How to Assess the Health of Open Source Software dependencies in an Organization's Intake Process: Insights from an Interview-survey and Case Study

JOHAN LINÅKER, RISE Research Institutes of Sweden, Sweden
THOMAS OLSSON, Robert Bosch AB, Sweden
EFI PAPATHEOCHAROUS, RISE Research Institutes of Sweden, Sweden

Motivation: The increasing reliance on Open Source Software (OSS) in organizational supply chains necessitates robust mechanisms to ensure the long-term viability and maintenance of these projects. Assessing OSS project health is complex due to the wide range of socio-technical factors involved. **Aim**: This study aims to develop a comprehensive framework for characterizing and assessing the health of OSS projects in the context of organizational intake processes. **Method**: We conducted an interview survey with 17 industry experts and a case study at a large international automotive manufacturer to synthesize a health assessment framework, and evaluate its practical application. **Results**: The study identified five key areas of OSS health: community productivity and stability, orchestration, production processes, and outputs. These areas encompass 21 health aspects with 71 connected attributes in total. The case study demonstrated the framework's utility in creating a tailored health assessment process for the organization. **Conclusion**: The proposed framework provides a valuable tool for organizations to take proactive sourcing decisions and address potential issues in OSS projects early on. By diagnosing symptoms early and applying necessary treatments, organizations can mitigate risks and ensure the long-term viability of their OSS dependencies, thereby enhancing software stability and reliability.

CCS Concepts: • **Software and its engineering → Software development methods**; **Open source model**.

Additional Key Words and Phrases: Open Source Software, Software Ecosystem, Health, Sustainability, Software Quality.

ACM Reference Format:
Johan Linåker, Thomas Olsson, and Efi Papatheocharous. 2018. How to Assess the Health of Open Source Software dependencies in an Organization's Intake Process: Insights from an Interview-survey and Case Study. In . ACM, New York, NY, USA, 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The presence and significance of Open Source Software (OSS) in organizations' supply chains, products, and services are indisputable and ever-growing [8, 14, 41]. By extension, so is the dependence on the OSS project's ability to stay viable and maintained long-term without interruption or weakening, also referred to as the OSS project health [11, 21]. Threats to the health of an OSS project may come in many shapes and forms. Toxicity [43] and non-inclusive culture [31], non-responsive communication [40], and lack of documentation [4] may prevent the on-boarding of new contributors. Burn-out or change of interest may cause maintainers to shift away from the OSS project [25]. Projects maintained by only a few individuals may risk becoming abandoned by consequence or have bugs and vulnerabilities emerge [41].

In earlier review work, we show how OSS project health is a wide topic, identifying 107 health aspects ranging across the socio-technical spectra of the OSS communities' peer-production processes, its orchestration and governance, and

Manuscript submitted to ACM

the software and other deliverables coming out of production process [21]. The health aspects could further be viewed from a project-centric or on the interconnected ecosystem level of abstraction [24].

The review further shows that extant research, including tools and models, does not provide a comprehensive overview, neither on the ecosystem nor project-centric level. The wide set of potential metrics does, however, pose a challenge for anyone to leverage in a health assessment of an OSS project. In this study, we continue on our path exploring **how the health of an OSS project can be assessed**, aiming to synthesize and prioritize a comprehensive yet distilled set of health aspects from which organizations aiming to set up a health assessment process can draw from. We specifically look at the context of an organization's intake process of OSS, where they evaluate and monitor OSS components adopted or considered for adoption in products and operations.

Building upon the reporting in literature [21], we turn to the practitioner side and empirically explore the problem domain. We report the outputs from a semi-structured interview survey of 17 experts from the industry and OSS ecosystem on what aspects they consider important and how these aspects can be further characterized from their experience and point of view. Findings are synthesized into a health assessment framework, including five areas of health aspects that need consideration, including the OSS community's productivity and stability, its orchestration, and the production process and output. Across the five areas, 21 health aspects were identified along with a total of 72 attributes that further help to characterize the different health aspects.

We consider the customization of the evaluation process pivotal as each organization may experience different risks and challenges and, thereby, needs based on the context (e.g., industry, market, technology) they operate within. Our framework and its underpinning data should hence be considered as a source of design knowledge [32] for the tailoring and implementation of the assessment process in the concerned organization. We specifically note that not all OSS projects can be assessed and compared equally and identify four project traits that should be considered before assessing the health of an OSS project, including its life-cycle stage, complexity, governance concentration, and strategic importance for concerned organizations.

To evaluate the use of the framework in the problem context, we perform a case study at a large international automotive manufacturer with high OSS dependencies present. Health attributes were narrowed down further to a questionnaire through a focus group and evaluated through four user observations where company developers applied the questionnaire to OSS projects of internal interest and discussed its potential implementation in practice. A candidate process is designed and proposed based on the context and needs of the case company.

Our study, accordingly, provides practitioners with a health assessment framework and guidance for how this can be applied to set up up a health assessment process internally. Such a process, much as going to a medical doctor, can help an organization to proactively identify potential symptoms, make conclusions of potential issues, and apply the necessary treatments early on to minimize or remove potential risks and harmful impact.

The rest of the paper is structured as follows. In Section2, we discuss related work on OSS quality models and health assessment motivating the gap and positioning of our work. Our research design spanning over two cycles is presented in further detail in Section 3. The health assessment framework is presented in detail in Section 4, followed by the presentation of how it may be applied in practice through our case study in Section 5. In Section 6, we discuss project traits that may impact how OSS projects should be assessed and compared. This is followed in Section 7 by a discussion on the limitations and how threats to validity are managed. Finally, the main conclusions of the study are summarized and presented in Section 8.

2 RELATED WORK

Below, we first provide an overview of related work on OSS health assessment based on our earlier literature survey of 146 publications [21]. Second, we explain and motivate the scope of our study, and how it addresses a gap in research contrasted through the related work.

2.1 OSS health assessment

Evaluation of OSS projects' quality aspects is thoroughly addressed with several early contributions generally characterized as quantitative means of aggregating technical metrics to arrive at a sourcing decision [27, 35, 42, 47]. Petrinja and Succi, e.g., propose the QualiPSo Open Source Maturity Model (OMM) based on the more general Capability Maturity Model Integration (CMMI) [27]. They focus specifically on analyzing the development process and evaluating the OMM through desk research on six OSS projects.

Some works focus less on metric definition and specifically on tool development and automation. Goeminne and Mens developed a tool for analyzing the developer activity across OSS projects within a wider ecosystem [10]. The tool is evaluated through an empirical case study of the GNOME OSS ecosystem. In an underpinning and earlier work, authors developed an automated tool for visualization and analysis of OSS project health leveraging metrics identified through the FLOSSMetrics.org project [9]. Samoladas et al. selected a set of common source code metrics through literature and developed the SQO-OSS quality model and an automated analysis tool for applying the model to OSS projects [35]. In more recent work, Gonzalez-Barahona et al. present a new generation of toolsets emerging that supports software development analytics, OSS included from the perspectives of companies, developers, and OSS foundations [12].

Certain works have a more explicit focus on the empirical investigation of OSS projects and their ecosystems, using OSS health as a lens of analysis. Kabbedijk and Jansen, e.g., perform an empirical investigation of the Ruby ecosystem and its large set of OSS projects (gems) [15]. They use a set of code and social media centrality metrics to analyze the activity and collaboration across the projects. Ververs et al. empirically investigate the Debian community to identify factors that promote developer participation in the development [45]. Gamalielsson et al. quantitatively investigate the Nagios OSS project's health using social network centrality metrics [7]. They look specifically at both individuals and service providers. Oriol et al. developed a tool, OSS-CARE [26], that implements a number of key health indicators from the Queso quality model [6], such as activeness, by aggregating individual metrics, such as a number of contributors and open bugs. The tool focuses specifically on an ecosystem level, considering multiple OSS projects integrated, further illustrated by their evaluation of the Eclipse Foundation's rich set of OSS projects.

Several works have taken an ecosystem perspective on health, stemming from the natural ecosystem analogy. Wang et al., e.g., designed an algorithm using a natural ecosystem analogy and developing related indicators through a grounded theory investigation of literature [46]. Carvalho developed a framework for health evaluation of software ecosystems (not specific for OSS) [3]. Fifty-eight metrics were collected from four previous studies and operationalized through automated tool-support. The framework is evaluated through desk research on a scientific software ecosystem. Van Lingen et al. present a framework with health indicators based on related work to evaluate the health of three Content management software ecosystems [44]. The indicators are evaluated using either computation, manual inspection, or a survey of community participants.

There has also been work focused on developing metrics and more or less comprehensive support on the project level. Liao et al. propose a model for predicting the OSS project health using GitHub data [18]. Valiev et al. empirically investigated a series of metrics through a mixed-methods investigation of the PyPi OSS ecosystem. Qiu et al. [30]

elicited health indicators by surveying maintainers and operationalized the indicators in a dashboard. The dashboard is focused on displaying social aspects of the community development efforts, thereby raising awareness and providing guidance for the maintainers to improve their health accordingly. Guzani et al. also developed a dashboard aimed at maintainers presenting metrics for improving attraction and retention of newcomers [13]. Singh et al. [38] focus on the community aspects, developing an assessment framework for ranking and comparing communities using the "Order of Preference by Similarity to Ideal Solution from Multi-Criteria Decision-Making toolkit." The framework is developed based on GitHub data and validated through the application on nine OSS projects, which are compared and contrasted in terms of their community health. Poth et al. developed an automated tool-support labeled Open Source Quality Radar (OSQR) to be used by development teams within the Volkswagen Group IT for self-selection of OSS components. The tools extract data from GitHub based on a set of metrics related to community, code quality, and issue management, which was derived from internal experts.

Finally, extant work has taken several perspectives in the development and application of OSS health metrics. Shaikh and Levina investigate the potential for growing and building alliances and business relations through OSS communities [37]. Li et al. [17], Adewumi et al. [1], and Spinelli [39] in turn focus on factors to consider before sourcing a specific OSS component, similar to Poth et al. [29]. Butler et al. look at factors that can improve internal capabilities to consume OSS [2]. Guizani et al. [13] and Qiu et al. [30] take the maintainers' perspective on how they can monitor and improve the health of their projects.

2.2 Study motivation and gap analysis

In contrast to related work, our study and contribution stand out in several ways. For example, our proposed health assessment framework is not limited to considering either the ecosystem or project level of analysis, quantitative over qualitative metrics, or focused on designing automated tooling before deriving empirically grounded metrics. The empirical investigation by Van Lingen represents an exception but is, however, more focused on investigating the problem context and less on designing a solution proposal [44]. The notable work by Qiu et al. [30] and Guizani et al. [13] provides an empirically grounded dashboard and support for health monitoring. These are, however, developed primarily from the maintainers' perspective, while our focus is on the organizational intake and dependency monitoring perspective.

Poth et al. developed a tool for quantitatively analyzing the health of OSS projects for the internal teams of Volkswagen Group IT to use when sourcing OSS components [29]. While taking the organizational intake perspective similar to our study (and others [17, 39]), they focus primarily on developing tool-support for the quantitative evaluation and less on qualitative metrics. Our study further considers the additional use case of enabling the analysis of existing OSS dependencies, also from the organizational perspective. An additional general limitation among related work is that any empirical validation of proposed models or tools beyond desktop research of smaller samples of OSS projects is limited. In our study, we perform a case study at a large international automotive manufacturer and consumer of OSS to evaluate and demonstrate the applicability of our proposed assessment framework.

3 RESEARCH DESIGN

This study adopts a design science research approach [32] as visualized in Fig. 1, building upon our prior research where we explored the literature to find out how the health of an OSS project may be assessed [21]. The literature survey provided a first design cycle during which we investigated the problem context and outlined an initial framework of 107 health aspects. In this study, we continue the design process in addressing our main research question of **how the**

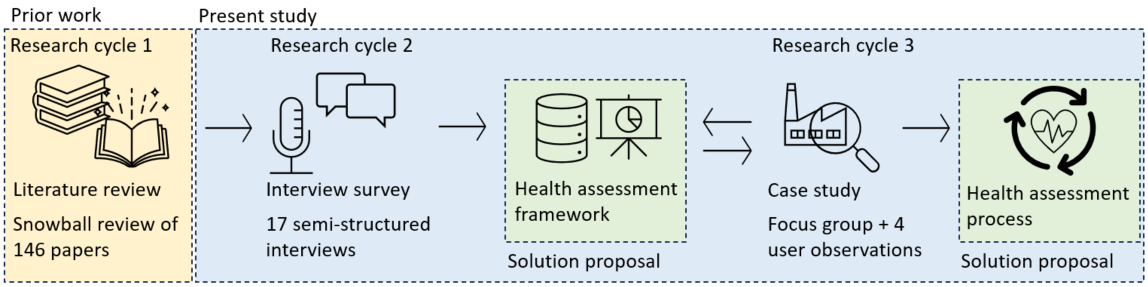

Fig. 1. Overview of the research process spanning over three design cycles. Building on prior work from cycle 1 [21], this study presents the outputs from cycles 2 and 3, constituted by an interview survey and case study, respectively.

health of an OSS project can be assessed. A health assessment framework is, therefore, presented and iteratively improved, along with a health assessment process illustrating how the framework may be applied in a real-world context.

We started with an interview survey with experienced practitioners (cycle 2, see Fig. 1) to further validate, extend, and contextualize the aspects previously identified. The revised framework rendered in five areas with a total of 21 health **aspects**, each covering *a particular part of OSS project health that can cause issues with consequences for the OSS project, its community, and end-users*. For each aspect, we elicit **attributes** help to *break down and enable the analysis of a concerned aspect in regards to an OSS project*. The framework provides design knowledge for organizations to draw from when designing and implementing their own health assessment processes, both from a sourcing and dependency-management perspective.

To evaluate and demonstrate the applicability of the framework, we perform a case study in cycle 3, investigating the potential implementation at a case company. The process started with a focus group where participants in groups prioritized aspects and related attributes most important in their context and practice. A questionnaire was generated, including the concerned aspects and attributes, and then evaluated through a set of user observations where participants from the focus group individually applied the questionnaire to evaluate the health of the OSS project of their choice. A final version of the questionnaire was transferred to the case company for further implementation into their internal development processes. Below, we present the research approach in further detail.

3.1 Interview Survey

Interviewees were sampled from two groups. The first group focused on a specific case company, which included experts from program management, cybersecurity, tools- and infrastructure, and product engineering. This sampling was motivated by the fact that we wanted to get a comprehensive view of the aspects from a confined context. By sampling interviewees from different parts of a case company, we could get complementary perspectives of aspects that were considered important for the case company at large. Interviewees were identified through snowball recommendation from our first interviewee, the open source program manager, until interviewees and we considered that all relevant views of the company had been captured.

Our second group of interviewees consisted of general experts with 10+ years of professional experience in working with OSS from a community or company perspective, either on a technical or strategic level. All interviewees also confirmed that they had repeated experience in analyzing the health of OSS projects, although based on their personal experience and rationale. Interviewees were identified through industry networks, including the Linux Foundation and the CHAOSS community. Our intention was to improve the generalizability of our interview survey, although we acknowledge the limitations that come with our qualitative approach and limited overall interview sample. Interviewees were added until saturation could be noted. In total, 17 interviews were conducted, of which five were from the first group sample and 12 from the second (see table 1).

Table 1. Overview of the interviewees, the sample they belong to (case or general), their title, type of organization they represent, and a brief motivation for their relevance for the study.

ID	Sample	Title	Type of Organization	Relevance for project
I1	Case	Open Source Program Officer, Company's OSPO	Automotive manufacturer	Responsible for OSS operations of the case company and 20+ years of working within the OSS ecosystem.
I2	General	Head of Research and Data Science	Data analytics	Responsible for the research and development of product features related to OSS health analytics.
I3	Case	Lead architect within software development tools	Automotive manufacturer	Oversees the sourcing, adoption, and integration of OSS in the area of software development tools.
I4	Case	Senior IT Security Architect	Automotive manufacturer	Responsible for processes and practices related to internal security reviews, including OSS.
I5	General	Expert engineer	Manufacturer of embedded software devices	Internal advocate and expert on OSS development and processes. Maintainer of several OSS projects.
I6	General	Director of Sales	Data analytics	Manages products and services focused on OSS health analytics. Has a PhD. degree on the topic.
I7	General	Open Source Program Officer, Company's OSPO	Car manufacturer	Responsible for the OSS program within the organization. Has long experience working with OSS strategically.
I8	General	Senior Manager, OSPO	OSS-based service and product provider	Responsible for community outreach and metrics programs, including the development, training, and implementation of Health metrics.
I9	General	Research Analyst, Company's OSPO	Cloud services	Supporting implementation of health metrics internally and engaging in external collaborations on their development.

I10	General	Open Source Manager, Company's OSPO	Telecom infrastructure	Responsible for the introduction of OSS security practices inside the company and externally engaged in OSS Security communities developing best practices on the topic.
I11	General	Consultant	Independent	Former senior manager of OSS operations in large OSS-based service providers. Long-term experience of OSS community growth and business.
I12	General	CEO	Data analytics	Manages products and services focused on OSS health analytics. Has a PhD. degree on a related topic.
I13	General	Director of Community and Developer Relations	Cloud-based Database service provider	Engaged in several community initiatives related to OSS health metrics, with leadership experience from OSS foundations.
I14	Case	Senior Solutions Architect	Automotive manufacture	Responsible for the introduction and harmonization of Security practices among development teams.
I15	Case	Security Engineer	Automotive manufacture	Experience in security evaluation and sourcing of OSS components for current and previous employers.
I16	General	Open Source Manager, Company's OSPO	Infrastructure software for cloud services	Responsible for implementing metrics inside the company and engaging in external development. Has a Ph.D. on an adjacent topic.
I17	General	Open Source Program Manager, Company's OSPO	Online Audio platform	Responsible for implementing metrics inside the company and adopting security practices.

A semi-structured approach was adopted, where interviewees were asked open questions in terms of how they would characterize a healthy OSS project, provided our definition. The open question was asked repeatedly through the two dimensions of the framework derived from literature [21]. After being allowed initial open reasoning, the interviewee was asked to reflect specifically on each related aspect within the specific dimension that had not yet been touched upon.

Each interview lasted for about 60 minutes and was conducted by the first and second authors. The first author facilitated the interviews, while the second author took notes. The interviews were conducted remotely via an online video platform, recorded, and automatically transcribed. Transcriptions were manually processed and structured by the second author. Interviewees were provided with a copy of the transcript with the option to correct, add, or retract any statement.

The transcripts were coded separately by both the first and second author using the health aspects identified by our previous literature review as an a-priori code book, also known as structural coding [34]. Each paragraph could be assigned multiple codes (simultaneous coding [34]) within or across different themes (i.e., health aspects), enabling analysis of co-occurrences and contextual understanding of the individual codes.

Both authors made personal notes attached to each coded paragraph to summarize the main points and capture initial reflections. Both authors made four synchronizations where codes identified codes were discussed per paragraph until agreement was made. The number of disputes was reduced to only minor variations in the last round. No inter-coder agreement was calculated as all 17 interviews were dually coded and synchronized.

The first author walked through the coded paragraph per each of the codes and synthesized observations from the interview statement and the first and second authors' previous notes. Codes with two (N=12), one (N=21), or no mentions (N=29) were excluded from the analysis, which resulted in a total of N=54 codes from the revised a-priori code book (N=117). The open coding was discussed iteratively with the second author throughout the process. The codes were then axially coded and synthesized within each of the 21 health aspects, which were further synthesized in five overarching areas through selective coding. In section 4, the final codes are presented as attributes per health aspect along with a brief description capturing the perspectives highlighted by the interviewees. The codebook is published and accessible as part of the supplementary material to this paper [20].

3.2 Case study: Implementation at a Case Company

The case company is a large international automotive manufacturer with 50,000+ employees. They use OSS both on board the automotives in a safety-critical environment, in the cloud for enabling connected services, and in the internal development and infrastructure environment.

3.2.1 Focus Group. The focus group included 16 participants from a broader team focusing on developing and maintaining the internal tools and infrastructure for software development, testing, and building pipelines. The first author facilitated the focus group, which lasted for two hours. Participants were first provided with a background on the study and a top-level description of the framework as elicited from the interview survey. Participants were then divided into three groups where they first had to prioritize the top five most important health aspects (as presented in section 4) and motivate why (hand-outs available in the supplementary material [20]). In the second step, groups were asked to prioritize the most important attribute per aspect (also as presented in section 4). Finally, they were asked to discuss their general thoughts about the health check process, and how it can be implemented into the current development processes, and what the barriers might be.

Each group made their own notes, which were collected by the first author. After the group discussions, the whole group openly discussed the top prioritized health aspects and related attributes, concrete examples of where a health check process would have been needed in the past, and how such a process could look in practice. Following the focus group, the first author made further notes summarizing the open discussion of the focus group. The participants' priorities were cross-compiled, resulting in a new and briefer version of the health assessment framework compared to the general version elicited from the interview survey. The new version was member-checked with the manager of the team (I1), who also attended the focus group.

3.2.2 User observations. A set of four user observation sessions was performed where users were sampled from the team attending the focus group and from the product development teams inside the case company. Sampling was performed in collaboration with the manager (I1) of the team who attended the focus group. Selection criteria included a general familiarity with OSS development practices, and an OSS project in particular that was to be evaluated.

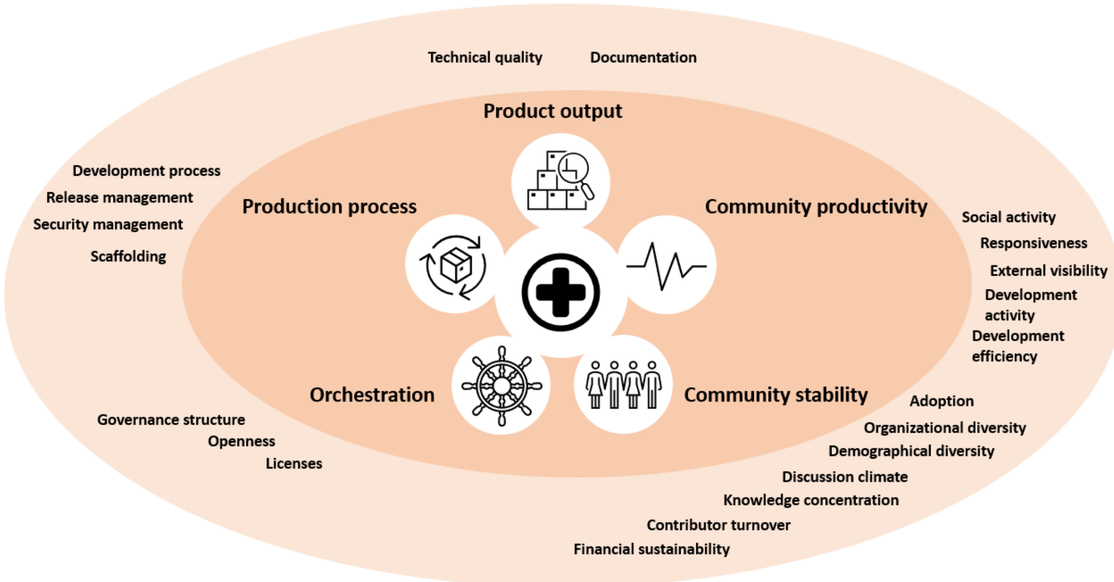

Fig. 2. Overview of the health assessment framework, including its 21 health aspects divided across five areas, that help to characterize the health of an OSS project. Each aspect has related attributes (not visualized) that help to characterize the aspect.

Observation sessions lasted between 60-120 minutes and were managed and recorded over an online videoconferencing platform. The first author of this study facilitated the sessions. Each session was initiated with an overview of the research project, the scope and purpose of the session, and the revised framework that came out of the focus group prioritization. Following this, the user provided a background and connection to the OSS project to be evaluated. The user then performed the evaluation based on the attributes identified from the focus group, which had been formalized in a questionnaire (see supplementary material [20]). The user was then asked to evaluate each attribute from the questionnaire based on their own experience and by using the online resources and information on the OSS project available at hand.

The first author, facilitating the evaluation, took notes continuously and kept an observational role throughout the evaluation, providing clarifications when needed, but did not direct or guide the user in any way to minimize the introduction of researcher bias. After each evaluation, the questionnaire was revised to improve clarity and context. Each evaluation was then synthesized as a case that exemplifies and provides context for how the framework and its attributes may be applied and evaluated differently for different projects.

4 THE HEALTH ASSESSMENT FRAMEWORK

Below, we present the synthesized output from our qualitative interview survey and case study in the form of a health assessment framework (see Fig. 2). The framework consists of 21 aspects, each covering *a particular part of OSS project health that can cause issues with consequences for the OSS project, its community, and end-users.* For each aspect, a number of **attributes** are defined to help *break down and enable the analysis of a concerned aspect in regards to an OSS project.* The health aspects are further ordered into five high-level themes.

- Community productivity - Aspects describing the pulse and activity that contribute to the collaborative development of the OSS project.
- Community stability - Aspects describing the robustness and ability of the community to withstand any interruption or change that may impact its population of maintainers and contributors,
- Orchestration - Aspects related to the governance and coordination mechanisms that enable the development and collaboration within the OSS community.
- Production process - Aspects of the peer-production process generating the OSS project outputs.
- Production outputs - Aspects describing the quality and comprehensiveness of the OSS project outputs, including source code and documentation.

See table 2 for an overview of the framework, including the concerned areas, aspects, and attributes.

Table 2. Overview of the health assessment framework elicited from the interview survey, including its 21 health aspects.

Aspect	**Description**
Community productivity	
Social activity	The activity from the OSS project's community and maintainers both in online channels and physically.
Responsiveness	The time to a response from the maintainers and community towards, e.g., discussions, pull requests, or issues.
External visibility	Visibility of the OSS project to an audience outside the community of individuals actively engaged in the project.
Development activity	The overall development activity by the community, including the many technical aspects and deliverables of the OSS project.
Development efficiency	The effectiveness and ease of the maintainers and contributors of an OSS project in managing and moving the development forward.
Community stability	
Adoption	Usage and technical adoption of the OSS project as a dependency in downstream software projects and by end-users.
Organizational diversity	The diversity of organizations within an OSS community in terms of governance, contribution, and adoption of the underpinning project.
Demographical diversity	The individual level of the maintainer and contributors to an OSS project in gender, race, time zone, language, and cultural aspects.
Discussion climate	The discussion climate in the community in regard to sentiment, tone, and manner in answers, messages, and general communication within the OSS project, as well as how potential conflicts are managed.
Knowledge concentration	The concentration or distribution of contributions and knowledge to specific individuals or groupings within an OSS project (also referred to as the bus or truck factor).

Contributor turnover	The attraction, retention, and attrition of maintainers and contributors to an OSS project.
Financial sustainability	The financial situation of maintainers and contributors of OSS projects and whether it enables sustainable and dedicated time for maintenance of the projects.
Orchestration	
Governance structure	The explicitness, formality, and general recognition of the ecosystem's governance structure and leadership.
Openness	To what extent the OSS project is welcoming to and accepting contributions and considering new ideas and general input and influence on the project's development from existing and new contributors
Licenses	License-related aspects and processes of managing and distributing the intellectual property maintained by the OSS project.
Production process	
Development process	The presence and quality of development processes are seen by multiple interviewees as an important marker of a mature and sustainable OSS project.
Release management	The release process should describe the governance and planning of how releases are made and at what cadence
Security management	The implementation and management of proactive and reactive measures to prevent and address security concerns of the OSS project.
Scaffolding	The availability and quality of the development and communication infrastructure used in the OSS project.
Production output	
Documentation	The presence and quality of documentation for the OSS project considering different stakeholders' perspectives, including developers and end-users.
Technical quality	The technical quality of the OSS and its source code, e.g., in terms of its architecture, source code, and other quality attributes.

4.1 Community Productivity

4.1.1 Social Activity. Concerns about the activity from the OSS project's community and maintainers both in online channels and physically offline (e.g., I5 and I11). Online, the activity can be in the shape of posts, discussions, and interactions in mailing lists, issue trackers, or the community's own social media channels. Offline, the activity may be in the form of dedicated conferences, hackathons, and meetups or contributions to such events but with a more general focus.

Beyond the focal OSS project under investigation, the presence of social communication with upstream and downstream projects was also stressed (I6C1). For libraries, it is mostly important to have active relationships with downstream users. For end-user-facing projects, relationships with upstream projects are important (I6C4).

Social activity is seen as a sign of healthy development and a social pulse of the OSS project. It may also provide a proxy for the popularity of a project, which by many interviewees was considered an ambiguous yet essential aspect of a project (e.g., I6 and I12).

The view on social activity varies through the life cycle stages of a project (I11). In the early phases, growth signals are important, e.g., through activity in communication and visibility on public events. A decline phase may be noted when the communication and technical activity slows down, although this can also be sign of a project that is stabilizing why additional aspects should be consulted.

Attributes:

- **A1** The OSS project's communication with up- and downstream projects, e.g., in reporting and assisting on the discussion of relevant issues (I6C1)
- **A2** The OSS project's activity on its communication channels, e.g., mailing lists, issue trackers, or social media.
- **A3** The OSS project's offline activity through arranging and participating in events, e.g., conferences, hackathons, and meetups.
- **A4** The social activity of the maintainers in contrast to the remaining community on the OSS project's communication channels, e.g., mailing lists, issue trackers, or social media.

4.1.2 Responsiveness. Responsiveness concerns the time to respond from the maintainers and community towards, e.g., discussions, pull requests, or issues (e.g., I2 and I8). The type of response depends on the medium and may, e.g., include an answer to a question, a code review of a pull request, or the prioritization or closing of an issue. Security- and vulnerability-related topics and issues were highlighted as necessary in terms of responsiveness due to their critical nature (e.g., I1 and I9). Some see responsiveness as a proxy for the availability of the maintainers, their workload, and the size and activity of the community in general (I2). By extension, the aspect is a sign of a community's productivity. Long lead times can signal bottlenecks in the community (I12), that the project has been abandoned or is in the process of becoming. The aspect may vary depending on the life-cycle stage (I8), yet should still be short, as responsiveness to vulnerabilities is essential regardless of whether a project is in its growth or stability phase. Accordingly, observing the responsiveness over time is important to catch any negative trends early on.

Attributes:

- **A1** The timeliness and quality of responses to, e.g., new discussion questions, pull requests, or issues (e.g., I5 and I17).
- **A2** The responsiveness specifically in terms of security-related discussions, pull requests, or issues (e.g., I1 and I10).
- **A3** The responsiveness of the maintainer in contrast to the rest of the community (e.g., I1 and I2).

4.1.3 External Visibility. Visibility of the OSS project to an audience outside the community of individuals actively engaged in the project. Simple signs of appreciation or attention of individuals, such as stars or followers, as well as reporting on external events, social media, forums, and news, are some potential indicators (I11C4 and I6C12).

Increased visibility can be a sign of increased external interest, which in turn can provide a foundation for, e.g., increased attraction of contributors, and chances for securing sponsorships (I6C11). However, it may not necessarily be a sign of increased adoption or development. Hence, as with many other aspects it needs to be consider in the context of complementary metrics to, e.g., gain an understanding of a project's popularity.

Visibility can be expected to vary across the life cycle of an OSS project (I11C4) related to a "hype" or new release of a project. Low overall visibility can be a sign of demise, but the project is also feature complete, i.e., stable, which is not necessarily negative (I13C7). Complementary metrics are hence needed to define where the life cycle of a project resides before considering this aspect in detail.

Attributes:

- **A1** The reporting of the project in external events, social media, forums, and news (I11C4 and I6C12).
- **A2** The project's external visibility signaled through various popularity indicators, e.g., stars and followers on GitHub (I12C22), or downloads from the package manager (I8C25).

4.1.4 Development Activity. Concerns the overall development activity by the community, including the many technical aspects and deliverables of the OSS project. Both code and non-code contributions should be considered (e.g., I6C27 and I16C18), although the latter may be more difficult to measure (I6C27). These may, e.g., be directed to the code base, documentation, peer-review and quality assurance process, or release management process.

The development activity is considered one of the more critical aspects by interviewees, signaling that support can potentially be provided if something were to go wrong (I3C9). I5C3 and I11C6 highlight that it should be considered with other aspects, such as social activity, to get an overall pulse of the OSS project. Looking at the activity historically, one can also guess how it may look onward (e.g., I7C2 and I13C10).

The development activity should be analyzed both on the project as a whole and in terms of its potentially different submodules or parts. A low or declining activity may be a sign of an unmaintained (orphan) code (I12C13). I16 further adds the importance of considering the activity in relation to the quality of the work, e.g., in terms of source code and documentation.

Another perspective provided by several interviewees is that the development activity should be put in relation to the number of maintainers and contributors doing the actual work. The ratio, tying into the aspect of knowledge concentration, can give further hints on how the workload distribution looks. Ideally, the project should have activity from both the maintainers and contributors (long-term and episodic) (I16C18).

Attributes:

- **A1** The contribution activity to the code base over time, e.g., last 45, 90, and 365 days.
- **A2** The activity in development-related activities, such as code reviews, merging of PRs, and actions on issues over equal periods of time.
- **A3** The activity in contributions to non-code tasks, e.g., documentation, test cases, and release management activities over equal periods of time (e.g., I10C17 and I6C27).
- **A4** The different types of activities in contrast between the maintainer(s), long-term contributors, and drive-by contributors (I9C4).

4.1.5 Development Efficiency. Concerns the effectiveness and ease of the maintainers and contributors of an OSS project in managing and moving the development forward. E.g., in terms of addressing, reviewing, merging, and closing pull requests, as well as performing regular and timely releases.

Development efficiency relates to the responsiveness in the communication of the OSS projects but speaks to the progression and productivity of the development (I9C27). Hence, it should be contrasted to the development activity, which describes the pulse of the general development, and less about the quality and productivity of the development (I16C18).

A growing backlog of unaddressed issues or PRs, or a growing ratio of those addressed, can be a bad sign, raising the question of whether the community can manage the workload (I15C3). The efficiency should, therefore, also be contrasted against, e.g., the knowledge concentration to get further perspective on the work distribution.

Cadence and consistency in terms of releases and updates is regarded as an important feature of a stable project (e.g., I16C18 and I17C12). Timeliness is especially highlighted for security-related issues and solutions, from how they are managed to being released (e.g., I5C12 and I15C3), which can signal how the project prioritizes its work.

Attributes:

- **A1** The size and evolution of the project's backlog in terms of open and unresolved issues (I12C16).
- **A2** The pace in how issues are being addressed and closed and PRs merged (I5C25).
- **A3** The corresponding responsiveness towards bugs and security-related issues and PRs (e.g., I5C12 and I15C3).
- **A4** The cadence and timeliness at which releases are made and planned (e.g., I1 and I17).

4.2 Community Stability

4.2.1 Adoption. Usage and technical adoption of the OSS project as a dependency in downstream software projects and by end-users is considered by several interviewees as an important marker for a healthy project (e.g., I16C7 and I3C7). Presence of actors in the community (I11C5), recorded dependencies (I5C7), and other popularity indicators (I12C22) may provide different signs of adoption.

Interviewees highlight that several metrics would be needed to get a fair understanding of the adoption as it is considered difficult to put a number on (I2C33). Also, several interviewees highlight that the technical adoption will vary pending the life cycle stage of the project (e.g., I11C5 and I16C5).

Adoption among larger organizations considered a positive sign in terms of quality and that they may be motivated to act if a vulnerability would be introduced (e.g., I5C7).

Attributes:

- **A1** The diversity of individuals and organizations represented in the project's various communication channels and ongoing development (I11C5).
- **A2** The adoption of the project by other downstream projects (I5C7).
- **A3** The project's adoption signaled through various popularity indicators, e.g., stars and followers on GitHub (I12C22), or downloads from the package manager (I8C25).

4.2.2 Organizational Diversity. The diversity of organizations within an OSS community in terms of governance, contribution, and adoption of the underpinning project. Some potential metrics highlighted are the distribution of governance positions (I1C21) or the relative size of contributions between actors (I9C20).

A commonly perceived risk in OSS projects with low organizational diversity in terms of contribution and governance is that the project may depend on the agenda of a limited number of organizations and that influence and contributions from other actors could be improved (e.g., I5C6 and I6C3). E.g., explicitly when the project is part of a single-vendor business model or implicitly when the number of maintainers or contributors from one organization is significantly larger than other actors. Two potential implications are a change of license or a drop of support for the project (e.g., I17C15 and I16C11).

Attributes:

- **A1** The diversity of organizations represented in the governance of the project and its distribution of positions, e.g., in governance and technical steering committees (I1C21).

- **A2** The diversity of organizations represented by the individuals contributing to the project and the distribution of contributions among them (I9C20).
- **A3** The diversity of organizations represented among the outspoken users of the project (I5C6).

4.2.3 Demographic Diversity. Demographic diversity looks at the individual level of the maintainer and contributors to an OSS project in terms of gender, race, time zone, language, and cultural aspects (e.g., I9C23 and I12C20). It may include both online in the social activity and development of the OSS project (I13C19) and offline at physical events related to the project (I6C10).

The importance is raised by many interviewees who consider demographical diversity as crucial for open culture and quality of development due to the different perspectives (e.g., I16C14). Stability and productivity are also implied by the broader exposure and adoption, and by extension, a large surface area for contributions (I11C2). One interviewee raised the Fedora OSS project as an example of a highly diverse project in terms of contributions despite mainly being driven by Red Hat (I11C2).

The aspect is, however, also considered difficult to measure practically online due to the general anonymity of a community (I6C10) and the sensitive and ethical nature of collecting such information (I9C24). Measuring diversity at physical events and larger formal organizations (such as foundations) is considered more accessible (I6C10).

Attributes:

- **A1** Reports in terms of diversity produced on the project considering, e.g., gender, race, time zone, language, and cultural aspects (I9C24).
- **A2** The turnout at physical events arranged by the project? (I6C10).

4.2.4 Discussion Climate. Regards the discussion climate in the community in regard to sentiment, tone, and manner in answers, messages, and general communication within the OSS project, and how potential conflicts are managed. Examples include rudeness, deliberate misunderstandings, and closing of issues without reason (I1C32).

Many interviewees consider the presence of toxicity and heated discussions as markers for an unhealthy community that will have a negative impact on both the attraction and retention of contributors (e.g., I5C13 and I9C17). The discussion climate should rather be friendly, constructive, and welcoming to create an inclusive environment, increasing the potential attraction and retention rates of contributors. Toxicity needs to be identified upfront and managed proactively (I9C25), typically by introducing and enforcing a code of conduct and setting up a governance structure that can manage conflicts in a structured way (I8C11).

The character of the discussion climate in a community is considered difficult to measure beyond observing the dialogues taking place and the potential presence of a code of conduct (I17C24).

Attributes:

- **A1** The overall sentiment, tone, and manner in answers, messages, and general communication within the OSS project and how it is reflected in the documentation (I1C32).
- **A2** Use of slang, irony, or idiomatic expressions in technical discussions.
- **A3** Presence of conflicts, and how they are managed (I11C8).
- **A4** Presence of a code-of-conduct, and a governance and process for implementing it (I8C11).

4.2.5 Knowledge Concentration. Concerns the concentration or distribution of contributions and knowledge to specific individuals or groupings within an OSS project. The aspect is typically described in terms of, e.g., bus or truck factor, meaning the number of people that have to abandon the project for it to go dormant (e.g., I2C7 and I8C6). One interviewee suggests looking at the number of individuals making up certain amounts (e.g., 50 or 80 percent) of the contributions to a project (I14C18). The knowledge concentration can further regard the project as a whole and its parts or modules. One interviewee refers to the latter as an orphan code (I12C13), i.e., an unmaintained part of a larger OSS project. Concentration of knowledge is also important to capture in terms of who is resolving bug reports or failing builds, tasks not necessarily done by the contributors (I6C15).

In the inception phase of a project, the knowledge concentration is typically limited to one or a very few individuals but may also remain valid for later phases of a project's life-cycle (e.g., I2C18 and I8C9).

A low level of knowledge concentration may indicate a higher risk of maintainer burnout as the burden can grow overwhelming if, e.g., questions, feature requests, and code reviews increase disproportionally to the amount of time available from the maintainer(s) (I2C22). Financial sustainability for the maintainer becomes a more critical factor for these projects, as well as the quality, e.g., of documentation that may enable others to take over if needed (I1C39).

One interviewee highlights that there are higher chances that the project may survive if it's in the confinement of a foundation or larger ecosystem (I6C3). A related factor concerns the amount and diversity of users and downstream projects, which decreases the potential risks related to a low knowledge concentration (I8C20). The size and complexity of a project and its strategic importance affect how companies value the risk associated with the low knowledge concentration (e.g., I16C4 and I17C12).

Attributes:

- **A1** The number of individuals doing most of the development in the project (i.e., bus factor), e.g., in terms of 50 and 80 percent (I14C18), and the corresponding number for organizations (i.e., elephant factor).
- **A2** Presence of any submodules or parts in the project with a low bus or elephant factor, also referred to as orphan code (I12C13).
- **A3** The corresponding bus and elephant factors in terms of who maintenance tasks beyond code contributions, e.g., resolving issues, performing code reviews, or resolving build failures.

4.2.6 Contributor Turnover. Contributor turnover regards the attraction, retention, and attrition of maintainers and contributors to an OSS project. This also includes episodic volunteers who leave after one or a limited number of contributions.

The turnover is seen by many of the interviewees (e.g., I6C25 and I9C16) as a key aspect describing the stability and resilience of an OSS project. Episodic volunteers specifically are seen from different perspectives. On one side, they are considered to add to the load of the maintainer without contributing to the long-term sustainability of the project (I9C12 and I6C26). Still, it provides a signal that there is interest in the project and that it is receptable to contributions (I2C33).

Retaining the episodic volunteers to stay on as long-term contributors and potentially maintainers is further considered a challenge and a narrow funnel, yet a key sign for an attractive and inclusive community (I9C15 and I13C13). On the opposite, if the attrition rate is greater than the attraction and retention, i.e., when there is a negative turnover, companies should be vary of the stability of the concerned project. Turnover can, however, also be a good sign in terms of maintainers as it signals an organic growth in the community, that governance is not locked-in (I13C13).

Attributes:

- **A1** The number of new contributors attracted to the community over time, e.g., last 45, 90, and 365 days (e.g., I6C26).

A2 The number of new contributors that have made recurrent contributions and, to some extent, been retained to the community over the same periods.

A3 The turnover of maintainers, long-term contributors, and individuals in central governance positions in the project (I13C13).

4.2.7 Financial Sustainability. Regards the financial situation of maintainers and contributors of OSS projects and whether it enables sustainable and dedicated time for maintenance of the projects. It may concern whether maintainers and contributors are employed, have a personal business set up, or receive sponsorships (I6C14).

There is a general sense among interviewees that financial sustainability among the maintainers of a project is a positive sign (I5C17). This implies that they can work professionally with the maintenance while potentially promoting better work-life balance. However, it may also be seen as a risk if the continued maintenance or the attraction of contributions is dependent on, e.g., reoccurring one-time sponsorships (I1C36). Commercial backing, i.e., when individuals are employed and representing larger organizations, is further seen as a positive sign (I11C18) but should be seen in contrast to the organization's agendas and the openness for collaboration and contributions if they are in a central position of the project's governance (I2C15).

Attributes:

A1 The extent maintainers in any way are paid or sponsored to work professionally on maintaining the project, and how (I1C37).

A2 The extent of employed contributors that engage in the project and how financially solid their employers are (e.g., I11C18 and I16C20).

4.3 Orchestration

4.3.1 Governance Structure. Concerns the explicitness, formality, and general recognition of the ecosystem's governance structure and leadership.

Governance and code-of-conduct are considered critical criteria for a healthy OSS project (e.g., I8C23 and I9C17) and for proactively managing and minimizing conflicts and toxicity, which by extension, may lead to people abandoning the project. Some interviewees (e.g., I1C20 and I9C14) specifically raise the aspect of having rules in place for how maintainership is regulated, distributed, and transitioned when a maintainer leaves a project. Regulations and processes should further be documented and openly available (I5C19) but continuously revised actively to stay up-to-date (I13C14).

An OSS project's complexity and stage in its life-cycle heavily impact the requirements and needs for formality in terms of a project's governance (I1C18). In a small project or the early stages, less rigor is considered acceptable, and concentrated leadership is the focus of development and agility (I8C23). As the project grows and increases in complexity, more mature governance is preferred by companies that ensure a neutral space for collaboration and provide means of settling disputes and agreeing on a common agenda (e.g., I5C20 and I11C14), considered especially important in cases where there are conflicting business incentives present (I10C13). I9C18 highlights with reference to the Kubernetes project that an efficient governance structure should enable a decentralized development where decisions are taken at a level as low as possible.

Governance is, however, considered difficult to measure quantitatively, although one may look for the presence of specific paragraphs or types of documents, such as a code of conduct (I2C23 and I8C23).

Attributes:

A1 Presence of a code-of-conduct and governance and process for how it is enforced project (e.g., I8C23 and I9C17).

A2 Presence of rules for how maintainership is regulated, distributed, and transitioned when a maintainer leaves a project (e.g., I1C20 and I9C14).

A3 Presence, availability, and up-to-date documentation of governance in terms of how decisions are made and conflicts managed, by whom, and how they are elected (I5C19 and I13C14).

A4 Maturity of the governance of the project compared to its current size and life-cycle stage (I1C18).

4.3.2 Openness. Regards to what extent the OSS project is welcoming to and accepting contributions and considering new ideas and general input and influence on the project's development from existing and new contributors (e.g., I1C34 and I2C7).

Openness is considered a significant factor for projects deemed strategically essential and where the governance and ownership are centered on one (or a limited number of) organization(s). One interviewee describes how their engineers typically enter a discussion with an OSS project about the ability to influence and the potential implementation of specific use cases (I17C16).

The openness is also considered crucial for the potential of a vibrant community which is an enabler for many of the benefits that OSS may bring (I16C14). The opposite imposes a negative culture that can impact the attraction and retention of people in the community.

Attributes:

A1 The openness for external contributions (I1C34).

A2 The extent contributions beyond maintainers and long-term contributors are accepted, e.g., from episodic volunteers (I2C25).

A3 Presence of an onboarding process and general support for newcomers for engaging and contributing to the project (I5C22).

4.3.3 Licenses. Concerns license-related aspects and processes of managing and distributing the intellectual property maintained by the OSS project.

Many interviewees highlight that a copyleft license may have a negative impact on company participation (e.g., I10C25 and I13C5). Factors such as license compatibility and flexibility with different business models are, however, highlighted as impacting factors (I12C23). The presence of a CLA may also impact the contribution process by adding friction, inhibiting contributions (I6C22), and raising the risk of license changes if there is a corporate entity driving the project (I17C17). However, some interviewees diverge in that it can both be a sign of maturity (e.g., I10C26 and I17C17). Export control may be another issue that can restrict or inhibit adoption from companies (I13C11).

From a commercial user perspective, the trustworthiness of the source of the OSS project in terms of its license compliance and correctness and how it may be verified with data from other sources (e.g., license databases) is specifically highlighted (I1C23). Depending on the business criticality of the project, a company has different requirements. For less critical projects, trusting the community and potentially performing a manual inspection in the hosting platform may be enough. For more critical OSS used in products, metadata may not be trusted, warranting scans and thorough compliance reviews (I1C24). Should the project reside within a more professional setting, such as a foundation, trust will probably be higher (I1C25).

Attributes:

A1 The types of OSS licenses the project is published under (e.g., I10C25 and I13C5).

A2 Presence of a Contributor License Agreement (I6C22), to whom the copyrights are transferred to (I17C17).

A3 Presence and quality of necessary legal information, e.g., in terms of included licenses and export control information (I13C11).

A4 Presence of a process for managing licenses and copyright within the OSS project.

4.4 Production Process

4.4.1 Development Process. The presence and quality of development processes are seen by multiple interviewees as an important marker of a mature and sustainable OSS project (e.g., I12C27 and I2C20). Each OSS project may be expected to have different ways of working, so it is important to consider the differences (I2C28). The development process encompasses multiple parts, including requirements engineering, design, implementation, testing, and release management.

The requirements engineering process should describe how new and existing requirements are managed (I1C45), e.g., through an issue tracker, how these are discussed and prioritized, and planned for, both in releases and on the more forward looking roadmap.

The development process should further provide guidance on how development is performed, including a contribution process describing how contributions are made and managed (e.g., I2C20 and I13C16). A process entailing how newcomers to a project can start to engage in the project is also highlighted among several interviewees (e.g., I1C35 and I13C21). The use of modern infrastructure for developing the OSS is a sign of process maturity and up-to-date in terms of modern ways of working (e.g., I1C55 and I16C21). Automated testing, CI/CD, and automatic scanning tools are highlighted examples (e.g., I5C24 and I6C15).

Attributes:

A1 Process for how requirements are identified, discussed, prioritized, and planned (I1C45).

A2 Process describing how contributions should be made and how these are managed (e.g., I2C20 and I13C16).

A3 Process for onboarding newcomers to the project, e.g., in terms of joining discussions and making contributions (e.g., I1C35 and I13C21).

A4 Process and infrastructure for quality assurance of the project, both continuously and per release (I9C27).

4.4.2 Release Management. The release process should describe the governance and planning of how releases are made and at what cadence. Handling security and bug fixes is of extra importance (e.g., I2C26 and I16C6). Releases should be clearly described and documented, including impacting dependencies (I14C12).

Attributes:

A1 The structure and transparency of the software release process (e.g., I1C42 and I14C12).

A2 The type of format releases are packaged in and whether they are signed with PGP appropriately (I1C40).

A3 Process for managing breaking changes (I2C26).

A4 The quality of releases, e.g., in terms of build quality and documentation (I14C15).

A5 The cadence and consistency of releases (e.g., I14C17 and I17C22).

4.4.3 Security Management. Software security is a pivotal area for all types of software (I11C17). One interviewee considers it difficult to predict the risk for vulnerabilities and that many vulnerabilities may be a good sign, just like none as a community might be good at identifying new ones (I2C30). Yet, the number of past and present vulnerabilities and the responsiveness in how these were resolved are highlighted by several interviewees as important numbers to consider (e.g., I10C21 and I1C30). Two interviewees note that it is important for users to be aware of their risk appetite (I13C10) and the level of trust that they put into concerned OSS projects (I3C19).

The dependencies are another important aspect to consider. If a project depends on a lot of other upstream projects, the risk increases for vulnerabilities being imported (I16C8). The presence of vulnerabilities in dependencies is, therefore, also a potential red flag that should be scanned for (I15C3). However, this is considered by many interviewees as a difficult and costly procedure, especially in relation to how thorough and far up the dependency tree one wants to go (e.g., I16C9 and I1C16). One interviewee highlights the importance of observing which ecosystem a project belongs to, as each works differently in terms of how releases and dependencies are managed (I2C5). In NPM, e.g., one is dependent on each project to update, while In Maven, you can brute force updates.

Security practices are important for all types of OSS projects (e.g., I17C7 and I5C27), including smaller ones, as these can often have central positions in dependency networks, allowing potential vulnerabilities to propagate widely in supply chains (I1C22). One interviewee notes that smaller projects might not have as many security updates as larger projects, implying that they can have different security practices without it being a problem for the health of the project (I16C3). Regardless, there should always be a strict review and quality assurance process in place for PRs before being merged into the project code base (I10C16).

Processes should further be in place and transparently documented regarding how vulnerabilities are reported, discussed, managed, disclosed, and communicated (e.g., I1C51 and I10C15). Contact details should be openly disclosed for reporting and discussion of security-related information (I14C9). Adopting best practices as suggested by industry best practice programs such as those of the Open Source Security Foundation is also highlighted (I6C17), including measures such as enabling multi factor authorization (I4C6), and running automated security and vulnerability scanning (I16C22).

Attributes:

- **A1** The presence of vulnerabilities that have been reported, e.g., during the last 1, 3, and 12 months, and at what rate have these been addressed (e.g., I10C21 and I1C30).
- **A2** The corresponding presence in upstream dependencies of the focal OSS project (I15C3).
- **A3** The review and quality assurance practices for pull requests to the OSS project (I10C16).
- **A4** Processes for how vulnerabilities are reported, discussed, managed, disclosed, and communicated by the OSS project (e.g., I1C51 and I10C15).
- **A5** Adoption of the best practices suggested by the Open Source Security Foundation (I6C17), such as enabling multi-factor authorization (I4C6) and running automated security and vulnerability scanning (I16C22).
- **A6** Security and release management processes in the ecosystem that the focal OSS project resides in (I2C5).

4.4.4 Scaffolding. Scaffolding concerns the availability and quality of the development and communication infrastructure used in the OSS project. The presence and use of CI/CD, as well as test automation, were especially highlighted as important markers of technical quality (e.g., I10C14 and I15C8). One interviewee describes how it creates trust in the quality assurance practices of the community (I13C16). The presence of code scanning tools, e.g., for static component analysis and fuzzy testing, and the presence of vulnerabilities (e.g., I6C16 and I16C22) were highlighted as markers of a more mature project, not to be expected of projects in earlier life-cycle stages.

Attributes:

- **A1** Presence of a functioning CI/CD pipeline (e.g., I10C14 and I15C8), its process, and maintenance of it.
- **A2** Presence of a functioning test automation (I13C16), and the test coverage of the project.

- **A3** Use of any type of scanning tools, e.g., for static component analysis, fuzzy testing, and presence of vulnerabilities (e.g., I6C16 and I16C22).

4.5 Production Output

4.5.1 Documentation. Documentation is considered critical among many of the interviewees, both from a developer (I10C22) and user perspective. It enables the capture, persistence, and dissemination of knowledge among the existing community as well as newcomers. It is not something that is always prioritized (e.g., I1C45 and I13C17) and should cover both deliverables and processes. Deliverables include source code, release notes, and export control (e.g., I15C11 and I1C43), while typical processes and process-related artifacts may include onboarding, contribution guidelines, requirements planning and roadmap, training, and FAQs (I13C17).

General quality attributes of the documentation, including its availability, correctness, and completeness, were further highlighted (e.g., I6C21 and I3C22). Accessibility was also raised in terms of enabling the visually impaired, different screen sizes, and mental structure of the text (I6C21). Such aspects may require a manual inspection to evaluate, although some quantitative metrics may be used, such as its revision history, when measuring development activity.

Attributes:

- **A1** The technical documentation of deliverables, including source code and releases, e.g., in terms of completeness, up-to-date, correctness, and accessibility.
- **A2** The process-related documentation, e.g., related to the development process, and governance structure, e.g., in terms of completeness, up-to-date, correctness, and accessibility.
- **A3** The onboarding documentation for enabling newcomers to engage with the OSS project, e.g., in terms of completeness, up-to-date, correctness, and accessibility.

4.5.2 Technical Quality. Concerns the technical quality of the OSS and its source code, e.g., in terms of its architecture, source code, and other quality attributes. Several interviewees highlight readability, clean code, and general adherence to common coding conventions as important markers (e.g., I15C9 and I17C5). A user needs to be able to understand the code and be able to build on top of it (I3C21). The architecture should further be investigated if it makes logical sense (I1C53) and preferably has a modularized structure that is easy to extend and add to (I6C23). The presence of circular dependencies, or different versions of the same project/package, is also highlighted as something to look for (I1C50).

Attributes:

- **A1** The readability and adherence to common coding conventions of the source code (e.g., I15C9 and I17C5).
- **A2** The logic and appropriateness of the OSS project's architecture (I1C53) and whether it has a modular structure (I6C23).
- **A3** Presence of circular dependencies, or to different versions of the same project/package (I1C50).

5 CASE STUDY: IMPLEMENTATION AT A CASE COMPANY

The health assessment framework (see Fig. 2) provides a knowledge base for organizations to pick-up and learn from when evaluating an OSS project's health. In cycle 3, we are specifically interested in the use case of an organization's intake process of OSS, including sourcing new OSS dependencies and monitoring those already running in products and operations.

We conducted a case study at an international automotive manufacturer to evaluate how the framework could be used in practice and support the implementation of health assessment in the intake process. The case company is a

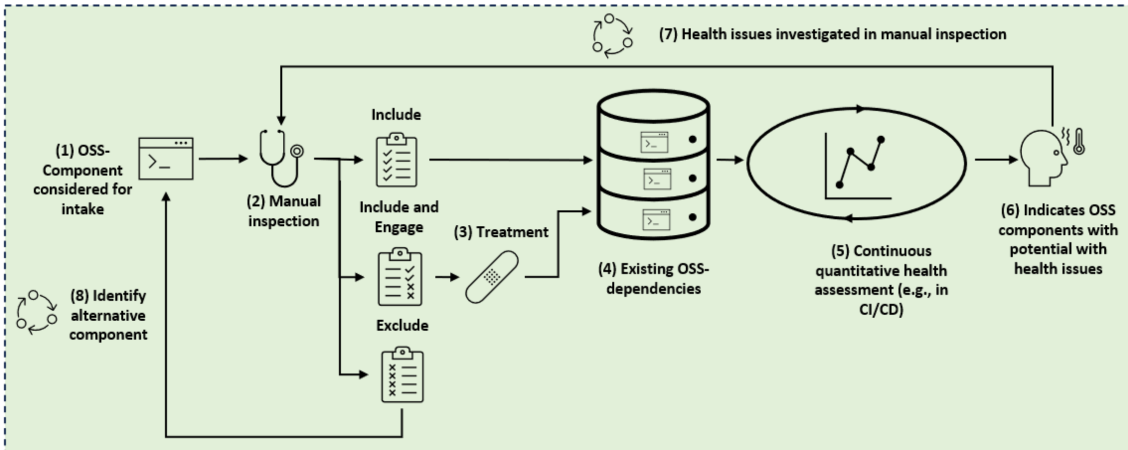

Fig. 3. Overview of the health assessment process, part of an organization's overarching intake process for OSS components.

partner in this study's overarching research project. A process design and questionnaire for health assessment was designed iteratively based on interviews of domain experts within the organization as part of the interview survey (I1, I3-4, I14-15), complemented with a follow-up focus group and user observations where the questionnaire was applied and evaluated.

The process, as visualized in Fig. 3, is typically triggered (1) by a developer identifying an OSS component and candidate for inclusion as a dependency in the organization's products and operations. If the component meets initial functional requirements, the developer performs a manual inspection (2) using a standardized questionnaire with questions identified as necessary based on the health assessment framework. For the case company, a condensed list of health aspects and metrics (see table 3) were prioritized through the focus group and user observations.

Tool-support should be developed and leveraged to support the developer is answering the questions and identify any health issues. Existing tooling from the CHAOSS project was discussed as potential candidates to start from [1]. The development of the tool-support should iteratively help to refine the questionnaire. The goal is to keep a manual inspection between 15 and 60 minutes, depending on the complexity of the OSS component. The time aspect was raised as a critical aspect in focus groups and user observations as time is already limited, and the cost for adding additional burden cannot be too high as it will then be de-prioritized and potentially avoided.

Based on the evaluation, the developer will make a triage decision including the three options:

- If the component is considered healthy, the component is included.
- If the component shows health-issues that are manageable, the component is included but with a treatment designed and implemented to address any issues identified.
- If the component shows health-issues that are not manageable, the component is excluded, and the search will continue for an alternative component (8).

For option number two, the treatment (3) should be designed based on the health issues identified. The health assessment framework along with the underpinning literature identified in earlier work [21] can provide input, along

[1] https://chaoss.community/software/

with best practice established in industry and the OSS community[2]. Interviewees highlighted that suitable treatments should be designed and documented iteratively and connected to the different questions asked in the health assessment. The definition of suitable treatments is, however, beyond the scope of this study.

For OSS included as dependencies (4) to the organization's products and operations, a quantitative analysis is run continuously (5), e.g., as part of the internal continuous integration, delivery, and deployment pipeline. The analysis should align with the questionnaire and be based on the health assessment framework but exclude aspects that can only be determined qualitatively. The tool-support developed for the manual inspection should preferably be reused, automated, and integrated into an internal notification system that alerts developers responsible for the component when indicators are passed above-identified thresholds (6).

Interviewees note that the inspection is potentially best performed by individuals with context knowledge and experience of the OSS component as this will help to interpret any health issues indicated by the quantitative analysis. The developer assigned to the manual inspection reiterate the inspection (7) focused on the areas identified by the quantitative analysis and enter the triage stage as described above.

Several interviewees both in the case company and in the general sample highlight that the health of an OSS project cannot be generalized by numbers. Hence, it is important as further raised that developers are continuously trained in how to apply the questionnaire and tool-support for the manual inspection. There must be an understanding of what the different questions mean, why they are asked, and how the answers can be interpreted, both in isolation and in combination.

Participants both from the focus groups and user observations raise the need for knowledge sharing and collective learning on OSS health and what it can imply. Iterative workshops and retrospects where developers can share their assessments, gain feedback and discuss is proposed as a key part. Standardized training modules and local champions who may provide points of contact are also emphasized.

A third point regards the persisting of health assessments in an organization-wide repository where developers across different units and departments can gain insights into how a project has evolved, both from the eyes of manual inspections and from the continuous monitoring of the quantitative analysis. Such records will also enable follow-up on the OSS project's evolution and help provide an understanding of how certain health issues have emerged and how they potentially should be treated.

Table 3. Overview of questions and attributes identified and prioritized by the case company to be used for manual inspections in the health assessment as part of the organization's overarching intake process of OSS components. Specific comments included from the focus group participants.

Community productivity - Development activity
A1: The contribution activity to the code base over time, e.g., last 45, 90, and 365 days.
A2: The activity in development-related activities, such as code-reviews, merging of PRs, and actions on issues over equal periods of time.
General note: *"Important because we need to know if the software is being worked on and is up-to date. Indicates that we will likely have future and current support."*
Community productivity - Responsiveness

[2]E.g., https://github.com/ossf/scorecard, https://chaoss.community/, https://standard.publiccode.net/

A1: The timeliness and quality of responses to, e.g., new discussions questions, pull requests, or issues.
General note: The group emphasizes the importance to consider both the social coding platform where a project is hosted, and also external platforms such as StackOverflow where project-related questions may be asked.

Community stability - Adoption

A3: The project's adoption signaled through various popularity indicators, e.g., stars and followers on GitHub, or downloads from the package manager.
General note: The group notes that *"More people and good scores means more activity (bug reports/fixes, acceptance, security findings, etc.)"*.

Orchestration - Licenses

A3: Presence and quality of necessary legal information, e.g., in terms of included licenses, and export control information.
General note: *"Indicates that legal factors have been taken into consideration"*.

Orchestration - Governance structure

A3: Presence, availability, and up-to-date documentation of governance in terms of how decisions are made, and conflicts managed, by whom, and how they are elected.

Production processes - Security management

A1: The presence of vulnerabilities that has been reported, e.g., during the last 1, 3 and 12 months, and at what rate has these been addressed.
General note: *"Shows that we can be confident that bugs are being reported/fixed and builds trust in our usage of it."*
A4: Processes for how vulnerabilities are reported, discussed, managed, disclosed, and communicated by the OSS project (e.g., I1C51 and I10C15).
A5: Adopted of the best practices suggested by the Open Source Security Foundation such as enabling multi factor authorization, and running automated security and vulnerability scanning.
General note: *"How do you trust that the Open Source is secure? You can scan the Open Source for security vulnerabilities using, e.g., Dynamic or static analysis"*.

Production processes - Development process

A4: Process and infrastructure for quality assurance of the project, both continuously, and per release.

Production processes - Release management

A1: The structure and transparency of the software release process.

Production output - Documentation

A1: The technical documentation of deliverables, including source code and releases, e.g., in terms of completeness, up-to-date, correctness, and accessibility.

"This means that we can utilize the project in the way it is intended to be used. Removes/reduces guess-work as well as less misconfigurations."
Production output - Technical quality
A1: The readability, and adherence to common coding conventions of the source code. A2: The logic and appropriateness of the OSS project's architecture (I1C53), and whether it has a modular structure. *"Evaluate 1) against our need and guardrails, 2) readability and understandability, and 3) use of proper structure and coding rules (naming convention, code linting etc.)"*

6 ANALYSIS: CAN ALL TYPES OF OSS PROJECTS BE ASSESSED AND COMPARED EQUALLY?

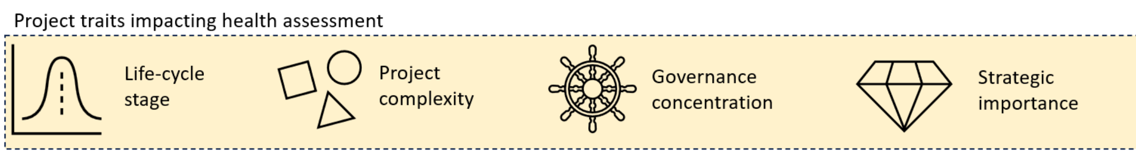

Fig. 4. Four OSS project traits that impact how OSS projects should be assessed and compared in terms of their health.

Analyzing the health of an OSS project is a complex task. In our interviews with practitioners, we synthesize and narrow down 21 aspects, where different attributes can be applied to further characterize the concerned health aspect. By extension, we cannot talk about OSS project health as one thing. Rather, there can be a number of potential health issues for OSS projects, just as for living beings. By noting symptoms and asking informed questions, these health issues may be identified, and suitable treatments may be prescribed.

However, aspects identified in our framework may not necessarily be evaluated equally across all types of OSS projects. As put by I10, *"…you first need to classify what type of project you have at hand and then, based on that type of project, you can factor this into the risk analysis"*. Accordingly, just as living beings have different conditions, traits, and personalities, so do OSS projects and their communities. Through our study, we noted certain traits as especially influencing how the different aspects potentially should be applied, specifically the life-cycle, complexity, and governance concentration of the OSS project and its strategic importance to the organization analyzing the project as part of its intake process (see Fig. 4).

Our proposed health framework does not consider these traits in detail. Such a mapping between the traits and how they impact each of the proposed health aspects requires a study of its own and, therefore, a topic for *future research* warranting both qualitative and quantitative investigations. Our understanding from practice and the research process for this study, is that numbers can only provide a subset of an answer, and will require contextual understanding and experience to translate into a nuanced problem understanding, and actionable insights.

For the moment, however, we strongly recommend practitioners aiming to adopt the presented health assessment framework to only comparing OSS projects with similar traits (e.g., complexity and life-cycle stage) and prioritizing and defining relevant aspects and attributes based on internal needs and risk appetite. The traits that are identified in our interviews are discussed further below.

6.1 Life-Cycle Stage

Determining what stage in a life-cycle a project resides in is in itself not an easy task. According to interviewees, the stage connects to (among other things) age, popularity, activity, and maturity. Yet it is difficult to generalize about any exact correlations. On the other hand, there is a wealth of literature attempting to classify [4, 36], and predict the life cycle evolution[5, 16, 19, 23, 28, 48]. Considering the interviewees, we observed four potential stages for an OSS project: 1) inception, 2) growth, 3) stabilization, and 4) decline.

The inception phase covers the initiation and early evolution of a project. Some projects may see a lot of traction and rapid development from a small core team and few or no drive-by contributors, while others will emerge at a slower pace. I11 stresses the importance of growth signals, e.g., in terms of technical and social activity. Interviewees also highlight that expectations on certain aspects, such as governance in younger projects, should not be as high as for more mature projects. In part because there may not be a need for it and in part because the focus is on development.

In a growth phase, the project continues to mature, both technically and socially. I2 and I8, e.g., highlight that there will probably be more open issues than closed as popularity continues to grow, and community takes shape. Contribution process starts to emerge, along with general development and governance processes. The latter will be needed to stakeholders with different agendas to find common ground and settle disputes (I11).

In the stabilization phase, the level of activity lowers, as does the number of new feature implementations. For projects in this phase, one should not expect active development but, e.g., require documentation to be up to date (I9) and some level of responsiveness still (I12). A stable project will likely still have releases and updates, at least in terms of security fixes either internally or upstream dependencies, whereas a dormant project likely will not (I16). Also, the importance of knowledge concentration to a limited number of individuals may be less than in previous phases (I8), yet with higher requirements on whether there are resources in place to enable others to take over the project should it go dormant (I3).

In the decline phase, activity is limited to non-existent impacting the quality and relevance of the OSS. The project may still solve a problem and be used but overtaken by another project (I2), which is also a good sign (I11). However, determining between if a project is stable or in decline can be tricky (I3) as limited or absent activity can also be due to feature completeness (I7).

Future work, should specifically look to how health aspects fluctuate and interrelate throughout the different life cycles. The priority of health aspects to consider, and acceptance criteria will potentially also fluctuate, and in need of investigation.

6.2 Project Complexity

The project complexity regards the scope, size, and technical complexity of the codebase maintained by the OSS project. For example, comparing Kubernetes with a smaller NPM JavaScript component will most likely render in different questions, and acceptance criteria.

Smaller projects with limited scope imply lesser requirements on maturity in governance as these projects need agility and speed. Too much governance can add friction (I11). In more complex projects, there will be a need for more mature governance processes to facilitate collaboration between, e.g., companies with different agendas. Also, as highlighted by I16, development processes may need to be more mature, e.g., in managing security issues. Yet, the quality of the security process is as essential as for complex projects. I1 believes that less complex projects can become more critical as vulnerabilities can propagate more widely through these in software supply chains unnoticed.

The view on knowledge concentration is also dependent on project complexity to a certain extent, where a small number of maintainers may be acceptable if the project is less complex. I16 explains the lesser requirements are because less complex projects are typically easier to replace. I16 and I3C further describe how some may assume in general that more complex projects are healthy by default, exemplified through Kubernetes and the Linux kernel OSS projects.

While our interviews highlighted scope, size, and technical complexity of the codebase, there may likely be other project complexities to consider when assessing the health of OSS project. *Future work* should explore what project complexities are considered relevant, both from a practitioners perspective, and quantitatively through mining software repository research.

6.3 Governance Concentration

This category concerns the concentration of governance and its impact on the project's openness to input and external influence on decisions and transparency of discussions with individuals and corporations engaged or with interest in the OSS project.

Companies see risk when an OSS project is governed by a single entity (e.g., leveraging a single-vendor OSS business model) (I9), both in terms of potential license changes to more restrictive versions and potential changes in the technical direction of development not in line with the overall community (I10). The active collection and transfer of copyright from contributors to a single entity through a Contributor License Agreement may be a warning sign (I17). Further, it may increase the knowledge concentration of the project, increasing the risk of the project going dormant if the company were to pivot and abandon the project (I11).

If a project belongs to a foundation, it is better prepared to care for its sustainability if contributors leave (e.g., I10 and I16). However, even though a project is under a foundation, one must look at the distribution of seats and power (I9), as the project can still be dominated by one or a few powerful actors (I13). The diversity of companies performing most of the contributions (e.g., 50 or 80 percent) can be an indicator metric to consider (I10), also referred to as the elephant factor by I9. If contributions are rather diverse, it may indicate an openness, which is why a concentrated governance structure may be an acceptable risk (e.g., I10 and I17). I13 further highlights that there should be a turnover of people in central positions; otherwise, dependence and lock-in to specific individuals are enforced.

I1, e.g., considers governance setups with Pay-to-play as a specific warning flag, i.e., sponsorship is required to get a seat at the table. Governance should preferably be open with influence based on technical merit, with companies' businesses kept separate from the technical development of the OSS (I6).

Future work should look into characterize the different examples of OSS projects with concentrated governance, and investigate how governance has propagated. There are several examples of single-vendor OSS projects, e.g., in the database space, that can serve as potential cases. In these, the vendor has transitioned from OSS to what may be referred to as source available and non-compete licenses. The effect these licence changes has in reality, and under what conditions should be clarified. Also, to what extent extant health aspects and attributes can be applied, and what can be acceptable thresholds.

6.4 Strategic Importance

OSS projects considered critical for an organization's business imply a lower risk appetite, implying stricter requirements of the concerned health aspects. Business criticality is a factor of both the strategic importance of the project and how easily it is to replace (I3). Smaller and less strategic projects should preferably have alternative solutions available with a low barrier to entry. I5 and I7 both highlight the case where products they ship, including embedded software, need to be maintained for the foreseeable future; why the same or corresponding requirements for internally developed software also need to apply for the OSS included in the products? Proactive health-improving measures may be motivated to secure the sustainability of concerned OSS (e.g., I7 and I8).

Considering the actor-perspective, financial support and stability of the project and its main supporting actors are considered pivotal (I2), including, e.g., the maintainer(s), contributors, and hosting OSS foundation. The need for the trustworthiness of the individuals in the project further increases, as does the organizational diversity of the project. Specifically, for projects with a low governance and knowledge concentration, there is a need for a high level of openness for external contributions and influence (e.g., I4 and I7). On the technical side, requirements also increase on the maturity and quality of, e.g., documentation, development processes, and security practices (e.g., I3 and I4).

While interviewees agree the strategic importance impact, determining the strategic importance is left in the mist. *Future work* should explore how such assessment can be made, e.g., from the business and technical perspectives [22]. The business model perspective can provide a potential lens in analysing how the OSS project is leveraged in the value creation and capture process.

7 LIMITATIONS AND THREATS TO VALIDITY

To discuss the limitations and threats to validity, we use the criteria defined by Runeson and Höst [33]; *construct validity, internal validity, external validity, and reliability.*

Construct validity concerns whether what was investigated was actually what the research had in mind. To guide our research, we have taken points from our earlier work and review of OSS health aspects in the literature and used established definitions of health in the planning and execution of our study. Interviewees were introduced to our definition and asked open questions related to the dimensions of the previously reported version of our health framework. In the analysis, transcripts were coded by the two first authors and discussed continuously to reach an agreement. Member checking was also thoroughly performed with all interviewees and study participants, both in cycles 2 and 3 of our research.

Internal validity considers whether external factors may have influenced the object under investigation. In terms of OSS project health, this may provide a significant threat as OSS health is a very complex construct, as illustrated in our earlier and present work [21]. We, therefore, urge readers to take specific care in the translation and application of our results in a real-world context. Multiple aspects need to be considered in combination together with the risk appetite of the organization. The experience of the individual performing a health assessment will also impact the analysis. Per our case study and the elicited assessment process, we, therefore, recommend that training, peer review, and knowledge sharing be adopted as cornerstones within any organization aiming to establish a health assessment component in their OSS intake process.

External validity considers the generalizability and transferability of the findings to other cases and contexts beyond what has been studied. As we note in Section 6, aspects and attributes that are considered relevant and acceptance criteria for these attributes will differ depending on the type of project. Four factors we identified from the interviews include the OSS project's life cycle stage, complexity and governance concentration, as well as strategic importance for the organization doing the health assessment. How these factors (and potential others) impact what aspects to investigate, what acceptance criteria to apply, and in what combinations or order is a comprehensive topic for future research. Until then, we recommend organizations aiming to adopt the health assessment framework to compare projects of equal complexity, type, life-cycle stage, etc., and to prioritize aspects and attributes similar to the case study presented in Section 5.

Reliability regards the replicability and transparency of the research method applied in the study. We have maintained an audit trail from transcriptions and throughout the coding process. The code book is published in the supplementary material along with the questionnaire and focus group material used in the case study to further enable replicability and transparency in our research process [20].

8 CONCLUSIONS

The dependence on OSS is ever-growing among organizations today, and accordingly, there is also the need for the OSS projects to stay healthy, i.e., the long-term and viable maintenance of the projects without interruption or weakening in their level of quality. Continuous assessment and monitoring of the health of OSS components used or considered for adoption is, therefore, a critical practice for these organizations to ensure the robustness and reliability of their software systems.

Our study presents a health assessment framework for organizations to implement in their intake processes for new or adopted OSS components. The framework highlights five key areas of health: community productivity and stability, orchestration, production processes, and outputs. These areas encompass 21 health aspects, each covering a particular part of the OSS project health that can cause issues with consequences for the OSS project, its community, and end-users. For each aspect, a number of attributes are defined to help break down and enable the analysis of a concerned aspect in regard to an OSS project.

The customization of the evaluation process is pivotal, as each organization faces unique risks and challenges based on its specific context, such as industry, market, and technology. Our framework serves as a source of design knowledge, enabling organizations to tailor and implement an effective health assessment process. The case study at a large international automotive manufacturer illustrates the practical application of our framework, demonstrating its utility in narrowing down health attributes to a questionnaire and designing a candidate process suited to the company's needs.

When assessing the health of an OSS project, organizations need to be aware of the type and traits of the OSS project at hand, as these factors may influence how the different health aspects of our proposed framework potentially should be applied and evaluated. Our interviewees specifically highlight the life-cycle, complexity, and governance concentration of the OSS project, and its strategic importance to the organization analysing the project as part of its intake process.

Our proposed health framework does not consider these traits in detail. Such a mapping between the traits and how they impact each of the proposed health aspects requires a study of its own and, therefore, a topic for future research. For practitioners aiming to adopt the presented health assessment framework, we firmly recommend only comparing OSS projects with similar traits (e.g., complexity and life-cycle stage) and prioritizing and defining relevant aspects and attributes based on internal needs and risk appetite.

Ultimately, this study provides practitioners with a valuable tool for proactively identifying potential issues within OSS projects, akin to a medical check-up. By diagnosing symptoms early and applying necessary treatments, organizations can mitigate risks and ensure the long-term viability and security of their OSS dependencies. This proactive approach not only enhances the stability and reliability of software products but also contributes to the overall sustainability of the OSS ecosystem.

REFERENCES

[1] Adewole Adewumi, Sanjay Misra, Nicholas Omoregbe, and Luis Fernandez Sanz. 2019. FOSSES: Framework for open-source software evaluation and selection. *Software: Practice and Experience* 49, 5 (2019), 780–812.

[2] Simon Butler, Jonas Gamalielsson, Björn Lundell, Christoffer Brax, Anders Mattsson, Tomas Gustavsson, Jonas Feist, Bengt Kvarnström, and Erik Lönroth. 2022. Considerations and challenges for the adoption of open source components in software-intensive businesses. *Journal of Systems and Software* 186 (2022), 111152.

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

[4] Jailton Coelho and Marco Tulio Valente. 2017. Why modern open source projects fail. In *Proc. of the 2017 11th Joint meeting on Foundations of Software Engineering* (Paderborn, Germany). ACM, New York, NY, USA, 186–196.

[5] Alexandre Decan, Eleni Constantinou, Tom Mens, and Henrique Rocha. 2020. GAP: Forecasting commit activity in git projects. *Journal of Systems and Software* 165 (2020), 110573.

[6] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. 2014. Queso a quality model for open source software ecosystems. In *2014 9th Int. conf. on Software Engineering and Applications* (Vienna, Austria). IEEE, 209–221.

[7] Jonas Gamalielsson, Björn Lundell, and Brian Lings. 2010. The Nagios community: An extended quantitative analysis. In *Open Source Software: New Horizons: 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010, Notre Dame, IN, USA, May 30–June 2, 2010. Proceedings 6*. Springer, 85–96.

[8] GitHub. 2023. *Octoverse: The state of open source and rise of AI in 2023*. Technical Report. GitHub.

[9] Mathieu Goeminne and Tom Mens. 2010. A framework for analysing and visualising open source software ecosystems. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)* (Antwerp, Belgium) *(IWPSE-EVOL '10)*. Association for Computing Machinery, New York, NY, USA, 42–47. https://doi.org/10.1145/1862372.1862384

[10] Mathieu Goeminne and Tom Mens. 2013. Analyzing ecosystems for open source software developer communities. In *Software Ecosystems*. Edward Elgar Publishing, 247–275.

[11] Sean Goggins, Kevin Lumbard, and Matt Germonprez. 2021. Open Source Community Health: Analytical Metrics and Their Corresponding Narratives. In *2021 IEEE/ACM 4th Int. Workshop on Software Health in Projects, Ecosystems and Communities* (Madrid, Spain). ACM, New York, NY, USA, 25–33.

[12] Jesus M. Gonzalez-Barahona, Daniel Izquierdo-Cortázar, and Gregorio Robles. 2022. Software Development Metrics With a Purpose. *Computer* 55, 4 (2022), 66–73. https://doi.org/10.1109/MC.2022.3145680

[13] Mariam Guizani, Thomas Zimmermann, Anita Sarma, and Denae Ford. 2022. Attracting and Retaining OSS Contributors with a Maintainer Dashboard. In *In 44th Int. conf. on Software Engineering: Software Engineering in Society* (Pittsburgh, PA, USA). ACM, New York, NY, USA, 5 pages.

[14] Red Hat. 2022. *The State of Enterprise Open Source*. Technical Report. Red Hat.

[15] Jaap Kabbedijk and Slinger Jansen. 2011. Steering Insight: An Exploration of the Ruby Software Ecosystem. In *Software Business*, Björn Regnell, Inge van de Weerd, and Olga De Troyer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 44–55.

[16] Xiaozhou Li, Sergio Moreschini, Fabiano Pecorelli, and Davide Taibi. 2022. OSSARA: Abandonment Risk Assessment for Embedded Open Source Components. *IEEE Software* 39, 04 (2022), 48–53.

[17] Xiaozhou Li, Sergio Moreschini, Zheying Zhang, and Davide Taibi. 2022. Exploring factors and metrics to select open source software components for integration: An empirical study. *Journal of Systems and Software* 188 (2022), 111255.

[18] Zhifang Liao, Mengjie Yi, Yan Wang, Shengzong Liu, Hui Liu, Yan Zhang, and Yun Zhou. 2019. Healthy or not: A way to predict ecosystem health in github. *Symmetry* 11, 2 (2019), 144.

[19] Zhifang Liao, Benhong Zhao, Shengzong Liu, Haozhi Jin, Dayu He, Liu Yang, Yan Zhang, and Jinsong Wu. 2019. A prediction model of the project life-span in open source software ecosystem. *Mobile Networks and Applications* 24, 4 (2019), 1382–1391.

[20] Johan Linåker, Tomas Ohlsson, and Efi Papatheocharous. 2024. Online Suppl. Material. https://doi.org/s/69182a009fb401a89dd2

[21] Johan Linåker, Efi Papatheocharous, and Thomas Olsson. 2022. How to characterize the health of an Open Source Software project? A snowball literature review of an emerging practice. In *Proceedings of the 18th International Symposium on Open Collaboration*. 1–12.

[22] Johan Linåker, Björn Regnell, and Daniela Damian. 2019. A Community Strategy Framework - How to obtain influence on requirements in meritocratic open source software communities? *Information and Software Technology* (2019).

[23] Héctor J Macho and Gregorio Robles. 2013. Preliminary lessons from a software evolution analysis of Moodle. In *Proc. of the First Int. conf. on Technological Ecosystem for Enhancing Multiculturality* (Salamanca, Spain). Association for Computing Machinery, New York, NY, USA, 157–161.

[24] Konstantinos Manikas and Klaus Marius Hansen. 2013. Reviewing the health of software ecosystems–a conceptual framework proposal. In *Proc. of the 5th Int. Workshop on Software Ecosystems*. Citeseer, 33–44.

[25] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up flossing? a study of contributor disengagement in open source. In *IFIP Int. conf. on Open Source Systems* (Montreal, QC, Canada). Springer, Cham, 116–129.

[26] Marc Oriol, Carlos Müller, Jordi Marco, Pablo Fernandez, Xavier Franch, and Antonio Ruiz-Cortés. 2023. Comprehensive assessment of open source software ecosystem health. *Internet of Things* 22 (2023), 100808.

[27] Etiel Petrinja and Giancarlo Succi. 2012. Assessing the open source development processes using OMM. *Advances in Software Engineering* 2012 (2012).

[28] Etiel Petrinja and Giancarlo Succi. 2012. Two evolution indicators for FOSS projects. In *IFIP Int. conf. on Open Source Systems* (Hammamet, Tunisia). Springer, Cham, 216–232.

[29] Alexander Poth, Dan-Alexander Levien, Olsi Rrjolli, and Matthias Wanjetscheck. 2022. Quality evaluation with the open-source quality-radar for a sustainable selection and use of FOSS components. In *European Conference on Software Process Improvement*. Springer, 503–517.

[30] Huilian Sophie Qiu, Anna Lieb, Jennifer Chou, Megan Carneal, Jasmine Mok, Emily Amspoker, Bogdan Vasilescu, and Laura Dabbish. 2023. Climate Coach: A Dashboard for Open-Source Maintainers to Overview Community Dynamics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–18.

[31] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. 2019. Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source. In *2019 IEEE/ACM 41st Int. conf. on Software Engineering* (Montreal, QC, Canada). IEEE, 688–699.

[32] Per Runeson, Emelie Engström, and Margaret-Anne Storey. 2020. The design science paradigm as a frame for empirical software engineering. In *Contemporary empirical methods in software engineering*. Springer, Cham, 127–147.

[33] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164.

[34] Johnny Saldaña. 2021. *The coding manual for qualitative researchers*. Sage.

[35] Ioannis Samoladas, Georgios Gousios, Diomidis Spinellis, and Ioannis Stamelos. 2008. The SQO-OSS quality model: measurement based open source software evaluation. In *Open Source Development, Communities and Quality: IFIP 20 th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy 4*. Springer, 237–248.

[36] Carlos Santos, George Kuk, Fabio Kon, and John Pearson. 2013. The attraction of contributors in free and open source software projects. *The Journal of Strategic Information Systems* 22, 1 (2013), 26–45.

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

[38] Jaswinder Singh, Anu Gupta, and Preet Kanwal. 2023. The vital role of community in open source software development: A framework for assessment and ranking. *Journal of Software: Evolution and Process* (2023), e2643.

[39] Diomidis Spinellis. 2019. How to select open source components. *Computer* 52, 12 (2019), 103–106.

[40] Igor Steinmacher, Marco Gerosa, Tayana U Conte, and David F Redmiles. 2019. Overcoming social barriers when contributing to open source software projects. *Computer Supported Cooperative Work (CSCW)* 28, 1 (2019), 247–290.

[41] Synopsis. 2023. *2024 Open Source Security and Risk Analysis Report*. Technical Report. Synopsis.

[42] Davide Taibi, Luigi Lavazza, and Sandro Morasca. 2007. OpenBQR: a framework for the assessment of OSS. In *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software, June 11–14, 2007, Limerick, Ireland 3*. Springer, 173–186.

[43] Parastou Tourani, Yujuan Jiang, and Bram Adams. 2014. Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem. In *24th Annual International conf. on Computer Science and Software Engineering* (Markham, ON, Canada), Vol. 14. IBM, 34–44.

[44] Sonny Van Lingen, Adrien Palomba, and Garm Lucassen. 2013. On the software ecosystem health of open source content management systems. In *5th Int. workshop on software ecosystems (iwseco 2013)*. Citeseer, 38.

[45] Eric Ververs, Rick Van Bommel, and Slinger Jansen. 2011. Influences on developer participation in the debian software ecosystem. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. 89–93.

[46] Lei Wang, Jing Wan, and Xinshu Gao. 2019. Toward the Health Measure for Open Source Software Ecosystem Via Projection Pursuit and Real-Coded Accelerated Genetic. *IEEE Access* 7 (2019), 87396–87409. https://doi.org/10.1109/ACCESS.2019.2926306

[47] Anthony Wasserman, Murugan Pal, and Christopher Chan. 2006. The business readiness rating model: an evaluation framework for open source. In *Proceedings of the EFOSS Workshop, Como, Italy*.

[48] Likang Yin, Zhuangzhi Chen, Qi Xuan, and Vladimir Filkov. 2021. *Sustainability Forecasting for Apache Incubator Projects*. ACM, New York, NY, USA, 1056–1067.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009