing. Net-SNMP illustrates continued maintenance of a mature role, while Ganglia illustrates usable software whose capacity for renewal has substantially diminished.

Technical succession, institutional transfer, convergence, and compatible forks can preserve knowledge or functionality when maintaining the original arrangement is no longer desirable or possible. A sustainability assessment should therefore examine the capacity to adapt and transfer responsibility, not only the uninterrupted activity of one repository.

Resources, Control, and Reinvestment

The application of the economic aspects also separates conditions that are often conflated. Economic value may be generated around a project without converting into resources. A company or foundation may provide substantial resources while retaining authority over their allocation, and external actors may reinvest through paid development even when the project

has no independent revenue. Conversely, institutional independence may coexist with severe underfunding and concentrated unpaid maintenance.

This complements prior characterizations of OSS business models. [Popp \(2023\)](#) describes revenue mechanisms such as hosting, support, consulting, proprietary features, dual licensing, and open core, while [Monaco \(2023\)](#) distinguishes arrangements that build shared innovation from those that reserve capabilities to one vendor. Our profiles add a sustainability perspective by examining who controls the resulting resources and whether value is reinvested in the shared project.

Commercial success and institutional independence are therefore not sustainability outcomes by themselves. Vendor-controlled projects show that recurring revenue and reinvestment can sustain large engineering efforts while governance and licensing authority remain concentrated. Net-SNMP and Ganglia show that the absence of a controlling vendor does not guarantee financial autonomy, contributor renewal, or succession. The relevant question is how value, resources, authority, knowledge, and maintenance responsibility are distributed, because different arrangements expose different strengths and risks.

Implications for Assessment and Ecosystem Practice

For adopters, release activity and prominence can serve as initial signals, but they do not explain governance authority, succession capacity, or where economic support goes. For maintainers, the results emphasize preserving decision rationale, knowledge transfer, and succession mechanisms alongside releases and technical documentation. For foundations and funders, the profiles can help distinguish whether the relevant constraint is absent resources, concentrated authority, weak contributor renewal, or another mechanism.

For ecosystem research, the findings indicate that interfaces, standards, forks, and institutional succession deserve explicit representation rather than assuming that repository identity remains stable. For sustainability assessment, the evidence summary and confidence attached to each aspect should be retained, and insufficient evidence should remain distinct from observed weakness. These are implications from the exploratory analysis, not validated recommendations; their usefulness in concrete decisions remains to be studied with practitioners.

5.3 Limitations

The cumulative design of this dissertation means that the final assessment inherits the limitations of both preceding studies. The 19 aspects depend on the terminology, selection criteria, and published literature included in the systematic mapping. Concerns that are common in practice but use different terminology or have not been studied may be underrepresented. Applying the aspects to observability projects examines their usefulness in concrete cases, but it

is not independent validation of the framework because the studies were designed and interpreted by the same authors.

The ecosystem boundary introduces a second inherited limitation. The 45 tools were identified from academic literature and their relations were derived from tool-name occurrences in source code. Academic publications lag current practice and may omit newer tools, forks, or components that are widely used but rarely named in abstracts. A source-code occurrence may represent an implementation, an interface, documentation, a test, or an incidental reference. The centrality score can also be inflated inside a small set of interconnected components. Consequently, the selected top-centrality tier is a census of the highest positions in this observed graph, not a sample of the most adopted or important OSS projects in general.

The sustainability profiles are constrained by public evidence. Mature and institutionally visible projects publish more governance, financial, and technical material than small or informal projects, so differences in documentation may affect both ratings and confidence. Public artefacts can show recognition mechanisms but cannot establish whether contributors' personal motivations are actually fulfilled. The author of this dissertation, a single assessor, collected and interpreted the evidence, assigned the ratings, and produced the confidence levels.

Finally, the assessment is a June–July 2026 snapshot of conditions that are inherently longitudinal. Historical evidence helps explain licensing changes, forks, activity decline, and institutional transitions, but the profiles do not predict future continuity.

5.4 Future Work

The first priority is to strengthen the assessment procedure. A complete evidence log that records the source, retrieval date, affected aspect, and scoring rationale for every consequential judgment will be provided with our journal submission. Different assessors should then independently rate a subset of project–aspect pairs, with disagreements used to revise the framework. Aspect-specific thresholds for weak, partial, and strong ratings help make the boundaries more reproducible, so it is something we would like to explore.

Another top priority should be to make both the ecosystem relations and sustainability assessment longitudinal. A structured way of representing the change over time in both the ecosystem graph and when evaluating each aspect could help test whether profile changes anticipate or explain transitions in governance, activity, licensing, and shifting ecosystem position.

A structured comparison with the OSS health characteristics identified by [Linåker *et al.* \(2022\)](#) could also clarify construct overlap and gaps. Similarly, the ecosystem view should be tested against alternative constructions. Comparing the current normalized weighted in-degree with PageRank and other global measures could show how sensitive the selected tier and interpretations are to the specific centrality metric, and whether these can reduce the misleading

centrality of tightly coupled components. Explicitly representing standards, interfaces, forks, and successor projects would also help distinguish continued ecosystem functions from the continuity of one implementation.

A way to involve practitioners would also be valuable. The inability to score personal motivations without interviews is a clear limitation that can be addressed that way. Besides scoring, it should also be used to validate which evidence practitioners consider credible.

With these improvements in mind, an important direction for future work would be to reproduce our approach in other OSS ecosystems, where different technical roles, institutions, and evidence practices may require adaptation.

5.5 Final Remarks

Taken together, the three studies show why technical activity and ecosystem prominence are insufficient on their own. Sustainability depends on multiple project conditions while ecosystem position changes how the consequences of those may extend beyond one project. The resulting profiles offer a structured way to examine these conditions together.

References

ALAMI, A. The sustainability of quality in free and open source software. In: **IEEE/ACM International Conference on Software Engineering Companion (ICSE-Companion)**. 2020. p. 222–225. Available: <<https://doi.org/10.1145/3377812.3381402>>. Citation on page 30.

Apache Software Foundation. **Initial Checkin of Kafka to Apache SVN**. 2011. Repository commit. <<https://github.com/apache/kafka/commit/642da2f28c>>. Accessed: 2026-07-23. Citations on pages 76 and 126.

Apache Software Foundation. **Apache Kafka 4.0.0 Release Announcement**. 2025. Release announcement. <<https://kafka.apache.org/blog/2025/03/18/apache-kafka-4.0.0-release-announcement/>>. Accessed: 2026-07-06. Citations on pages 75 and 126.

Apache Software Foundation. **Apache Kafka Trademark Guidelines**. 2026. Project legal guidance. <<https://kafka.apache.org/community/trademark/>>. Accessed: 2026-07-23. Citation on page 126.

Apache Software Foundation. **apache/kafka**. 2026. Software repository. <<https://github.com/apache/kafka>>. Accessed: 2026-07-06. Citations on pages 75, 76, and 126.

Apache Software Foundation. **Contributor Agreements**. 2026. Foundation legal guidance. <<https://www.apache.org/licenses/contributor-agreements.html>>. Accessed: 2026-07-23. Citation on page 126.

Apache Software Foundation. **Kafka Improvement Proposals**. 2026. Improvement proposal index. <<https://cwiki.apache.org/confluence/display/KAFKA/Kafka+Improvement+Proposals>>. Accessed: 2026-07-06. Citation on page 75.

BANON, S. **Doubling down on open, Part II**. 2021. Company blog post. <<https://www.elastic.co/blog/licensing-change>>. Accessed: 2026-07-07. Citations on pages 77, 78, 84, 129, and 130.

BANON, S. **Elasticsearch Is Open Source, Again**. 2024. Company blog post. <<https://www.elastic.co/blog/elasticsearch-is-open-source-again>>. Accessed: 2026-07-07. Citations on pages 77, 78, 84, 129, and 130.

BECKER, C.; CHITCHYAN, R.; DUBOC, L.; EASTERBROOK, S.; PENZENSTADLER, B.; SEYFF, N.; VENTERS, C. C. Sustainability design and software: The karlskrona manifesto. In: **IEEE/ACM International Conference on Software Engineering (ICSE)**. 2015. v. 2, p. 467–476. Available: <<https://doi.org/10.1109/icse.2015.179>>. Citations on pages 24, 30, 31, 60, 61, 64, and 66.

BERLINGUER, M. Open source and economic models in an evolutionary approach. In: MONACO, F. J. (Ed.). **Business Models and Strategies for Open Source Projects**. Hershey, PA, USA: IGI Global, 2023. p. 18–49. Available: <<https://doi.org/10.4018/978-1-6684-4785-7.ch002>>. Citations on pages 60, 62, and 83.

- BERNTSEN, K. R.; OLSEN, M. R.; LIMBU, N.; TRAN, A. T.; COLOMO-PALACIOS, R. Sustainability in software engineering - a systematic mapping. In: **International Conference on Software Process Improvement (CIMPS)**. 2017. p. 23–32. Available: https://doi.org/10.1007/978-3-319-48523-2_3. Citation on page 31.
- BEYER, B.; JONES, C.; PETOFF, J.; MURPHY, N. **Site Reliability Engineering: How Google Runs Production Systems**. Sebastopol, CA, USA: O'Reilly Media, 2016. Available: <https://sre.google/sre-book/>. Citations on pages 25 and 46.
- CHENGALUR-SMITH, I.; SIDOROVA, A.; DANIEL, S. Sustainability of free/libre open source projects: A longitudinal study. **Journal of the Association for Information Systems**, Association for Information Systems, v. 11, n. 11, p. 657–683, Nov. 2010. Available: <https://doi.org/10.17705/1jais.00244>. Citations on pages 31 and 40.
- Cloud Native Computing Foundation. **Cloud Native Computing Foundation Announces Prometheus Graduation**. 2018. Foundation announcement. <https://www.cncf.io/announcements/2018/08/09/prometheus-graduates/>. Accessed: 2026-07-06. Citations on pages 73 and 74.
- Cloud Native Computing Foundation. **Cloud Native Computing Foundation Announces Jaeger Graduation**. 2019. Foundation announcement. <https://www.cncf.io/announcements/2019/10/31/cloud-native-computing-foundation-announces-jaeger-graduation/>. Accessed: 2026-07-07. Citation on page 76.
- Cloud Native Computing Foundation. **CNCF Charter**. 2025. Foundation charter. <https://github.com/cncf/foundation/blob/main/charter.md>. Accessed: 2026-07-22. Citations on pages 74 and 75.
- Cloud Native Computing Foundation. **CNCF Landscape**. 2025. Available: <https://landscape.cncf.io>. Accessed: 2025-10-26. Citation on page 46.
- Cloud Native Computing Foundation. **Cloud Native Computing Foundation Announces OpenTelemetry's Graduation**. 2026. Foundation announcement. <https://www.cncf.io/announcements/2026/05/21/cloud-native-computing-foundation-announces-opentelemetrys-graduation-solidifying-status-as-the-de-facto-observability-standard/>. Accessed: 2026-07-07. Citation on page 75.
- Cloud Native Computing Foundation. **Resources for CNCF Projects**. 2026. Foundation resource guide. <https://contribute.cncf.io/resources/>. Accessed: 2026-07-22. Citations on pages 74 and 75.
- CONFLUENT. **License Changes for Confluent Platform**. 2018. Company blog post. <https://www.confluent.io/blog/license-changes-confluent-platform/>. Accessed: 2026-07-06. Citation on page 76.
- CRUZES, D. S.; DYBA, T. Recommended steps for thematic synthesis in software engineering. In: **ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. 2011. p. 275–284. Available: <https://doi.org/10.1109/esem.2011.36>. Citation on page 35.
- DEEPSEEK-AI. **DeepSeek-V3 Technical Report**. 2024. Available: <https://doi.org/10.48550/arxiv.2412.19437>. Citation on page 49.

DURUMERIC, Z.; LI, F.; KASTEN, J.; AMANN, J.; BEEKMAN, J.; PAYER, M.; WEAVER, N.; ADRIAN, D.; PAXSON, V.; BAILEY, M.; HALDERMAN, J. A. The matter of heartbleed. In: **ACM Internet Measurement Conference (IMC)**. 2014. p. 475–488. Available: <https://doi.org/10.1145/2663716.2663755>. Citation on page 24.

EGHBAL, N. **Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure**. 2016. Available: <https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/>. Citations on pages 23, 30, and 60.

ELASTIC. **Contributing to Elasticsearch**. 2026. Contributor documentation. <https://github.com/elastic/elasticsearch/blob/main/CONTRIBUTING.md>. Accessed: 2026-07-07. Citations on pages 77, 78, 129, and 130.

ELASTIC. **elastic/elasticsearch**. 2026. Software repository. <https://github.com/elastic/elasticsearch>. Accessed: 2026-07-07. Citation on page 77.

ELASTIC. **Elasticsearch release notes**. 2026. Release notes. <https://www.elastic.co/docs/release-notes/elasticsearch>. Accessed: 2026-07-07. Citation on page 129.

Elastic N.V. **Annual Report for the Fiscal Year Ended April 30, 2026**. 2026. Available: <https://www.sec.gov/Archives/edgar/data/1707753/000170775326000018/estc-20260430.htm>. Accessed: 2026-07-27. Citations on pages 77 and 78.

FONTES, G.; ANDRIKOPOULOS, V.; NAKAGAWA, E. Open source software ecosystem for cloud observability: An overview and trends. In: **IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS)**. 2026. p. 9–15. Available: <https://doi.org/10.1145/3786163.3788453>. Citations on pages 17, 28, 63, 66, 70, 82, and 85.

FONTES, G.; SANTOS, V. d.; NAKAGAWA, E. Y. Is my dependency sustainable? a systematic mapping on open source software sustainability. In: **International Workshop on Designing Software (Designing)**. 2026. p. 50–57. Available: <https://doi.org/10.1145/3786152.3788587>. Citations on pages 17, 28, 63, 64, 82, and 85.

FOSTER, D. **The New Dynamics of Open Source: Relicensing, Forks, and Community Impact**. 2024. Available: <https://doi.org/10.48550/arxiv.2411.04739>. Citation on page 83.

Ganglia Development Team. **Update COPYING to the New BSD Licence**. 2010. Repository commit. <https://github.com/ganglia/monitor-core/commit/08fbf7402002876c43e03edac7dae98e3bd2e3a>. Accessed: 2026-07-23. Citations on pages 80 and 135.

Ganglia Development Team. **ganglia/ganglia-web**. 2026. Software repository. <https://github.com/ganglia/ganglia-web>. Accessed: 2026-07-07. Citations on pages 80 and 135.

Ganglia Development Team. **ganglia/monitor-core**. 2026. Software repository. <https://github.com/ganglia/monitor-core>. Accessed: 2026-07-07. Citations on pages 80 and 135.

GOGGINS, S.; LUMBARD, K.; GERMONPREZ, M. Open source community health: Analytical metrics and their corresponding narratives. In: **IEEE/ACM International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal)**. 2021. p. 25–33. Available: <https://doi.org/10.1109/soheal52568.2021.00010>. Citation on page 62.

Grafana Labs. **Our New Partnership with AWS Gives Grafana Users More Choice**. 2020. Company blog post. <<https://grafana.com/blog/announcing-amazon-managed-service-for-grafana/>>. Accessed: 2026-07-07. Citations on pages 77 and 78.

Grafana Labs. **Grafana, Loki, and Tempo will be relicensed to AGPLv3**. 2021. Company blog post. <<https://grafana.com/blog/grafana-loki-tempo-relicensing-to-agplv3/>>. Accessed: 2026-07-07. Citations on pages 77, 78, 84, and 131.

Grafana Labs. **Grafana Labs Surpasses \$400M ARR and 7,000 Customers**. 2025. Press release. <<https://grafana.com/press/2025/09/30/grafana-labs-surpasses-400m-arr-and-7000-customers-gains-new-investors-to-accelerate-global-expansion/>>. Accessed: 2026-07-07. Citation on page 78.

Grafana Labs. **Prometheus data source - Grafana Documentation**. 2025. Available: <<https://grafana.com/docs/grafana/latest/datasources/prometheus/>>. Accessed: 2025-10-30. Citations on pages 54 and 55.

Grafana Labs. **Grafana Labs Signs Strategic Collaboration Agreement with AWS**. 2026. Press release. <<https://grafana.com/press/2026/03/10/grafana-labs-signs-strategic-collaboration-agreement-with-aws-to-accelerate-open-observability-adoption-at-scale/>>. Accessed: 2026-07-07. Citations on pages 77 and 78.

Grafana Labs. **grafana/grafana**. 2026. Software repository. <<https://github.com/grafana/grafana>>. Accessed: 2026-07-07. Citation on page 78.

Grafana Labs. **Licensing**. 2026. Licensing documentation. <<https://grafana.com/licensing/>>. Accessed: 2026-07-07. Citation on page 131.

GREENSTEIN, S.; NAGLE, F. Digital dark matter and the economic contribution of apache. **Research Policy**, v. 43, n. 4, p. 623–631, 2014. Available: <<https://doi.org/10.1016/j.respol.2014.01.003>>. Citations on pages 23, 30, and 60.

HATA, H.; TODO, T.; ONOUE, S.; MATSUMOTO, K. Characteristics of sustainable oss projects: A theoretical and empirical study. In: **IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)**. 2015. p. 15–21. Available: <<https://doi.org/10.1109/chase.2015.9>>. Citation on page 37.

HAUGE, Ø.; AYALA, C.; CONRADI, R. Adoption of open source software in software-intensive organizations: A systematic literature review. **Information and Software Technology**, v. 52, n. 11, p. 1133–1154, 2010. Available: <<https://doi.org/10.1016/j.infsof.2010.05.008>>. Citation on page 23.

HILTY, L.; LOHMANN, W.; HUANG, E. Sustainability and ICT - an overview of the field. **Notizie di Politeia**, v. 27, n. 104, p. 13–28, 2011. Available: <<https://doi.org/10.5167/uzh-55640>>. Citation on page 31.

Icinga. **The Story Behind the Name: Why Icinga Is Called Icinga**. 2023. Project retrospective. <<https://icinga.com/blog/why-is-icinga-called-icinga/>>. Accessed: 2026-07-27. Citation on page 79.

INFLUXDATA. **InfluxDB Initial Public Licence**. 2013. Repository artefact. <<https://github.com/influxdata/influxdb/blob/2ed11585554de3f08822ad50f479cd287756346b/LICENSE>>. Accessed: 2026-07-23. Citations on pages 78 and 132.

INFLUXDATA. **Individual Contributor License Agreement**. 2018. Company legal agreement. <<https://www.influxdata.com/legal/cla/>>. Accessed: 2026-07-26. Citations on pages 78, 84, and 132.

INFLUXDATA. **Announcing InfluxDB 3 Enterprise Free for At-Home Use and an Update on InfluxDB 3 Core's 72-Hour Limitation**. 2025. Community forum announcement. <<https://community.influxdata.com/t/announcing-influxdb-3-enterprise-free-for-at-home-use-and-a-n-update-on-influxdb-3-cores-72-hour-limitation/56109>>. Accessed: 2026-07-07. Citation on page 78.

INFLUXDATA. **The Future of Flux**. 2026. Product documentation. <<https://docs.influxdata.com/flux/v0/future-of-flux/>>. Accessed: 2026-07-07. Citation on page 78.

INFLUXDATA. **influxdata/influxdb**. 2026. Software repository. <<https://github.com/influxdata/influxdb>>. Accessed: 2026-07-07. Citations on pages 78 and 132.

INFLUXDATA. **Which InfluxDB 3 should I use?** 2026. Product documentation. <<https://docs.influxdata.com/influxdb3/which-influxdb-3/>>. Accessed: 2026-07-07. Citations on pages 78 and 132.

Jaeger. **Update to Apache 2.0 License**. 2017. Repository commit. <<https://github.com/jaegertracing/jaeger/commit/d16de316a8bfc0ba7a6d5d1b7e43d3c48156d8ae>>. Accessed: 2026-07-23. Citations on pages 76 and 127.

Jaeger. **Jaeger v2 Released: OpenTelemetry in the Core!** 2024. Foundation blog post. <<https://www.cncf.io/blog/2024/11/12/jaeger-v2-released-opentelemetry-in-the-core/>>. Accessed: 2026-07-19. Citation on page 76.

Jaeger. **jaegertracing/jaeger**. 2026. Software repository. <<https://github.com/jaegertracing/jaeger>>. Accessed: 2026-07-07. Citations on pages 76 and 127.

Kepler. **Kepler v0.1 Licence**. 2022. Tagged repository artefact. <<https://github.com/sustainable-computing-io/kepler/blob/v0.1/LICENSE>>. Accessed: 2026-07-23. Citations on pages 76 and 128.

Kepler. **sustainable-computing-io/kepler**. 2026. Software repository. <<https://github.com/sustainable-computing-io/kepler>>. Accessed: 2026-07-07. Citations on pages 76 and 128.

KOSIŃSKA, J.; BALIŚ, B.; KONIECZNY, M.; MALAWSKI, M.; ZIELIŃSKI, S. Toward the observability of cloud-native applications: The overview of the state-of-the-art. **IEEE Access**, v. 11, p. 73036–73052, 2023. Available: <<https://doi.org/10.1109/access.2023.3281860>>. Citations on pages 50 and 51.

KOZIOLEK, H. Goal, question, metric. In: **Dependability Metrics**. Springer Berlin Heidelberg, 2008. p. 39–42. Available: <https://doi.org/10.1007/978-3-540-68947-8_6>. Citation on page 33.

LAGO, P.; KOÇAK, S. A.; CRNKOVIC, I.; PENZENSTADLER, B. Framing sustainability as a property of software quality. **Communications of the ACM**, v. 58, n. 10, p. 70–78, 2015. Available: <<https://doi.org/10.1145/2714560>>. Citations on pages 30 and 31.

LEWIS, G. A. Role of standards in cloud-computing interoperability. In: **Hawaii International Conference on System Sciences (HICSS)**. 2013. p. 1652–1661. Available: <<https://doi.org/10.1109/hicss.2013.470>>. Citation on page 46.

- LIAO, Z.; DENG, L.; FAN, X.; ZHANG, Y.; LIU, H.; QI, X.; ZHOU, Y. Empirical research on the evaluation model and method of sustainability of the open source ecosystem. **Symmetry**, MDPI AG, v. 10, n. 12, p. 747, Dec. 2018. Available: <https://doi.org/10.3390/sym10120747>. Citation on page 31.
- LINÅKER, J.; PAPATHEOCHAROUS, E.; OLSSON, T. How to characterize the health of an open source software project? a snowball literature review of an emerging practice. In: **International Symposium on Open Collaboration (OpenSym)**. 2022. p. 1–12. Available: <https://doi.org/10.1145/3555051.3555067>. Citations on pages 24, 31, 60, 62, and 93.
- LINDEN, F. van der; LUNDELL, B.; MARTTIIN, P. Commodification of industrial software: A case for open source. **IEEE Software**, IEEE Computer Society Press, v. 26, n. 4, p. 77–83, jul 2009. Available: <https://doi.org/10.1109/ms.2009.88>. Citations on pages 23, 30, and 46.
- LIU, M.; HANSEN, S.; TU, Q. Keeping the family together: Sustainability and modularity in community source development. **Information and Organization**, Elsevier BV, v. 30, n. 1, p. 100274, Mar. 2020. Available: <https://doi.org/10.1016/j.infoandorg.2019.100274>. Citation on page 31.
- LOUTAS, N.; KAMATERI, E.; BOSI, F.; TARABANIS, K. Cloud computing interoperability: The state of play. In: **IEEE International Conference on Cloud Computing Technology and Science (CloudCom)**. 2011. p. 752–757. Available: <https://doi.org/10.1109/cloudcom.2011.116>. Citation on page 46.
- MANOLEDAKI, N.; CHOOCHOTKAEW, S. **Kepler, Re-architected: Improved Power Accuracy and a Community Call to Action**. 2026. Foundation blog post. <https://www.cncf.io/blog/2026/06/30/kepler-re-architected-improved-power-accuracy-and-a-community-call-to-action/>. Accessed: 2026-07-07. Citations on pages 76 and 128.
- MASSIE, M. L.; CHUN, B. N.; CULLER, D. E. The ganglia distributed monitoring system: design, implementation, and experience. **Parallel Computing**, v. 30, n. 7, p. 817–840, 2004. Available: <https://doi.org/10.1016/j.parco.2004.04.001>. Citations on pages 80 and 135.
- MCLEAN, M.; SIGELMAN, B. **Announcing OpenTelemetry: the Merger of OpenCensus and OpenTracing**. 2019. Company blog post. <https://opensource.microsoft.com/blog/2019/05/23/announcing-opentelemetry-cncf-merged-opencensus-opentracing/>. Accessed: 2026-07-07. Citation on page 74.
- MONACO, F. J. Coherent synergy: Fostering innovation in open source ecosystems. In: MONACO, F. J. (Ed.). **Business Models and Strategies for Open Source Projects**. Hershey, PA, USA: IGI Global, 2023. p. 128–174. Available: <https://doi.org/10.4018/978-1-6684-4785-7.ch005>. Citations on pages 63, 74, 83, and 92.
- Monitoring Plugins Development Team. **Monitoring Plugins — About**. 2026. Project website. <https://www.monitoring-plugins.org/>. Accessed: 2026-07-06. Citation on page 79.
- MOURÃO, B. C.; KARITA, L.; MACHADO, I. do C. Green and sustainable software engineering - a systematic mapping study. In: **Brazilian Symposium on Software Quality (SBQS)**. 2018. p. 121–130. Available: <https://doi.org/10.1145/3275245.3275258>. Citation on page 31.
- Nagios Core Development Team. **Initial Import of Nagios Code**. 2002. Repository commit. <https://github.com/NagiosEnterprises/nagioscore/commit/a63e5a1603e8bcfb5f836ae172762d9251233ef0>. Accessed: 2026-07-23. Citations on pages 79, 84, and 133.

Nagios Core Development Team. **Contributing to Nagios Core**. 2026. Repository documentation. <<https://github.com/NagiosEnterprises/nagioscore/blob/master/CONTRIBUTING.md>>. Accessed: 2026-07-26. Citations on pages 79, 84, and 133.

Nagios Enterprises. **Nagios Core Legal Notices**. 2026. Repository legal file. <<https://github.com/NagiosEnterprises/nagioscore/blob/master/LEGAL>>. Accessed: 2026-07-23. Citations on pages 79, 84, and 133.

Nagios Enterprises. **NagiosEnterprises/nagioscore**. 2026. Software repository. <<https://github.com/NagiosEnterprises/nagioscore>>. Accessed: 2026-07-06. Citations on pages 79 and 133.

NAGLE, F.; DANA, J.; HOFFMAN, J.; RANDAZZO, S.; ZHOU, Y. **Census II of Free and Open Source Software: Application Libraries**. 2022. Available: <<https://doi.org/10.70828/kheh5209>>. Citation on page 23.

NAKAKOJI, K.; YAMAMOTO, Y.; NISHINAKA, Y.; KISHIDA, K.; YE, Y. Evolution patterns of open-source software systems and communities. In: **International Workshop on Principles of Software Evolution (IWPSE)**. 2002. p. 76–85. Available: <<https://doi.org/10.1145/512035.512055>>. Citation on page 41.

NAZIR, S.; FATIMA, N.; CHUPRAT, S.; SARKAN, H.; F, N.; SJARIF, N. N. Sustainable software engineering: A perspective of individual sustainability. **International Journal on Advanced Science, Engineering and Information Technology**, v. 10, n. 2, p. 676–683, 2020. Available: <<https://doi.org/10.18517/ijaseit.10.2.10190>>. Citation on page 30.

Net-SNMP Development Team. **History — Net-SNMP**. 2026. Project history page. <<https://www.net-snmp.org/about/history.html>>. Accessed: 2026-07-27. Citations on pages 80 and 134.

Net-SNMP Development Team. **net-snmp/net-snmp**. 2026. Software repository. <<https://github.com/net-snmp/net-snmp>>. Accessed: 2026-07-07. Citations on pages 79 and 134.

NIEDERMAIER, S.; KOETTER, F.; FREYMAN, A.; WAGNER, S. On observability and monitoring of distributed systems—an industry interview study. In: **International Conference on Service-Oriented Computing (ICSOC)**. 2019. p. 36–52. Available: <https://doi.org/10.1007/978-3-030-33702-5_3>. Citations on pages 25 and 46.

NOUREDDINE, A. **JoularJX First Public Release**. 2021. Repository commit. <<https://github.com/joular/joularjx/commit/021f4e2b9b2819cbe86d0c225a53765bb3316dd7>>. Accessed: 2026-07-23. Citations on pages 80 and 137.

NOUREDDINE, A. **PowerJoular First Public Beta Release**. 2021. Repository commit. <<https://github.com/joular/powerjoular/commit/325a5efb08fdc4566a7e76fba381040e9f9ca408>>. Accessed: 2026-07-23. Citations on pages 80 and 136.

NOUREDDINE, A. PowerJoular and JoularJX: Multi-platform software power monitoring tools. In: **International Conference on Intelligent Environments (IE)**. 2022. p. 1–4. Available: <<https://doi.org/10.1109/ie54923.2022.9826760>>. Citations on pages 80, 136, and 137.

NOUREDDINE, A. **Joular Core: The Newer Replacement for PowerJoular**. 2025. Repository issue. <<https://github.com/joular/powerjoular/issues/93>>. Accessed: 2026-07-29. Citation on page 136.

- NOUREDDINE, A. **Joular Code – Java: The New Replacement for JoularJX**. 2026. Repository issue. <<https://github.com/joular/joularjx/issues/90>>. Accessed: 2026-07-29. Citation on page 137.
- NOUREDDINE, A. **The Joular Project**. 2026. Project website. <<https://www.nouredine.org/research/joular/>>. Accessed: 2026-07-07. Citations on pages 80, 83, 136, and 137.
- NOUREDDINE, A. **joular/joularjx**. 2026. Software repository. <<https://github.com/joular/joularjx>>. Accessed: 2026-07-07. Citations on pages 80, 81, and 137.
- NOUREDDINE, A. **joular/powerjoular**. 2026. Software repository. <<https://github.com/joular/powerjoular>>. Accessed: 2026-07-07. Citations on pages 80 and 136.
- npm. **kik, left-pad, and npm**. 2016. Npm blog post. Available: <<https://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm>>. Accessed: 2026-07-30. Citation on page 24.
- Open Source Initiative. **The SSPL is Not an Open Source License**. 2021. Press release. <<https://opensource.org/blog/the-sspl-is-not-an-open-source-license>>. Accessed: 2026-07-25. Citation on page 77.
- Open Source Security Foundation. **OpenSSF Scorecard: apache/kafka**. 2026. Security assessment. <<https://securityscorecards.dev/viewer/?uri=github.com/apache/kafka>>. Accessed: 2026-07-06. Citation on page 126.
- Open Source Security Foundation. **OpenSSF Scorecard: ganglia/monitor-core**. 2026. Security assessment. <<https://securityscorecards.dev/viewer/?uri=github.com/ganglia/monitor-core>>. Accessed: 2026-07-07. Citation on page 135.
- Open Source Security Foundation. **OpenSSF Scorecard: NagiosEnterprises/nagioscore**. 2026. Security assessment. <<https://securityscorecards.dev/viewer/?uri=github.com/NagiosEnterprises/nagioscore>>. Accessed: 2026-07-06. Citation on page 133.
- Open Source Security Foundation. **OpenSSF Scorecard: net-snmp/net-snmp**. 2026. Security assessment. <<https://securityscorecards.dev/viewer/?uri=github.com/net-snmp/net-snmp>>. Accessed: 2026-07-07. Citation on page 134.
- Open Source Security Foundation. **OpenSSF Scorecard: prometheus/prometheus**. 2026. Security assessment. <<https://securityscorecards.dev/viewer/?uri=github.com/prometheus/prometheus>>. Accessed: 2026-07-06. Citation on page 74.
- OpenTelemetry. **Initial OpenTelemetry Specification Commit**. 2019. Repository commit. <<https://github.com/open-telemetry/opentelemetry-specification/commit/4e9fb68f21822414e252aae3943d3426fce0eb4d>>. Accessed: 2026-07-23. Citations on pages 75 and 125.
- OpenTelemetry. **Prometheus and OpenTelemetry - Better Together**. 2024. OpenTelemetry blog post. Available: <<https://opentelemetry.io/blog/2024/prom-and-otel/>>. Citation on page 55.
- OpenTelemetry. **OpenTelemetry Graduation Application**. 2025. Graduation application. <<http://github.com/cncf/toc/issues/1739>>. Accessed: 2026-07-22. Citation on page 75.
- OpenTelemetry. **Specification Status Summary**. 2025. Project specification documentation. <<https://opentelemetry.io/docs/specs/status/>>. Accessed: 2026-07-27. Citation on page 75.

OpenTelemetry. **Contributor License Agreement**. 2026. Contributor guide. <<https://github.com/open-telemetry/community/blob/main/guides/contributor/CLA.md>>. Accessed: 2026-07-23. Citations on pages 75 and 125.

OpenTelemetry. **open-telemetry/community**. 2026. Governance repository. <<https://github.com/open-telemetry/community>>. Accessed: 2026-07-07. Citations on pages 75 and 125.

OpenTelemetry. **Prometheus Client Libraries vs. OpenTelemetry**. 2026. Project compatibility guide. <<https://opentelemetry.io/docs/compatibility/prometheus/client-libraries/>>. Accessed: 2026-07-27. Citation on page 75.

PAKHALE, K. **Comprehensive Overview of Named Entity Recognition: Models, Domain-Specific Applications and Challenges**. 2023. Available: <<https://doi.org/10.48550/arxiv.2309.14084>>. Citation on page 48.

PENZENSTADLER, B.; BAUER, V.; CALERO, C.; FRANCH, X. Sustainability in software engineering: A systematic literature review. In: **International Conference on Evaluation and Assessment in Software Engineering (EASE)**. 2012. p. 32–41. Available: <<https://doi.org/10.1049/ic.2012.0004>>. Citations on pages 30 and 31.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, Elsevier BV, v. 64, p. 1–18, 2015. Available: <<https://doi.org/10.1016/j.infsof.2015.03.007>>. Citations on pages 32, 43, and 64.

PICORETI, R.; CARMO, A. P. do; QUEIROZ, F. M. de; GARCIA, A. S.; VASSALLO, R. F.; SIMEONIDOU, D. Multilevel observability in cloud orchestration. In: **IEEE International Conferences on Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing; Big Data Intelligence and Computing; and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)**. 2018. p. 776–784. Available: <<https://doi.org/10.1109/dasc/picom/datacom/cyberscitech.2018.00134>>. Citations on pages 25 and 46.

PIJNACKER, B.; SETZ, B.; ANDRIKOPOULOS, V. Container-level energy observability in Kubernetes clusters. In: **International Conference on ICT for Sustainability (ICT4S)**. 2025. p. 69–79. Available: <<https://doi.org/10.1109/ict4s68164.2025.00016>>. Citations on pages 76 and 128.

POPP, K.-M. A business model framework for open source software companies. In: MONACO, F. J. (Ed.). **Business Models and Strategies for Open Source Projects**. Hershey, PA, USA: IGI Global, 2023. p. 1–17. Available: <<https://doi.org/10.4018/978-1-6684-4785-7.ch001>>. Citations on pages 63, 83, and 92.

Prometheus. **Add Apache License 2.0 Boilerplate**. 2012. Repository commit. <<https://github.com/prometheus/prometheus/commit/44f8802ae76506fd17e0acb0a1261c9ce78a0710>>. Accessed: 2026-07-23. Citations on pages 74 and 124.

Prometheus. **Community**. 2026. Project website. <<https://prometheus.io/community/>>. Accessed: 2026-07-06. Citation on page 73.

Prometheus. **Contributing to Prometheus**. 2026. Contributor guide. <<https://prometheus.io/docs/guides/contributing/>>. Accessed: 2026-07-23. Citations on pages 74 and 124.

Prometheus. **Prometheus Governance**. 2026. Governance document. <<https://prometheus.io/governance/>>. Accessed: 2026-07-06. Citations on pages 73 and 74.

Prometheus. **prometheus/prometheus**. 2026. Software repository. <<https://github.com/prometheus/prometheus>>. Accessed: 2026-07-06. Citations on pages 73, 74, and 124.

Prometheus. **Using Prometheus as Your OpenTelemetry Backend**. 2026. Technical guide. <<https://prometheus.io/docs/guides/opentelemetry/>>. Accessed: 2026-07-07. Citation on page 75.

RAZAVIAN, M.; PROCACCIANTI, G.; TAMBURRI, D. A. Four-dimensional sustainable e-services. In: **International Conference on Environmental Informatics (EnviroInfo)**. 2014. p. 221–228. Citation on page 30.

Red Hat. **Monitoring Performance with Net-SNMP**. 2026. Red Hat Enterprise Linux documentation. <https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/deployment_guide/sect-system_monitoring_tools-net-snmp>. Accessed: 2026-07-27. Citations on pages 80 and 134.

RIEHLE, D. The single-vendor commercial open source business model. **Information Systems and e-Business Management**, Springer, v. 10, n. 1, p. 5–17, 2012. Available: <<https://doi.org/10.1007/s10257-010-0149-x>>. Citations on pages 63, 77, and 83.

ROBLES, G.; STEINMACHER, I.; ADAMS, P.; TREUDE, C. Twenty years of open source software: From skepticism to mainstream. **IEEE Software**, v. 36, n. 6, p. 12–15, 2019. Available: <<https://doi.org/10.1109/ms.2019.2933672>>. Citation on page 23.

SANTOS, C.; KUK, G.; KON, F.; PEARSON, J. The attraction of contributors in free and open source software projects. **The Journal of Strategic Information Systems**, Elsevier BV, v. 22, n. 1, p. 26–45, Mar. 2013. Available: <<https://doi.org/10.1016/j.jsis.2012.07.004>>. Citations on pages 37 and 40.

SCHWEIK, C. M.; ENGLISH, R. C. **Internet success: a study of open-source software commons**. Cambridge, MA, USA: MIT Press, 2012. Available: <<https://doi.org/10.7551/mitpress/9780262017251.001.0001>>. Citation on page 30.

The Linux Foundation. **Linux Foundation Announces OpenSearch Software Foundation**. 2024. Foundation announcement. <<https://www.linuxfoundation.org/press/linux-foundation-announces-opensearch-software-foundation-to-foster-open-collaboration-in-search-and-analytics>>. Accessed: 2026-07-07. Citations on pages 77 and 78.

The Linux Foundation. **Trademarks**. 2026. Foundation legal page. <<https://www.linuxfoundation.org/legal/trademarks>>. Accessed: 2026-07-23. Citations on pages 74, 124, 125, 127, and 128.

UCD-SNMP Development Team. **Early UCD-SNMP Copyright Notice**. 1996. Repository artefact. <<https://github.com/net-snmp/net-snmp/blob/ea98b50a7ee85c484274643c0ce0ea73093449c/COPYING>>. Accessed: 2026-07-23. Citations on pages 79 and 134.

VENTERS, C. C.; LAU, L.; GRIFFITHS, M. K.; HOLMES, V.; WARD, R. R.; JAY, C.; DIBSDALE, C. E.; XU, J. The blind men and the elephant: Towards an empirical evaluation framework for software sustainability. **Journal of Open Research Software**, v. 2, n. 1, p. 1–6, 2014. Available: <<https://doi.org/10.5334/jors.ao>>. Citation on page 31.

VOULVOULIS, N.; GIAKOUMIS, T.; HUNT, C.; KIOUPI, V.; PETROU, N.; SOULIOTIS, I.; VAGHELA, C.; ROSELY, W. binti W. Systems thinking as a paradigm shift for sustainability transformation. **Global Environmental Change**, v. 75, p. 102544, 2022. Available: <<https://doi.org/10.1016/j.gloenvcha.2022.102544>>. Citation on page 30.

WACHS, J.; NITECKI, M.; SCHUELLER, W.; POLLERES, A. The geography of open source software: Evidence from github. **Technological Forecasting and Social Change**, Elsevier BV, v. 176, p. 121478, Mar. 2022. Available: <<https://doi.org/10.1016/j.techfore.2022.121478>>. Citation on page 30.

YIN, L.; CHAKRABORTI, M.; YAN, Y.; SCHWEIK, C.; FREY, S.; FILKOV, V. Open source software sustainability: Combining institutional analysis and socio-technical networks. **Proceedings of the ACM on Human-Computer Interaction**, Association for Computing Machinery (ACM), v. 6, n. CSCW2, p. 1–23, Nov. 2022. Available: <<https://doi.org/10.1145/3555129>>. Citation on page 30.

ZAHAN, N.; KANAKIYA, P.; HAMBLETON, B.; SHOHAN, S.; WILLIAMS, L. OpenSSF scorecard: On the path toward ecosystem-wide automated security metrics. **IEEE Security & Privacy**, v. 21, n. 6, p. 76–88, 2023. Available: <<https://doi.org/10.1109/msec.2023.3279773>>. Citation on page 62.

OSS Sustainability Fragments and Keywords

This appendix presents the sustainability-related fragments and keywords extracted from the studies selected in Chapter 2. Study identifiers correspond to Table 1 and link to their DOI records. The extracted keywords are highlighted in bold and visualized collectively in Figure 5.

Table 9 – Sustainability fragments and keywords extracted from the selected studies

ID	Relevant fragment(s); extracted keywords in bold
S001	[...]its sustainability could be threatened by that complexity or non-uniform evolution of some packages.
S002	The sustainability of an OSS project is heavily based on the underlying community of contributors and on the knowledge and skills they bring to the project
S003	[...] we may confidently conclude that ‘sustainable systems’ are at least systems that suit the needs of users [...]. Beyond that, sustainable systems will have to take ecological and social requirements into account. it is not only the free and open character of the software but also the participative development process that is a factor of sustainability.
S004	One of the more complex aspects of the original analysis is the qualification of release rate as an indicator of the sustainability of project activity. The assumption here is that projects which make releases too quickly cannot maintain the pace of activity [...].
S005	[...]we can say that the sustainability of OSS is closely related to three factors – community growth , financial resources and software management [...]. [...] growth and continuity of the community can result more naturally in financial resources that, if well managed, can be reversed in benefits to the community and encourage its growth, feeding the OSS sustainable life cycle. Factors related to source code structure can greatly contribute to OSS projects sustainability[...]. The code with a modular structure is an incentive for developers to enter and to continue in the project development.
S006	[...] it provides information on the outcome of the forks , in order to see if forking undermines the sustainability of the projects.
S007	An growing number of users and a high probability of new revisions being published is correlated with the sustainability of the project.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S008	In the context of open source software, this includes raising the quality standards of products by implementing more complex processes and rigorous methodologies .
S009	[...] compatibility of a foundation with the projects' needs is a crucial factor in ensuring the project's sustainability
S010	In this paper we address the role of code forking [...] in ensuring the long-term sustainability of a software system. We take the view of the consumer and focus on two central elements: quality and staying power – how to create a high-quality product that is usable as long as possible . The possibility to fork is one of the key factors that ensure that open source will continue to evolve and thus remain sustainable.
S011	projects depend on the continuity of their development communities to remain sustainable
S012	[...] in order to examine the sustainability potential of the open source model it is important to understand the relationship between contribution motivations and the threat of external appropriation the sustainability of open source projects depends to a large extent on contributors' willingness to contribute to such projects, which is, in turn, affected by contributors' motivations, fairness, and perceptions of the open source model's justice .
S013	OSS development project sustainability is defined as the ability of the project to exhibit software development and maintenance activity over the long term Make our software {stable (reduce bugs)} Feature set Documentation Reduce time between releases Attract users/contributors Keep users/contributors involved Leverage the power of the leader/main company Fundraising Legal issues (license, copyright assignments) Communication inside the project Communication upstream/downstream
S014	Community governance The role of commercial affiliates Maintaining the family atmosphere Cross-project knowledge sharing Project coordination

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S015	The results demonstrated a temporal persistence in the relationship between sustainability and the speed of defect-removal and functionality-enhancement , whereas the effect that the defect-removal rate and functionality-enhancement rate have on development sustainability was discovered to significantly decrease as the project grows old . Several researchers have studied open source software. Many of these have studied the various factors that drive OSS projects' development sustainability, including developer and user attraction , development base and project age , having developers with higher levels of different skills , project status and activity , having a nonmarket sponsor , and having a copy-left licence . In the existing OSS literature, there are few longitudinal studies that investigate the relationships between OSS sustainability and its antecedents. As an example, Midha [...] investigated the potential impact of certain factors (e.g. software complexity and modularity). [...] Subramaniam et al. examined the impact of some time-dependent predictors (e.g. project status) and some time-invariant predictors (e.g. project licence).
S016	[...] examine what attracts contributors , creating an environment likely to improve and promote the project sustainably. As commercial organizations increasingly sanction an open source strategy, the user uptake will determine the level of sponsorships , and the sustainability of FOSP [...] groups of visitors, users and developers provide FOSP with varying and unique sets of development resources that together promote and improve the project towards the sustainability of recruiting new resources and generating more work activities and other indirect contributions[...].
S017	For this reason, sustainability of communities has been identified as essential for long-term sustainability of OSS. Earlier research also suggests that an effective structure of governance is a basis for healthy and sustainable OSS communities. [...] aspects such as clear leadership , congruence in terms of project goals , and good team spirit are of fundamental importance. [...]. Further, the licensing of OSS may affect the community.
S018	The long-term viability of a technical solution . The concern being that if the project contributors overreach their collective abilities and their capacity to develop and maintain good quality software, then there is a risk the project may cease to be viable.
S019	Maintaining and increasing the populations in software development communities are challenges of OSS projects for sustainability. Yamashita et al. proposed a pair of population metrics, namely, magnetism and stickiness To increase the utility of writing code compared to the utility of just discussing, projects need to setup the development environment , which can decrease the cost of writing code . Employment is a big incentive to write code. The project itself or other third-parties can select this option. Although it is not easy to make innovations, innovations can decrease the cost and may increase the reward . For example, developing new tools like Git, a version control system, and deploying new services like GitHub, a web-base hosting service and social networking system for developers, can be regarded as such innovations.
S020	Atiq et al. point out that the importance of financial support for the progress of OSS projects is well-recognized by the research community. And they find that proper management is essential to the projects' sustainability, even if the financial benefits received by the developers are unequal. With the guidance of the overjustification theory, we analyze and the changes of users' contributions and responses over time and discover that they become significantly more active after getting sponsorships . And we also find that providing sponsorships to developers can improve their own or contributed projects.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S021	[...] a continuous influx of people willing to contribute is essential to a FOSS project's sustainability.
S022	[...] identify the types and instances of citizenship behaviors that are critical to the functioning and sustainability of FLOSS communities.
S023	The overarching goal of this study is to bring to attention the importance of interpersonal trust in OSS sustainability.
S024	To secure sustainability, OSS projects need to maintain a healthy influx of newcomers
S025	A large base of voluntarily contributing members is one of the most important success factors of OSS[...]
S026	As an example, the modularity of a software architecture , which is considered a key feature that contributes to the sustainability of large scale projects [...]
S027	Governance rules enable the coordination of developers in order to advance the project during its whole lifespan and, more importantly, their evolution allows the project to adopt societal changes on the way people want to collaborate, thus promoting the sustainability of the development process.
S028	A key concern in any FLOSS project is its sustainability, and a major factor that affects this is the project's ability to retain contributors
S029	Sustainability: to promote the project to fulfill its goals , to survive changes , and to incorporate new demands over time . Maintainers perform an unquestionably central work to the success and long-term sustainability of communities. [...] In fact, we found that Communication was considered the most important attribute of a great maintainer with Quality Assurance being the second-ranked [...].
S030	The sustainability and long-term survival of Open Source Software (OSS) projects depend not only on attracting but, more crucially, retaining motivated developers . The reasons behind a developer's decision to stay or leave an OSS project can depend on different intrinsic or extrinsic factors, including an individual's feelings of identity and belonging to the community.
S031	Sustainable Open Source Software (OSS) projects are characterized by volunteering work, continuous recruitment , and effective governance . [...] sustainable projects during episodic changes can adapt themselves to institutional statements more efficiently [...] changes in self-governance can change the trajectory of project sustainability. Sustainable projects can process and translate self-governance rules and policies into sociotechnical changes , and vice versa, more effectively than unsustainable projects.
S032	To help with sustainability, companies contribute financially as well as upstream fixes which in return prevents carrying any technical debt internally. To create a sustainable project, companies need to open-source projects they use and care about internally.
S033	[...] we consider research software as sustainable if it has a long life span and remains live.
S034	[...] commercial participation also brings challenges and risks to the long-term development of OSS and the sustainability of critical open source ecosystems.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S035	Most projects carefully regulate admission of outsiders to full developer privileges [...]. Understanding the factors that influence the who, how and when of this process is critical, both for the sustainability of FLOSS projects, and for outside stakeholders who want to gain entry and succeed.
S036	The stability and permanence of [...] most active developers is of great importance for the evolution and sustainability of the project.
S037	attracting new and retaining old contributors for achieving sustainable success
S038	To measure the sustainability of the developer community participation, we [...] measure the contribution of the developer community in terms of software configuration management artifacts . [...] we measure commit counts as a proxy of developer community participation and use commit counts per time-interval to measure the sustainability of the developer community participation.
S039	Since volunteer developers are free to participate in the development and also free to leave, OSS projects always seek newcomers and their contributions to make the projects sustainable
S040	In recent years, researchers and practitioners attempted to defined catalogs of MCR anti-patterns , which become a major problem that hinders software quality, maintainability and sustainability Hence, if reviews with divergent scores are not carefully resolved, they may contribute to a tense reviewing culture and may slow down integration and lead to detrimental effects on contributors' continuing participation in the community affecting the sustainability of the project
S041	The bus factor risk is significant for project sustainability. Archived projects tend to have a smaller number of bus factor contributors, a higher PR acceptance rate , and a longer issue response duration .
S042	Less complex code may facilitate the addition of new features and bug fixing in a sustainable way and therefore supports maintainability .
S043	Sustainability is signaled by three factors – community growth , financial resources , and software management Researchers define project health through the collection of success measures; including metrics related to project output, process, and the outcomes for project members. In this context, success is the result of project activity and the release of code, otherwise the project is abandoned. Further measures of success explore the growth and diversity of projects. Activity measures tell you when a project is finished; health and sustainability measures would identify that trajectory in advance. While sustainability evokes shared commitment , survivability aims to surface indications of a project's vigor, resilience, and organization in the face of risk Sustainability and survivability, together, aspire to provide clear signals of a project's likelihood to continue producing quality software . Much of the early literature on open source project health focused on success measures, however as open source health research has matured, the focus has moved to measures of sustainability and further, to an understanding that open source health must include considerations for social interactions and project diversity when estimating survivability. Quality model - product quality, process maturity and sustainability Development Base (size), project age and the size of niche
Patch contribution and patch process	
Community growth, financial resources and software management	
Relationships among people, technologies, and organizations Virtuous Circle - Feedback loop of open source best practices	

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
	Ability to Fork
	Commits, retention of committers
S044	The sustainability of an OSS project refers to the potential initiative to maintain the functionality and evolution of the software system in the future
S045	Furthermore, peripheral developers come with extensive external social networks that are important for the sustainability of FLOSS projects Given the observed prevalence of EV [Episodic Volunteering], and the limitations to volunteers becoming habitual, the effective incorporation of episodic volunteers may become an important competency in FLOSS project sustainability.
S046	[...] it is of particular interest to understand how episodic contributors can be 'retained' [...]. Retention is appealing because returning contributors require less assistance than newcomers and retention is one of the key factors in FLOSS project sustainability.
S047	The sustainability of OSS projects has attracted great interest from researchers. Previous studies have investigated the motivations behind and barriers to developers' joining and retention in OSS projects.
S048	For an OSS project, good support to newcomers ' early career would help retain its most valuable asset, and motivate newcomers to make sustainable contributions, hence reducing the high turnover problem.
S049	Investigating mentors' engagement can help OSS communities to devise support mechanisms to optimize the mentoring process and further promote the healthy and sustainable development of communities. The sustainability of OSS projects depends on the continuous inflow and retention of newcomers
S050	[...] it is essential to motivate, engage, and retain new developers in order to promote a sustainable community in a project.
S051	Many projects leverage the contribution of outsiders and the sustainability of the project relies on retaining some of these newcomers.
S052	We find systematic differences among modules; these differences are stable over time, which suggests that certain architectural features, commercial interests , or module-specific practices lead to distinct sustainable equilibria. Our proposed framework to quantify maintainer practices and productivity scaling may lead to a better understanding of the factors that allow rapidly growing projects to be sustainable and to practices that reduce risk of project failures
S053	[...] projects may be considered sustainable from a code maintainability perspective if they conform to modular and extensible architectures , from a community perspective if they successfully attract and retain newcomers ; and from an economic perspective if they ensure low total cost of ownership and high added value . For example, competition was listed as a major driving force behind one project.
S054	[...] the model proposed in this paper can be used by developers to check the maintenance status of an open source project, before deciding to use it. This information has a key value, since there is a growing concern on the sustainability of modern open source projects.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S055	In this work, we analyze open source projects to determine whether they exhibit a rich-club behavior, i.e., a phenomenon where contributors with a high number of collaborations (i.e., strongly connected within the collaboration network) are likely to cooperate with other well-connected individuals. The presence or absence of a rich-club has an impact on the sustainability and robustness
S056	Studying what makes projects attractive is especially important because, as opposed to individual motivation which is typically inherent to the potential contributors, project attractiveness can be to a larger extent controlled by the project maintainers, as we will argue in the remainder of this paper. Therefore, increasing project attractiveness has the potential not only to reduce some onboarding barriers , but also to improve the sustainability of open-source projects. Indeed, among the 11 participants who talked about team size , five mentioned reasons why a big project may be a better choice for newcomers. One reason is that with more contributors in the team, the project can be more sustainable. [...] stakeholders could focus on developing reliable new signals for the less readily observable project qualities we identified as important. Ultimately, these signals could help direct contributor effort to open-source projects where this effort is most needed, contributing to the sustainability of open-source ecosystems as a whole. [...] lack of time or interest of the main contributors poses serious sustainability risks. Recruiting new contributors can, therefore, help ensure the sustainability of open-source projects.
S057	For example, Zhou et al. found that the distribution of work among the Linux kernel maintainers followed the 80:20 rule for most modules, suggesting that a few maintainers may bear the brunt of the increased workload. These issues have caused OSS projects to increasingly worry about their sustainability.
S058	One perceived risk of using FOSS software in commercial environments is its sustainability. [...] . I propose to establish evidence of sustainability measures for software quality taking place in FOSS communities. I define sustainability of quality as the ability of the community to continuously deliver software of industrial standards . My PhD study is to demonstrate that FOSS communities gain sustainability for quality through participation motives (e.g. hackers ethics), a governance mechanism for software change process , and the adoption of good practices in the pull request (PR) evaluation process . Participation sustainability is an issue in some communities. For a FOSS project to remain sustainable, it must evolve with its user base and its developer base.
S059	[...] the community always needs newcomers , as creativity requires fresh minds and an ongoing flow of ideas and new contributions . Ostracizing those who are not inside the community may hinder its evolution and sustainability. Each PR governance reduces the threat of poor code. Consistency and governance create a culture of excellence. [...] This perspective contribute to the sustainability of software quality in FOSS.
S060	These roles , although known and important for the projects' sustainability and community evolution , are often not formally recognized in OSS communities.
S061	[...] sustainability is a measure related more to the human and social aspect (e.g., the ability to take responsible collective action , and an open and inclusive atmosphere , etc.) than the technical aspect (defect density, technical advantage, etc.).
S062	A more relevant [donation] mechanism is needed to promote the healthy and sustainable development of the ecosystem.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S063	Financial support for OSS maintainers and developers is a major issue in terms of sustaining OSS projects, and the ability to donate to individuals is expected to support the sustainability of developers, projects, and community.
S064	A better understanding of the effectiveness of using social media to attract attention to OSS projects could directly impact the projects' success and sustainability. For example, prior work found that popular OSS projects are perceived as having higher quality and better community support , tend to be more attractive to new contributors, and tend to be more successful at fundraising . That is, they tend to be more sustainable.
S065	One major challenge of OSS sustainability is to attract and retain capable newcomers . To help newcomers become familiar with an OSS project, GitHub provides a list of best practices [...]. Even if sufficient documentation is provided, it is still very challenging for newcomers to locate suitable development tasks to start with.
S066	Contributors are vital to the sustainability of open source ecosystems, and disengagement threatens that sustainability.
S067	To achieve commercial goals , companies have made substantial contributions to large open-source software (OSS)[...]. However, they often withdraw their employees for a variety of reasons, which may affect the sustainability of OSS projects. While the turnover of individual contributors has been extensively investigated, there is a lack of knowledge about the nature of companies' withdrawal .
S068	Seeking less uncertainty, many OSS projects join established software communities [...] to guide projects toward sustainability. In recent work we showed that OSS project sustainability can be effectively predicted [...] from longitudinal project and process metrics supplemented by socio-technical network metrics (developer communications and code contributions), [...].
S069	We, therefore, propose the concept of software digital sociology that represents mining abundant software-activity data to achieve such understanding. The research objects range from individual learning , group collaboration , and ecosystem sustainability , along with a particular focus on the complex dependencies coined as software supply chain.
S070	[...]. These strengths are threatened when a single organization can exert exclusive control over the future evolution of an OSS project, which in turn can jeopardize the future sustainability of OSS projects and ecosystems. The findings of this study may help OSS communities to increase their awareness of this issue and the associated risks, and adjust their governance mechanisms so as to ensure their future sustainability. As corporate participation in OSS ecosystems is growing, their influence on the future evolution of these ecosystems also becomes stronger, which in turn can have significant consequences for the sustainability of these ecosystems.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S071	[...] we hypothesize that sustainable projects may exhibit different code and process patterns than unsustainable ones, and that those patterns can grow more apparent as projects evolve over time. Apache Software Foundation (ASF), which provide common standards and guidance in exchange for higher community uniformity. The popularity and high standards of foundation supported projects can contribute to their continued sustainability, by attracting a steady supply of programmer effort. Those studies found that different patterns of social interactions and different socio-technical behavior mediate differential success outcomes, in particular with respect to sustainability metrics. This paper is a first step showing that it is possible to associate sustainability with code, quality and process metrics. During the incubation, the projects' long-term goal is to become self-sustainable, i.e., the project's community can self-govern itself to sustain its activity and productivity over time. To help incubating projects achieve such level of sustainability, ASF, different from most OSS foundations, provides each incubating project with in-depth mentorship from senior ASF committers. More recently, Yin et al. showed that socio-technical factors (mostly social and technical networks) can be used to forecast the incubation sustainability outcome for ASF incubator projects. They found that a higher bug-fixing and feature enhancement rate, together with an increase in the frequency of releases helps with long term sustainability. From a sustainability perspective, minor contributors could become committers and maintainers of the project. Thus, projects and maintainers should consider attracting and providing guidance to minor contributors as a way of expanding the project's community.
S072	[...] refers to a project's ability to be maintained and developed over time, including aspects such as governance and community engagement.
S073	In this paper, we make the first effort toward understanding OSS project sustainability using a dual-view analysis, by combining institutional analysis with socio-technical systems analysis. In particular, we (i) use linguistic approaches to extract institutional rules and norms from OSS contributors' communications to represent the evolution of their governance systems, and (ii) construct socio-technical networks based on longitudinal collaboration records to represent each project's organizational structure. In software engineering, the focus has been on understanding success and sustainability from the socio-technical perspective: the OSS developers' day-to-day activities and the artifacts they create. In the management domain, on the other hand, emphasis has been on institutional designs [...] that structure governance and OSS project administration. [...] key hurdles that OSS projects have to demonstrate to graduate is that they can (1) produce new releases, and (2) show the ability to attract new developers. Both of these factors arguably are key to the sustainability of OSS projects.
S074	A large base of voluntarily contributing members is one of the most important success factors of OSS[...]
S075	Steinmacher et al. argued that the sustainability of many OSS projects relies on retaining newcomers. They discussed some barriers faced by newcomers to OSS [6].

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S076	Sustainability is defined in this paper as the ability of the project to exhibit software development and maintenance activity over the long term . Sustainability is conceptualized as the ability to remain active for a longer period of time . The ability to attract resources is an indicator of the perceived project legitimacy , which in turn is a strong predictor of the project's future sustainability. [...] we believe that the number of open and closed artifacts is an appropriate measure of the sustainability of a FLOSS project. [...] project demographic characteristics such as age and size , as well as the niche size occupied by the project, influence the project's ability to attract and retain user and developer resources in the future.
S077	[...] inspired by the vital importance of sustainability as a concept that expresses the need to create the necessary conditions for future generations to use and evolve present artifacts , we target the software engineering domain and propose a systematic way towards measuring the extent to which a software artifact developed and applied in the cultural heritage domain is sustainable. This fact is more than evident considering that maintainability (the official term for sustainability in the software engineering language) is one of the most important quality characteristics according to ISO/IEC 25010:2011
S078	Accordingly, we define open source ecosystem sustainability as the potential initiative of the software ecosystem to permanently sustain or support its own dynamic health and its evolutionary evolution . This definition includes two meanings: First, the system is now in a state of steady development ; second, the system will continue to be in that state in the future . Openness : The ability of the entire ecosystem to communicate and transformation with the outside environment or within the ecosystem. Integrity : The internal composition, structure and function of an ecosystem and the integrity of its external biophysical environment. Stability : The anti-interference ability of the structure, state and behavior of the ecosystem. Including avoidance, tolerance and resilience. Regulation : The ecosystem has certain resistance to interference and has a certain ability to recover after being disturbed. It can coordinate and maintain stability. Resilience : The ability of ecosystems to maintain function under pressure. Diversity : Rich and balanced species within the ecosystem. Productivity : The biological production capacity of the ecosystem. Organizational Structure : Components and structures in the ecosystem. Drivers : Natural or man-made disturbances or stress on ecosystems that change ecosystems. Propensity for Growth : Ecosystem growth is good. Activity. Conform to Development Trend : The ability to meet the needs of contemporary people. Extensibility.
S079	[...] many people don't understand how Open Source could be economically sustainable, and some may even feel that its potential negative effect upon the proprietary software industry is an overall economic detriment. Fortunately, [...] Open Source is both sustainable and of tremendous benefit to the overall economy .
S080	Our research identified several factors that influence the performance of the projects, including the less-explored role of core developers and the importance of promoting the projects .
S081	[...] [this research] will help FLOSS communities in understanding the factors that contribute to the successful socialization of new members . It will thus help communities to tailor proper socialization initiatives that lead to behaviours that match community values , and increase the community's sustainability

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S082	[...] this research project will help FOSS communities in understanding the factors that characterize the socialization of new members and will thus help communities in better designing socialization initiatives and programs [...] that will contribute to the survival, sustainability, and overall long-term success of FOSS communities.
S083	Since the current model of OSS development depends on this volunteer workforce, who are ideologically motivated , introducing compensations/rewards that are not transparent and community oriented, may alienate volunteers and threaten the sustainability of OSS communities. As volunteers' motivation to associate with , and contribute via these communities is vital to the sustainability of OSS development [...] Developers perceive asymmetry in compensation/rewards when they believe that funds/resources available to a project have been distributed unequally/unfairly/asymmetrically . Our findings suggest that OSS projects where only some people get financial benefits may fail if they are mismanaged . From our empirical research, we suggest that fair-terms, transparency through effective communication are essential to OSS project sustainability, even if the governing agency have a policy of unequal financial benefits.
S084	The contributions of this paper include [...] providing a quick reference check for those interested in conducting research on understanding developer forking motivation reasons and consequences [...] to predict project survivability and sustainability [...].
S085	The social sustainability of a community depends on the individual characteristics of its members , on its size and form , and the division of labor and power in the community. Cultural aspects of interaction depend on interpretation, language, and coherent patterns of behaviour . The most recent economic literature is dealing with the companies'
S086	An improved patch contribution process will lower the contribution barrier , helping to improve the sustainability of critical open source projects. Our study of open source projects generally supports the idea that a managed patch contribution process improves project sustainability.
S087	Architecture, Maintainability, Security and testing, License, Market, Support, Initialization, Dependencies, Reuse, Development process and governance, Developer base, User base. Robustness, Scalability, Usability, Effectiveness, Corrections, Improvements, Security, Test process, Coverage, License Type, Dual Licensing, Commercial resources, Commercial training, Industry Adoption, Nonprofit / foundation support, For profit / company support, Donations, Installability, Configurability, Self Contained, Resource utilization, Complexity, Modularity, Instability, Cohesion, Governance model, Project roadmap, Code of conduct, Documentation standards, Coding standards, Developer attracted, Active developers, Number of open issues, Open versus closed issues, Source code documentation, Localization process, Issue tracking activity, User guide.
S088	We posit that a project's need for sustainability support can be determined by comparing measures of active use to measures of active maintenance .
S089	[...] firms control who can commit to a given repository and who cannot (typically, only firm employees can commit, whereas others need to obtain approval via pull requests). This partly explains why firms remain the main contributor to a repository of a technology they initially released. The technology is therefore 'open source' in terms of the code license, but is not developed following an open or shared governance model whereby developers from different firms collectively decide its future direction (the Linux kernel is an obvious exception, as it features contributions from a very diverse group of firms).

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
	It seems less controversial to suggest that increased firm involvement is likely to influence project decisions of all kinds, from technical to licensing ones. And indeed the open status of industrial public goods is now at risk. We mapped a firm-project FOSS coproduction network on GitHub, which was dominated by a minority of key players, such as Linux and Microsoft. We found diverse firm contributory models, including quasi-monopolies, indirect cooperation between firms, and extreme fragmentation in the case of Linux. Despite paid workers making the majority of contributions, unpaid workers played a significant role, indicating the informal nature of these hybrid work arrangements. Our mapping of exclusive firm ‘contribution territories’ led us to define indirect firm cooperation as ‘selective’ and raises the question of the existence of ‘contribution deserts’ – projects and code that are neglected by large IT firms and can only rely on volunteer labor, despite being important for the sustainability and diversity of the open source ecosystem.
S090	The practical significance of this research lies in the ability to accurately predict developer churn, enabling open-source communities to proactively retain core developers and enhance the sustainability of their projects. By leveraging historical data, this model takes into consideration various factors such as developers’ activities, collaborative networks, and community context information.
S091	In this paper, we use sustained activity as the main proxy for studying OSS sustainability, following previous work. We further find that issue comments from non-code contributors (#iss_comment_n) are positively related to sustained activity. Our results suggest that projects, especially organizational projects, should pay attention to the retention of experienced core developers (with higher #issue_all_c, #pro_oneyear_c) and active peripheral contributors (for higher cmt_dev_std, #cmt_p). Therefore, we suggest that initial project maintainers should be careful not to let these “process” activities take up too much time and effort, since some non-code activities take resources and sometimes slow down development activities. Finally, since the presence of organizations and companies has largely positive effects in our analysis, we are interested in the underlying mechanisms and the roles that organizations and companies may play during OSS launching.
S092	Without an active and engaged community of users, the software fails to solve a real-world problem and will not be impactful. [...] if the software continues to be relevant, then community managers or equivalent should be engaged to ensure the community is active, provides feedback and reports bugs [...]. Maintaining and sustaining code is simplest when a community is most active; however, even without an active community, code should be routinely updated and maintained. Bug fixing is required to ensure the software remains trustworthy, and implementing new features is required to ensure the software remains relevant. Automated testing of code is best practice and should be implemented by developers whenever possible. [...] developers should also consider how stakeholder engagement and end-user testing should be incorporated into ongoing testing and maintenance.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
	Finally, funding for software can be acquired via grant funding , creating software as a service , offering tiered subscriptions (e.g., for quicker bug fixing), dual-licensing , and other models—there is no turnkey solution for financial sustainability and all software will need a bespoke solution. [...]
S093	[...] we use the number of commits over time to characterize the sustainability of projects.
S094	Sustainability is particularly visible in the complex and often brittle dependency chains in OSS library ecosystems like npm, PyPi, and CRAN.
S095	The risk of packages severely losing downloads (i.e. peaking) decreases when they are more consistently maintained. [...] the effects of no maintenance, particularly read-only archived projects, on package downloads decreases for smaller projects [...] [...] projects progressively become less satisfactory unless continually adapted. [...] small and widely-reused projects to be feature-complete and less dependent on maintainers to thrive.
S096	The emerging threat of toxic behaviors in developer interactions is a critical issue jeopardizing the sustainability of OSS communities.
S097	However, newcomers face several barriers in the OSS context [...]. These barriers differently affect underrepresented populations [...]. The consequences of these challenges [...] add to the diversity imbalance in OSS.
S098	In the context of OSS projects, sustainability refers to the ability of a project to maintain its longevity, relevance, and effectiveness over time. [...] the article by Yin et al. found that the number of active developers positively associates with the sustainability outcome [...].
S099	A key factor for the sustainability of OSS projects is to attract long-term contributors. Zhang et al. found a positive association between the diversity of contribution models and the number of volunteers and a negative impact of company domination on the sustainability of OSS projects. Valiev et al., however, found that the involvement of companies has a significant effect on the sustainability of projects in the PyPI ecosystem. This indicates that, although women usually have a low percentage in many OSS projects, promoting gender diversity is worthwhile in OSS development (or at least the Rust community). We find that core paid developers tend to contribute more frequently; commits contributed by one-time paid developers have bigger sizes; peripheral paid developers implement more features; and being paid plays a positive role in becoming a long-term contributor. Companies should become more sensitive to how they engage with OSS communities, in certain ways as suggested by this study.
S100	We conclude that an environment that is open for communication and multidisciplinary collaboration facilitates the opportunities of serendipitous support and ultimately increases the sustainability of FOSS projects.

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
S101	Empirical evidence suggests that more atypical projects tend to draw more attention from users, possibly due to a competitive advantage relative to other packages available to the open-source community. However, more atypical projects tend to be developed by a smaller set of contributors, which could lead to problems in their sustainability. We found that innovative projects tend to draw more attention, primarily in the form of GitHub stars, and more contributors, especially in the long term. At the same time, we found that innovative projects require more effort from developers on average and face greater challenges with sustaining their activity.
S102	Mentoring goes beyond assisting contributors in overcoming technical and systemic challenges; it also serves as an essential instrument in fostering a sustainable, diverse, and inclusive collaboration environment. Mentors help with technical challenges by guiding mentees and fostering a culture of knowledge-sharing and collaboration. Ultimately, today’s mentees evolve into tomorrow’s mentors [2,9]. Shifting to diversity, mentoring can be strategically designed to pair individuals based on geographical or cultural demographics. This approach directly counteracts biases and fosters a welcoming, inclusive OSS community. This can simultaneously help community growth and sustainability, attracting and retaining contributors for OSS projects’ long-term success.
S103	[...] we consider a DL package as “sustainable” if it has sustained activity in its last 12 months prior to its most recent commit.
S104	Amount of perceived work for maintainers generated from the OSS community (e.g., support requests, bug fixes, community management) need to balance against maintainers’ capacity. Creating a work-life balance requires personal prioritization by the maintainer over their perceived or experienced need to maintain their projects. Toxicity is commonly experienced and requires a proactive approach and building of a positive culture enforced, e.g., through a code of conduct. Episodic contributors should be encouraged and empowered to make their contributions as high quality as possible. A Sense of inclusiveness is considered pivotal for new contributors to stay around. Highlighted means include encouragement and appreciation for contributions, openness in decision-making, and responsiveness in answers. Awareness and actions are needed to address the potential negative impact of a project’s characteristics (e.g., programming language, position in the stack, and general stability) on the attractiveness of new contributors. Maintainers need help with marketing and outreach, as they may feel uncomfortable and prefer to focus on technical aspects of their projects. Sharing knowledge in communication and documentation decreases dependency on the single maintainer while improving onboarding and inclusiveness. Contribution barriers need to be removed while support measures help increase the quality of contributions without consuming too much of the maintainers’ available time. Full-time employment dedicated to working on projects is seen as a dream by many [...].

Table 9 – continued

ID	Relevant fragment(s); extracted keywords in bold
	Being able to spend part-time employment to maintain a project is viewed favorably [...]. Building a business comes with many variants and is considered risky to different extents. Prioritizing the needs of paying customers and the community requires a delicate balance. Personal sponsorship is considered a symbol of gratitude, while corporate sponsorship is more significant. None is, however, considered a sustainable source of income.
S105	A well-functioning voting system is indicative of a project’s potential for success and sustainability
S106	This raises the question of whether unpaid open-source developers feel more motivated by their ability to put values into action than paid ones. Have fun Community Spread knowledge Skill development Values into actions Duty to users Personal use Organization’s reputation Collaboration Commercial use Improve existing projects
S107	However, much less is known about how factors external to the project, related to its position and role in the overall open-source ecosystem, impact the process of attracting and retaining new contributors. The prevailing empirical studies on open-source sustainability, and attracting new contributors in particular, have focused on the influence of project-level characteristics. In our paper, we provide an alternative, ecosystem-level perspective by suggesting the project’s labor pool as an important factor. While open-source developers and researchers have devoted much effort to making individual projects more successful and sustainable, little was discussed about the influence of those efforts on other projects in the ecosystem.
S108	[...] maintaining up-todate documentation alongside code changes is critical for project sustainability.
S109	[...] the open-source software community has faced various onboarding challenges over the past decades which has posed a threat to its sustainability
S110	In a survey of 49 developers from the NPM ecosystem, we find that developers are more likely to maintain their own packages rather than contribute to the ecosystem. For researchers, this study calls for more research into how the different relationships between developers of a package may assist it with its sustainability. First, we observe that personal drivers are the key factors behind developers making contributions to their own packages (40 responses). They also reserve more challenging tasks for own packages, as oppose to trying to make to other packages. Second, we find that developers contribute to other packages more due to professional (39 responses) rather than personal drivers.
S111	[...] motivate, engage, and retain new developers is the way to promote a sustainable amount of developers in a project. [...] presents five categories: Social Interactions, Finding a Way to Start, Documentation Problems, Code Issues, and Newcomers’ Knowledge.

Detailed Sustainability Profiles

The tables in this appendix give the full per-aspect rating, confidence level, and evidence summary for each of the 14 tools evaluated in Chapter 4. They support the consolidated heatmap presented in that chapter's Results (Table 8). N/A indicates that public evidence was insufficient to assign a rating; confidence is therefore not applicable. The evidence summaries condense the public sources and collection procedure described in the chapter's Research Method (Section 4.3.4).

Table 10 – Sustainability profile: Prometheus

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	~50-member governance team, long-tenured maintainers, steady weekly commits over 13 years
Contributor Attraction	2	High	1,400+ lifetime contributors; low-hanging-fruit labels, Gitpod, GSoC/LFX mentorship
Internal Communication	2	High	Public dev/user mailing lists, CNCF Slack, IRC/Matrix, Discourse, dev summits, PromCon
Governance Structure	2	High	Published governance (team/maintainer roles, supermajority vote); CNCF-graduated; CNCF CoC
Openness/Onboarding	2	High	CONTRIBUTING guide, DCO, developer docs, one-click Gitpod, recurring mentorship
Legitimacy	2	High	CNCF's 2nd graduated project; de facto metrics standard
External Communication	2	High	Exposition format/OpenMetrics standard, remote read/write, OTLP, large exporter ecosystem
<i>Technical</i>			
Sustained Activity	2	High	Six-week release cadence; v3.13 (2026-07); commits merged daily
Functionality/Scope	2	High	Actively evolving (native histograms, OTLP, v3); documented stability policy
Quality Assurance	2	High	~270 test files; race/fuzzing/CodeQL/govulncheck CI; OpenSSF Scorecard 8.1; pre-release benchmark soak
Methodologies/Practices	2	High	SemVer, release shepherds, LTS line, release branches, structured changelog, code review
Architecture/Maint.	2	High	Modular components, pluggable service discovery, CODEOWNERS, documented internal architecture
Licensing	2	High	Apache-2.0 from the first licence-bearing public commit in 2012 through the current history; DCO rather than a vendor CLA; Prometheus mark held by the Linux Foundation (Prometheus, 2012 ; Prometheus, 2026d ; Prometheus, 2026b ; The Linux Foundation, 2026)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors' motivations are not observable from public data
Personal Knowledge	2	High	~50 team members, per-subsystem maintainers, design/architecture docs, multi-person release rule
<i>Economic</i>			
Financial Resources	2	Medium	Diversified CNCF backing and Prometheus's strategic importance give its mature governance meaningful influence over resource priorities
External Value Capture	2	High	Managed offerings (Grafana Cloud, AWS, Google, Azure), Thanos/Cortex/Mimir vendors, and training generate substantial value
Reinvestment	2	Medium	CNCF support and employer-backed maintainers from Grafana Labs, Red Hat, Google, and others sustain several project needs
<i>Environmental</i>			
Environmental Req.	1	Medium	Resource efficiency is a core concern (TSDB, native histograms, perf work); no explicit environmental framing

Table 11 – Sustainability profile: OpenTelemetry

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	Second-largest CNCF project; large distributed base, elected leadership with staggered terms and an emeritus path
Contributor Attraction	2	High	Among the highest contributor inflows in the CNCF; membership ladder, many SIGs, good first issue
Internal Communication	2	High	Public SIG meetings, CNCF Slack, spec process, published GC/TC charters, minutes, and elections
Governance Structure	2	High	Elected Governance + Technical Committees, SIGs, CoC; CNCF-neutral; leadership spans direct competitors
Openness/Onboarding	2	High	Member/triager/approver/maintainer ladder, CONTRIBUTING, SIG onboarding; scope breadth is the only barrier
Legitimacy	2	High	CNCF-graduated (2026), “de facto observability standard”; backed by every major vendor
External Communication	2	High	OTLP is the common ingestion protocol; Collector translates dozens of formats; ~12 language SDKs
<i>Technical</i>			
Sustained Activity	2	High	Second-most-active CNCF project; continuous commits across dozens of repositories
Functionality/Scope	2	Medium	Traces, metrics, logs, profiling across ~12 languages, definitive scope, but breadth outruns completion (logs/profiling maturing; Collector 0.x)
Quality Assurance	2	High	Extensive tests (Collector ~807/1,740 Go files), spec-compliance testing, CI, documented security process
Methodologies/Practices	2	High	OTEP (enhancement-proposal) process, spec versioning, SemVer, SIG structure, documented project management
Architecture/Maint.	2	Medium	Clean API/SDK/OTLP/Collector layering with pluggable components; tempered by multi-repo, multi-language consistency cost
Licensing	2	High	Apache-2.0 from the first 2019 specification commit through the current project; CNCF/Linux Foundation CLA and trademark governance reduce single-vendor relicensing risk (OpenTelemetry, 2019 ; OpenTelemetry, 2026b ; OpenTelemetry, 2026a ; The Linux Foundation, 2026)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	2	Medium	Spec and OTEPs preserve rationale; many maintainers and elected succession, but per-language SIG depth is uneven
<i>Economic</i>			
Financial Resources	2	Medium	Diversified CNCF backing provides durable resources, while OpenTelemetry governance controls project requests and allocation priorities
External Value Capture	2	High	The observability vendor industry generates substantial value from OTLP ingestion, distributions, and managed offerings
Reinvestment	2	High	CNCF support and broad employer-funded maintainership across competing companies cover several project needs
<i>Environmental</i>			
Environmental Req.	1	Medium	Real efficiency concern (Collector performance, sampling to cut data volume); no explicit environmental framing

Table 12 – Sustainability profile: Apache Kafka

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	1,700+ lifetime contributors, 100+ committers and a large PMC, long-tenured core, 15 years active
Contributor Attraction	2	High	13,500+ merged PRs; KIP process and JIRA newbie labels; steady inflow despite a heavy codebase
Internal Communication	2	High	Very active dev@ list (400–800 msgs/month in 2026), archived KIP wiki and JIRA, ASF Slack, Kafka Summit
Governance Structure	2	High	ASF top-level project: meritocratic committer/PMC model, the Apache Way, chair reports to the ASF board
Openness/Onboarding	2	Medium	Documented contribution + KIP process, committer tooling; but JVM/Scala/Gradle and KIP overhead raise the barrier
Legitimacy	2	High	De-facto event-streaming standard; used by thousands of companies over 15 years
External Communication	2	High	Connect connector ecosystem, multi-language clients, protocol reimplemented by Redpanda/WarpStream and others
<i>Technical</i>			
Sustained Activity	2	High	Roughly quarterly releases (4.0 in 2025 to 4.3 by mid-2026); commits merged daily (Apache Software Foundation, 2025 ; Apache Software Foundation, 2026b)
Functionality/Scope	2	High	Actively evolving (KRaft/ZooKeeper removal, tiered storage); KIP-driven roadmap
Quality Assurance	2	Medium	2,235 test files, extensive CI, security model docs, ASF CVE process; but OpenSSF Scorecard 6.1 (Open Source Security Foundation, 2026a) and a known flaky-test burden
Methodologies/Practices	2	High	KIP design process (RFC-like, voted), SemVer, time-based release plans, mandatory committer review
Architecture/Maint.	2	High	~20 modules (clients, core, streams, connect, raft, storage); KRaft rewrite shows evolvability
Licensing	2	High	Apache-2.0 from the initial 2011 ASF import through the current Kafka, Connect, and Streams code; ASF contributor agreements and trademark governance, with no substantive core relicensing in that history (Apache Software Foundation, 2011 ; Apache Software Foundation, 2026b ; Apache Software Foundation, 2026c ; Apache Software Foundation, 2026a)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors' motivations are not observable from public data
Personal Knowledge	2	High	100+ committers, KIPs preserve design rationale, extensive docs; but activity skews to Confluent employees
<i>Economic</i>			
Financial Resources	1	Medium	ASF provides real resources, but evidence of meaningful project-level allocation control is limited
External Value Capture	2	High	Confluent Cloud/Platform, Amazon MSK, Aiven, Cloudera, Instaclustr, IBM Event Streams, and training generate substantial value
Reinvestment	1	Medium	ASF support and employer-backed committers sustain the project, but this work is concentrated around Confluent
<i>Environmental</i>			
Environmental Req.	1	Medium	Efficiency/cost are real concerns (tiered storage, KRaft); no explicit environmental framing; heavy JVM/replication footprint

Table 13 – Sustainability profile: Jaeger

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	Medium	Long-tenured founder still active; graceful emeritus process; steady maintainer continuity
Contributor Attraction	2	Medium	~490 lifetime contributors, 4,300+ merged PRs, CNCF mentorship; but a narrow niche
Internal Communication	2	High	Public mailing list, CNCF Slack, GitHub, blog, KubeCon maintainer track
Governance Structure	2	High	Formal GOVERNANCE.md (2/3 votes, emeritus, 2FA rules); CNCF-graduated; six maintainers across six employers
Openness/Onboarding	2	Medium	Documented contribution + maintainer path, DCO, CNCF mentorship; small reviewer pool
Legitimacy	2	Medium	CNCF's 7th graduated project; named production adopters (Uber, Red Hat, Ticketmaster); standing pressured by OTel consolidation
External Communication	2	High	Deepest standards alignment in the study: OpenTracing then OpenTelemetry, OTLP-native, many storage backends
<i>Technical</i>			
Sustained Activity	2	High	Roughly monthly releases (v2.19 in mid-2026); commits merged daily
Functionality/Scope	2	Medium	Actively evolved (v2 on the OTel Collector), but scope narrowed as instrumentation was ceded to OTel
Quality Assurance	2	High	~1:1 test-to-source ratio, e2e CI across 6+ storage backends, CodeQL, threat model; OpenSSF Scorecard 8.2
Methodologies/Practices	2	High	SemVer, rotating release manager, explicit deprecation-grace policy, DCO, code review
Architecture/Maint.	2	High	Modular, pluggable storage, component-based; the v2 rebuild on the OTel Collector shows evolvability
Licensing	2	High	Publicly changed from MIT to Apache-2.0 in 2017 and unchanged thereafter; DCO rather than a vendor CLA; Jaeger mark held by the Linux Foundation (Jaeger, 2017 ; Jaeger, 2026 ; The Linux Foundation, 2026)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors' motivations are not observable from public data
Personal Knowledge	1	Medium	Good docs and a founder-authored book, knowledge spread across employers, but only six active maintainers
<i>Economic</i>			
Financial Resources	1	Medium	CNCF provides real resources, but evidence of meaningful project-level allocation control is limited
External Value Capture	1	Medium	Hosting and support exist, but the ecosystem is modest and value is shifting to OTel-native backends such as Tempo
Reinvestment	2	Medium	CNCF support and employer-backed development across six employers provide recurring support for several project needs
<i>Environmental</i>			
Environmental Req.	1	Medium	Sampling and storage efficiency are core concerns (trace volume/cost); no explicit environmental framing

Table 14 – Sustainability profile: Kepler

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	1	Medium	Formal 6-month active/emeritus review cycle, but a major 2025 turnover: 12 maintainers to emeritus, incl. the original creator and the entire Intel contingent
Contributor Attraction	1	Medium	good first issue/help wanted labels and CNCF visibility, but a niche domain, ~1.5k stars, and an explicit “community call to action” signalling too few contributors
Internal Communication	2	Medium	CNCF Slack, regular community meetings, GitHub Discussions, public enhancement proposals, and a candid re-architecture writeup
Governance Structure	2	High	Formal GOVERNANCE.md (roles, election by 3-maintainer approval, Advisory Committee, CoC); CNCF-neutral; maintainer list synced to the CNCF registry
Openness/Onboarding	2	Medium	CONTRIBUTING guide, DCO, pre-commit, design docs; but a steep kernel/power-metrics domain barrier and an unpopulated reviewer (OWNERS) tier
Legitimacy	1	Medium	CNCF Sandbox and peer-reviewed academic attention, but limited adoption and a young, pre-1.0 project that has not progressed beyond Sandbox
External Communication	2	Medium	Prometheus exporter, Helm/OCI, operator, Grafana dashboards: cooperates cleanly with the surrounding ecosystem
<i>Technical</i>			
Sustained Activity	2	High	Daily commits, frequent releases (0.10.0 rewrite through 0.11.4 by mid-2026)
Functionality/Scope	1	Medium	Relevant role, but Kepler 0.7.2 produced unreliable container attribution in a single-server synthetic evaluation despite less than 1% total-energy error (PIJACKER <i>et al.</i> , 2025); the implementation was later rewritten and the VM case remains experimental
Quality Assurance	2	Medium	79/146 Go test files, CI, Codecov, OpenSSF Best Practices + Scorecard badges, e2e tests, and empirical accuracy validation against IPMI; tempered by pre-1.0 status
Methodologies/Practices	2	High	SemVer, enhancement-proposal (RFC-like) process, release-management role, commitlint, pre-commit, mandatory review
Architecture/Maint.	2	Medium	Rewrite gave a documented, modular, service-oriented design, though the need to rewrite is itself evidence the prior architecture was unmaintainable
Licensing	2	High	Apache-2.0 in the earliest public release (v0.1, 2022) and current tree, with no substantive project relicensing found; DCO and Linux Foundation trademark governance (Kepler, 2022; Kepler, 2026; The Linux Foundation, 2026)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	1	Medium	Design docs and EPs aid transfer, but the bus factor is thin (3 Red Hat engineers wrote ~84% of the rewrite) and the founding generation that held the prior domain knowledge just left
<i>Economic</i>			
Financial Resources	1	Medium	CNCF hosting provides real resources, but day-to-day governance remains with Kepler’s maintainers and project-level allocation control is limited
External Value Capture	1	Medium	Limited value is generated mainly through Red Hat/OpenShift and Grafana rather than a broad market
Reinvestment	1	Medium	CNCF support and employer-backed development sustain the project, but current work is concentrated among a few Red Hat contributors (MANOLEDAKI; CHOOCHOTKAEW, 2026)
<i>Environmental</i>			
Environmental Req.	2	High	The project exists to measure energy for sustainability: explicit green-software framing (CNCF TAG Environmental Sustainability, datacenter/AI energy motivation) throughout

Table 15 – Sustainability profile: Elasticsearch

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	16 years active; long-tenured core of Elastic engineers, steady daily commits; retention strong but employer-bound
Contributor Attraction	1	Medium	good first issue/help wanted labels and a Contributor Program, but CLA, all-Elastic code ownership, and “we generally do not assign issues to contributors outside of Elastic” cap external inflow
Internal Communication	2	Medium	Active GitHub issues/PRs and the discuss.elastic.co forum; but no open RFC/design process, strategic decisions (2021 relicensing) were announced, not deliberated in public
Governance Structure	1	Medium	Single-vendor: no foundation or external steering, CODEOWNERS entirely @elastic teams, CLA concentrates distribution rights in Elastic B.V.; credible and stable, but unilateral control was exercised in 2021 and split the ecosystem
Openness/Onboarding	1	Medium	Extensive docs (50 KB CONTRIBUTING, BUILDING, TESTING), issue/PR templates; but CLA sign-off, a JVM/Gradle megaproject, and Elastic-internal issue assignment raise the barrier
Legitimacy	2	High	16 years; ubiquitous ELK/Elastic Stack; broad enterprise adoption
External Communication	2	High	Beats/Logstash/Kibana, official clients, OTel/APM ingest; but the 2021 relicense forked its own API (OpenSearch) rather than converging it
<i>Technical</i>			
Sustained Activity	2	High	9.0 GA April 2025, 9.4.3 by mid-2026; commits merged daily through bot-assisted release automation (ELASTIC, 2026c)
Functionality/Scope	2	High	Actively evolving well beyond search, vector/semantic search, ES/QL, observability, security
Quality Assurance	2	Medium	~11,500 test files, extensive Buildkite CI; but 253 currently-muted flaky tests, a dedicated test-failure template, and no in-repo security policy
Methodologies/Practices	2	High	SemVer, automated backport and changelog tooling, mandatory review, license-header pre-commit gates; design process internal rather than public RFC
Architecture/Maint.	2	High	Modular (server/modules/plugins/libs/x-pack), Lucene-based, CODEOWNERS, heavy build tooling
Licensing	0	High	Elastic exercised single-vendor control to remove the OSS option in 2021; AGPLv3 was restored in 2024, but the history remains unstable and x-pack remains ELv2-only (BANON, 2021 ; BANON, 2024 ; ELASTIC, 2026a)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	2	Medium	Large team, extensive developer docs, CODEOWNERS; but knowledge concentrated in Elastic teams
<i>Economic</i>			
Financial Resources	2	Medium	Elastic’s broad recurring revenue and Elasticsearch’s flagship status provide durable capacity under direct company allocation
External Value Capture	2	High	Elastic Cloud, Amazon OpenSearch Service, Bonsai, Instaclustr, and a large training and consulting market generate substantial value
Reinvestment	2	Medium	Elastic substantially and recurrently funds development, infrastructure, security, documentation, and project operations
<i>Environmental</i>			
Environmental Req.	1	Medium	Efficiency work (Better Binary Quantization, storage tiering, compression); no explicit environmental/green framing; heavy JVM/Lucene footprint

Table 16 – Sustainability profile: Kibana

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	13 years active; long-tenured Elastic engineering core, steady daily commits; retention strong but employer-bound
Contributor Attraction	1	Medium	good first issue labels exist, but a CLA, all-@elastic code ownership, and a very large TS/JS monorepo cap external inflow
Internal Communication	2	Medium	Active GitHub issues/PRs and the discuss.elastic.co forum; no public RFC process, the 2021 relicensing was announced, not deliberated
Governance Structure	1	Medium	Single-vendor: no foundation, CODEOWNERS is 72 @elastic teams with no external owners, CLA concentrates distribution rights in Elastic; unilateral relicensing exercised in 2021
Openness/Onboarding	1	Medium	Developer guide and issue templates, but a CONTRIBUTING that splits “employee at Elastic” from “external developer”, CLA sign-off, and a huge monorepo raise the barrier
Legitimacy	2	High	Ubiquitous as the Elastic Stack frontend; 13 years; ~21k stars, though its legitimacy is largely derivative of Elasticsearch’s
External Communication	2	Medium	Plugin system, official Stack integration; but the 2021 relicense forked its role into OpenSearch Dashboards rather than converging it
<i>Technical</i>			
Sustained Activity	2	High	9.x line current, v9.4.3 by mid-2026; commits merged daily
Functionality/Scope	2	High	Actively evolving: Lens, observability and security apps, alerting, and an AI assistant; clearly still fills and grows its role
Quality Assurance	2	Medium	Large test suite and extensive CI, and (unlike Elasticsearch) an in-repo SECURITY.md; but a very large open-issue backlog (~14.8k)
Methodologies/Practices	2	High	SemVer aligned with the Stack, mandatory review, backport tooling, license-header gates
Architecture/Maint.	2	Medium	Modular plugin architecture and CODEOWNERS, but a large TS/JS monorepo with heavy build tooling
Licensing	0	High	Elastic exercised single-vendor control to remove Kibana’s OSS option in 2021; AGPLv3 was restored in 2024, but the history remains unstable and x-pack remains ELv2-only (BANON, 2021; BANON, 2024; ELASTIC, 2026a)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	2	Medium	Large team and thorough developer docs; but knowledge concentrated in Elastic teams
<i>Economic</i>			
Financial Resources	2	Medium	Elastic’s broad recurring revenue and Kibana’s core role in its flagship Stack provide durable capacity under direct company allocation
External Value Capture	2	High	Elastic Cloud, OpenSearch Dashboards, training, and consulting generate substantial value around the project lineage
Reinvestment	2	Medium	Elastic substantially and recurrently funds development, infrastructure, security, documentation, and project operations
<i>Environmental</i>			
Environmental Req.	1	Medium	Some client/query efficiency work; no explicit environmental/green framing

Table 17 – Sustainability profile: Grafana

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	12 years, long-tenured core incl. the creator, ~39-member team; retention strong but overwhelmingly Grafana Labs
Contributor Attraction	2	Medium	Grafana Champions program, good first issue, a real external contributor tail; tempered by CLA and vendor dominance
Internal Communication	2	High	Public grafana-developers/grafana-team mailing lists, documented [VOTE] process, community forum, Slack
Governance Structure	1	Medium	Elaborate GOVERNANCE.md (consensus, votes, some external team members) but “governance changes are made by Grafana Labs”, ~90% of the team and all CODEOWNERS are Grafana Labs, CLA concentrates rights
Openness/Onboarding	2	Medium	Detailed CONTRIBUTING and contribute/ guides, Champions program; tempered by CLA and a large Go+TypeScript monorepo
Legitimacy	2	High	De-facto dashboarding standard: highest pulls of any candidate (5.26B), 75k stars, 2nd most studied (15)
External Communication	2	High	Data-source-agnostic plugin ecosystem; cooperative with AWS/Azure managed offerings; Apache-2.0 SDK libraries
<i>Technical</i>			
Sustained Activity	2	High	~6,000 commits in H1 2026; v12.0 GA in 2025; commits merged continuously
Functionality/Scope	2	High	Actively evolving (Drilldown, Scenes, alerting, cloud migration); broad and central visualisation role
Quality Assurance	2	High	~1,919 Go + ~2,450 TS/TSX test files, extensive CI, Playwright e2e, ESLint, signed commits (2026); security handled off-repo
Methodologies/Practices	2	High	SemVer, mailing-list RFC/vote process, detailed changelog, mandatory review, lefthook hooks
Architecture/Maint.	2	High	Modular monorepo (packages/pkg/apps), plugin SDK, data-source architecture, squad-based CODEOWNERS
Licensing	1	High	Unilateral Apache-2.0 to AGPLv3 change (2021) from a single-vendor position under a CLA; the core nevertheless remains entirely under an OSI/FSF-approved licence, SDK libraries remain Apache-2.0, and no proprietary code is in-tree (Grafana Labs, 2021 ; Grafana Labs, 2026c)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition mechanisms are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	2	Medium	~39-member team, squad ownership, extensive docs, multi-maintainer; but knowledge concentrated in Grafana Labs
<i>Economic</i>			
Financial Resources	2	Medium	Grafana Labs’ recurring revenue and Grafana’s flagship status provide durable capacity under direct company allocation
External Value Capture	2	High	Grafana Cloud, Amazon and Azure managed offerings, and a large plugin and consulting market generate substantial value
Reinvestment	2	Medium	Grafana Labs substantially and recurrently funds development, infrastructure, security, documentation, and community operations
<i>Environmental</i>			
Environmental Req.	1	Medium	Real efficiency work (Scenes rendering, query caching) but no environmental/green framing

Table 18 – Sustainability profile: InfluxDB

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	2	High	12 years, long-tenured InfluxData core (incl. the founder); retention strong but employer-bound
Contributor Attraction	1	Medium	CLA, single-vendor development, a Rust rewrite, and migration churn cap external inflow
Internal Communication	2	Medium	Public GitHub, community forum, Discord, blog; but no open governance, and pivotal decisions are announced, sometimes to pushback
Governance Structure	1	Medium	Single-vendor (InfluxData); CLA; no foundation or external steering
Openness/Onboarding	1	Medium	CONTRIBUTING and a buildable tree, but CLA and a “talk to us before significant changes” gate
Legitimacy	2	High	12 years, leading dedicated TSDB, TICK stack, >1B pulls
External Communication	2	High	Line protocol + Telegraf ecosystem, SQL/InfluxQL/Flight SQL, Parquet/Arrow, v1/v2 write compatibility
<i>Technical</i>			
Sustained Activity	2	High	~46k commits since 2013; v3 line active (~v3.7 by mid-2026), commits daily
Functionality/Scope	2	Medium	Capable, modern, actively evolving (Arrow/DataFusion), but the open edition is capability-gated and generations churn
Quality Assurance	2	High	~427 of ~972 Rust files carry tests; CircleCI + GH Actions; SECURITY.md; supply-chain deny.toml
Methodologies/Practices	2	Medium	SemVer, review, Rust tooling; tempered by repeated breaking migrations (v1 to v2 to v3, Go to Rust, Flux to SQL)
Architecture/Maint.	2	High	Modern modular Rust engine (diskless, object storage, Parquet), many crates; a clean re-architecture
Licensing	1	High	MIT throughout v1/v2 and MIT/Apache-2.0 in v3 remain OSI/FSF-approved, but InfluxData’s CLA grants sublicensing and future relicensing authority, while one vendor controls the proprietary feature boundary (INFLUXDATA, 2013; INFLUXDATA, 2026b; INFLUXDATA, 2018; INFLUXDATA, 2026c)
<i>Individual</i>			
Personal Motivations	N/A	–	Employment is visible, but contributors’ motivations are not observable from public data
Personal Knowledge	2	Medium	Large long-tenured team, modular crates, docs; but concentrated in one company
<i>Economic</i>			
Financial Resources	1	High	InfluxData provides real capacity, but allocation remains concentrated in and controlled by one vendor
External Value Capture	1	High	Cloud, Enterprise, marketplace, and consulting activity exists, but monetization is dominated by InfluxData
Reinvestment	2	High	InfluxData substantially and recurrently funds development, infrastructure, quality assurance, documentation, and project operations, including the v3 rewrite
<i>Environmental</i>			
Environmental Req.	1	Medium	Real efficiency work (diskless, Parquet, Rust performance); no environmental/green framing

Table 19 – Sustainability profile: Nagios

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	1	Medium	Small stable maintenance team, but the project has repeatedly shed contributor communities to forks
Contributor Attraction	1	Medium	Some external bug-fix contributions, but changes are “at the discretion of the development team”; energy went to forks
Internal Communication	1	Medium	Company-run support forums and GitHub issues; no public roadmap or design process; decision-making opaque
Governance Structure	0	High	Single-vendor control (Nagios Enterprises); the 2009 Icinga fork was an explicit governance failure
Openness/Onboarding	1	Medium	CONTRIBUTING guide and PRs accepted for fixes, but features gated by the vendor; questions redirected to forums
Legitimacy	1	Medium	Historically foundational and heavily studied (10 studies), but eroding, overtaken by cloud-native tools and its own forks
External Communication	1	Medium	Plugin API is a de facto standard across the family, but the project alienated the plugin community (2014)
<i>Technical</i>			
Sustained Activity	1	High	Regular CVE and bug-fix releases, but maintenance-only: no major version since the 4.x line introduced in 2013 (Nagios Enterprises, 2026b)
Functionality/Scope	1	Medium	Still fills the classic check role; not evolved for cloud-native/dynamic environments, which drove users elsewhere
Quality Assurance	1	Medium	Has tests and CI, responds to CVEs; but OpenSSF Scorecard 3.5 (Open Source Security Foundation, 2026c) and a steady stream of C/CGI security fixes
Methodologies/ Practices	1	Medium	Changelog, SemVer-style releases, PR review; no design/RFC process, no public roadmap
Architecture/Maint.	1	Medium	Elegant, extensible plugin model; but a dated single-node C/CGI core that forks rewrote to modernise
Licensing	2	Medium	GPLv2 from the initial 2002 public import through the current core, with no substantive relicensing or public CLA requirement found; company-held marks and the separate Nagios XI product do not alter Core’s licence (Nagios Core Development Team, 2002 ; Nagios Enterprises, 2026b ; Nagios Core Development Team, 2026 ; Nagios Enterprises, 2026a)
<i>Individual</i>			
Personal Motivations	N/A	–	Recognition and contributor exits are visible, but contributors’ motivations cannot be established from public data
Personal Knowledge	1	Medium	Docs and a stable codebase, but knowledge concentrated in a small vendor team; historic single-maintainer bottleneck
<i>Economic</i>			
Financial Resources	1	Medium	Company-led maintenance provides capacity, but project-level funding and allocation are not disclosed and remain controlled by one vendor
External Value Capture	1	Medium	Commercial activity exists but is dominated by the trademark-holding vendor and constrains third parties
Reinvestment	1	Medium	Company-led maintenance supports the core, but its extent is not disclosed and support remains concentrated and company-directed
<i>Environmental</i>			
Environmental Req.	N/A	–	No explicit environmental or resource concern was found; lightweight C alone does not support a rating

Table 20 – Sustainability profile: Net-SNMP

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	1	Medium	Exceptional core continuity (founder active 30 yrs; a steady lead committer), but a very thin, highly concentrated base
Contributor Attraction	1	Medium	A long tail of occasional contributors and 300+ forks, but no onboarding and an old C codebase
Internal Communication	1	Medium	Long-standing public mailing lists, a wiki, and active GitHub issue handling, but thin participation
Governance Structure	1	Medium	No governance doc, but a functioning maintainer-led process (named release manager, documented schedule, version branches), informal, not absent or captured
Openness/Onboarding	1	Medium	Extensive wiki/docs and a buildable tree, but no CONTRIBUTING and a large, old C codebase
Legitimacy	2	High	The de facto reference SNMP implementation; 30+ yrs; ubiquitous in distros and network devices
External Communication	2	High	Implements the IETF SNMP standards; AgentX/embedded extensibility; the substrate other tools consume
<i>Technical</i>			
Sustained Activity	2	Medium	~300 commits/yr every year 2021–2026 and active branches, but stable releases are infrequent (5.9.4 in 2023; 5.10 still pre-release)
Functionality/Scope	2	Medium	Complete, current implementation of a mature protocol; still fully fills its ecosystem role
Quality Assurance	1	Medium	A test suite and CI exist and the lead is a rigorous maintainer, but no in-repo security policy; OpenSSF Scorecard 5.0 (Open Source Security Foundation, 2026d)
Methodologies/Practices	1	Medium	Versioned releases, release manager, patch branches, ChangeLog, established but old-school and thinly staffed
Architecture/Maint.	1	Medium	Modular, portable, extensible C (library/agent/tools, AgentX, MIB modules), but a large aging codebase
Licensing	2	High	Multi-party BSD-style terms from the earliest inspectable 1996 history through the current tree; later changes add permissive notices, with no restrictive relicensing, CLA, or single corporate rights holder visible (UCD-SNMP Development Team, 1996 ; Net-SNMP Development Team, 2026b)
<i>Individual</i>			
Personal Motivations	N/A	–	Contributors' motivations are not observable from public data
Personal Knowledge	0	Medium	Heavy dependence on 1–2 individuals for a large, critical C codebase; docs/wiki help, but the bus-factor risk is real and realised
<i>Economic</i>			
Financial Resources	0	Medium	No project-controlled funding, foundation, company, or visible sponsorship; maintenance depends on volunteers
External Value Capture	2	Medium	Commercial and community distributions such as Red Hat Enterprise Linux have long embedded Net-SNMP, so downstream value capture is substantial (Net-SNMP Development Team, 2026a ; Red Hat, 2026)
Reinvestment	0	Medium	Little visible support returns from downstream beneficiaries; no public evidence of paid maintainers or funded stewardship was identified
<i>Environmental</i>			
Environmental Req.	N/A	–	No explicit environmental or resource concern was found; an inferred low footprint for embedded use does not support a rating

Table 21 – Sustainability profile: Ganglia

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	0	Medium	Core dev team dispersed; <code>monitor-core</code> no commits since 2021; only sporadic web patches remain
Contributor Attraction	0	Medium	No good-first-issue labels, no onboarding, a dormant C core; only incidental drive-by PHP fixes
Internal Communication	1	Medium	SourceForge mailing list and GitHub issues exist (~86/91 open), but discussion is largely quiet/stale
Governance Structure	0	Medium	No governance doc, no MAINTAINERS/CODEOWNERS; informal “Ganglia Development Team” mailing list; no decision process
Openness/Onboarding	1	Medium	Code is open (BSD) and buildable, with older man/pod docs and a wiki, but a stub README, dead CI, and no CONTRIBUTING
Legitimacy	1	Medium	Genuine historical/academic legitimacy (25+ yrs, 500+ clusters, widely cited paper), but present-day importance is fading
External Communication	1	Medium	Open formats (XML/XDR, RRDtool), a Nagios bridge, third-party Prometheus exporters; but no forward integration (no OTel/native Prom)
<i>Technical</i>			
Sustained Activity	0	Medium	<code>monitor-core</code> has had no commits since 2021; <code>ganglia-web</code> receives only sporadic compatibility/security patches, with no core release in about a decade (Ganglia Development Team, 2026b; Ganglia Development Team, 2026a)
Functionality/Scope	1	Medium	Still works for its HPC-cluster niche, but scope is frozen and displaced for cloud-native use
Quality Assurance	0	Medium	CI badge points to the defunct travis-ci.org; no security policy; aged C/PHP practices; OpenSSF Scorecard 3.4 (Open Source Security Foundation, 2026b)
Methodologies/Practices	0	Medium	Historical release and change-management artefacts remain, but the public process is now lapsed
Architecture/Maint.	1	Medium	Sound, modular, low-overhead design (gmond/gmetad/gweb, multicast, RRDtool), but aging tech and no active upkeep
Licensing	2	High	Permissive from the initial 2002 history; clarified to BSD-3-Clause in 2010 with no later substantive licence change, CLA, or current corporate controller visible (Ganglia Development Team, 2010; Ganglia Development Team, 2026b)
<i>Individual</i>			
Personal Motivations	N/A	–	Contributors’ motivations are not observable from public data
Personal Knowledge	1	Medium	Knowledge is externalised in docs and the design paper, but no active maintainers hold the C core, a realised bus-factor
<i>Economic</i>			
Financial Resources	0	Medium	No current project-controlled funding was identified; historical grant origins do not provide present capacity (MASSIE <i>et al.</i> , 2004)
External Value Capture	0	Medium	No current vendors, hosted offerings, or meaningful consulting activity was identified in the reviewed sources
Reinvestment	0	Medium	No public evidence of paid maintainers, a current sponsor, or funded stewardship was identified
<i>Environmental</i>			
Environmental Req.	1	Medium	Low per-node overhead is an explicit design value (HPC), but never framed environmentally

Table 22 – Sustainability profile: PowerJoular

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	1	Medium	One principal author sustained five years of work; contributors beyond him are transient, with no community to retain (NOUREDDINE, 2026d)
Contributor Attraction	1	Medium	A few student/collaborator contributions; no onboarding infrastructure; niche language (Ada)
Internal Communication	1	Medium	Public GitHub issues provide a channel, but there is no mailing list, meeting, or design process
Governance Structure	0	Medium	Informal single-maintainer control with no documented structure or succession
Openness/Onboarding	0	Medium	No CONTRIBUTING file or issue templates; the niche implementation language further limits onboarding
Legitimacy	1	Medium	Real academic standing (~130 citations, IEEE IE'22 best-demo award), but limited evidence of adoption beyond academic use
External Communication	1	Medium	Reads RAPL/powercap, nvidia-smi, writes CSV, feeds JoularJX; otherwise ecosystem-isolated
<i>Technical</i>			
Sustained Activity	1	Medium	Maintained 5 years but decelerating (103 commits in 2024, 23 in 2025, 1 by mid-2026); development is expected to stop after Joular Core replaces it (NOUREDDINE, 2025)
Functionality/Scope	2	Medium	Multi-platform scope is documented (RAPL, GPU, VMs, Raspberry Pi models), but the available evidence does not independently validate effectiveness (NOUREDDINE, 2022)
Quality Assurance	0	Medium	Build CI is present, but there are neither tests nor a security policy
Methodologies/Practices	1	Medium	SemVer, releases, Alire packaging; no design or review process (single maintainer)
Architecture/Maint.	2	Medium	Clean modular Ada with reference docs; but Ada and the lone maintainer shrink the pool that can evolve it
Licensing	2	High	GPL-3.0-only from the first public beta (2021), with no substantive licence-text change or CLA/DCO policy found in the public repository history (NOUREDDINE, 2021b; NOUREDDINE, 2026d)
<i>Individual</i>			
Personal Motivations	N/A	–	Academic incentives are visible, but contributors' motivations are not observable from public data
Personal Knowledge	0	Medium	The study's clearest bus-factor case: ~77% one author, no co-maintainer of equal standing, niche language, no tests to encode behaviour; usage docs but no maintenance succession
<i>Economic</i>			
Financial Resources	0	High	The project reports no dedicated funding; limited equipment and activity support in 2020–2021 did not create recurring project-controlled capacity (NOUREDDINE, 2026b)
External Value Capture	0	Medium	No commercial products, hosted offerings, consulting, or paid external participation is visible
Reinvestment	1	Medium	Academic supervision and limited material/activity funding have supported development, but support is narrow and non-recurring (NOUREDDINE, 2026b)
<i>Environmental</i>			
Environmental Req.	2	High	Its entire purpose is measuring energy for green software; explicit framing, down to an energy-efficient implementation language

Table 23 – Sustainability profile: JoularJX

Aspect	Rating	Conf.	Evidence summary
<i>Social</i>			
Contributor Retention	1	Medium	Author-led (~60% of commits) but with several repeat external contributors, better than PowerJoular (NOUREDDINE, 2026c)
Contributor Attraction	1	Medium	Java’s broad pool draws real outside contributors (incl. industry and research); no onboarding infrastructure
Internal Communication	1	Medium	Public GitHub issues provide a channel, but there is no mailing list, meeting, or design process
Governance Structure	0	Medium	Informal single-maintainer control with no documented structure or succession
Openness/Onboarding	1	Medium	No CONTRIBUTING, but a widely-known language and a buildable test suite lower the barrier
Legitimacy	1	Medium	Real academic standing (shared IE’22 paper, Jalen lineage), but limited evidence of adoption beyond academic use
External Communication	1	Medium	JVM agent, multi-OS power sources, Joular Core; otherwise ecosystem-isolated
<i>Technical</i>			
Sustained Activity	2	Medium	Steadier than PowerJoular: 142 commits (2023) through ~17 by mid-2026; Joular Code – Java is its alpha-stage successor, but no end date for JoularJX maintenance is stated (NOUREDDINE, 2026a)
Functionality/Scope	2	Medium	Per-method attribution through a multi-OS JVM agent is documented, but the available evidence does not independently validate attribution effectiveness (NOUREDDINE, 2022)
Quality Assurance	1	Medium	Has a JUnit test suite and Maven CI (unlike PowerJoular), though coverage is light and there is no security policy
Methodologies/Practices	1	Medium	SemVer, Maven, releases, some PR review from the contributor base; no formal design process
Architecture/Maint.	2	Medium	Modular Java (cpu/monitor/result/utls), agent design, power-reading factored into Joular Core; wide language pool
Licensing	2	High	GPL-3.0-only from the first public release (2021); later LICENSE moves retained the text, and no CLA/DCO policy was found in the public repository history (NOUREDDINE, 2021a; NOUREDDINE, 2026c)
<i>Individual</i>			
Personal Motivations	N/A	–	Academic incentives are visible, but contributors’ motivations are not observable from public data
Personal Knowledge	1	Medium	Better than PowerJoular’s: ~60% one author but a real repeat-contributor tail, tests encoding behaviour, a widely-known language
<i>Economic</i>			
Financial Resources	0	High	The project reports no dedicated funding; limited equipment and activity support in 2020–2021 did not create recurring project-controlled capacity (NOUREDDINE, 2026b)
External Value Capture	0	Medium	No commercial products, hosted offerings, consulting, or paid external participation is visible
Reinvestment	1	Medium	Academic supervision and limited material/activity funding have supported development, but support is narrow and non-recurring (NOUREDDINE, 2026b)
<i>Environmental</i>			
Environmental Req.	2	High	Its entire purpose is attributing energy to software for green software; explicit framing throughout

