Evidence Based Library and Information Practice

Identifying Socio-Technical Risks in Open-Source Software for Scholarly Communications: Tools, Metrics, and Opportunities for Libraries to Support Sustainable Development

Pierre Lasou, Tomasz Neugebauer and Pamela Carson

Volume 20, Number 1, 2025

URI: https://id.erudit.org/iderudit/1117511ar
DOI: https://doi.org/10.18438/eblip30630

See table of contents

Publisher(s)

University of Alberta Library

ISSN

1715-720X (digital)

Explore this journal

Cite this article

Lasou, P., Neugebauer, T. & Carson, P. (2025). Identifying Socio-Technical Risks in Open-Source Software for Scholarly Communications: Tools, Metrics, and Opportunities for Libraries to Support Sustainable Development. *Evidence Based Library and Information Practice*, *20*(1), 20–48. https://doi.org/10.18438/eblip30630

Article abstract

Objective – In the interest of helping libraries make evidence based decisions about open-source software (OSS), the objective of this research is to establish whether tools that automate the evaluation of OSS project communities could be used specifically on scholarly communications OSS (SC-OSS) projects to provide actionable insights for libraries to guide strategic decision making and corrective interventions.

Methods – Seven OSS project communities were selected for evaluation, chosen from widely used scholarly communications software applications used in Canada for repositories, journal hosting, and archives. While all aspects of OSS projects may be evaluated at the project or network/ecosystem level, addressing the actors, software, or orchestration (Linåker et al., 2022), community evaluation that looks at the interaction patterns between project contributors is the practical focus of this research paper since there are multiple human factors that librarians who may not be software developers can impact. We identified a community analysis tool called csDetector (Almarimi et al., 2021) from the software engineering literature. This tool was chosen based on two main criteria: 1) ability to analyze data from GitHub repositories (the code sharing platform used by all selected SC-OSS projects) and 2) capacity to automatically produce results without manual intervention. Since some of the seven OSS projects were spread across multiple GitHub repositories, a total of 11 datasets from GitHub, each containing three months' worth of data, were analyzed using csDetector.

Results – The results produced by csDetector are interesting though not without limitations. The tool is complex and requires the user to have software development skills to use it effectively. It lacked sufficient documentation, which made interpreting the results challenging. The analysis from csDetector, which identifies community smells (i.e., types of organizational and social dysfunction within software projects [Tamburri et al, 2015, 2021a]), suggests that these SC-OSS project communities are experiencing knowledge sharing difficulties, weak collaboration practices, or other member interaction dysfunctions that can eventually permanently affect community health. Having a software tool that can take metrics from GitHub and detect community smells is a valuable way to illustrate problems in the project's community and point the way to remedying dysfunction.

Conclusion – While the OSS community analysis tool csDetector currently presents several hurdles before it can be used, and results generated come with caveats, it can be part of an approach to support evidence based decision-making pertaining to SC-OSS in libraries. The information provided can be worth monitoring (especially social network metrics such as centrality) and their results, particularly for community smells, identify problems that may be addressed by non-developers. Awareness of community smells in OSS can provide a deeper understanding of OSS sustainability as it provides a language to identify suboptimal social dynamics.

Evidence Based Library and Information Practice

Research Article

Identifying Socio-Technical Risks in Open-Source Software for Scholarly Communications: Tools, Metrics, and Opportunities for Libraries to Support Sustainable Development

Pierre Lasou
Scholarly Communications Librarian
Library of Université Laval
Québec City, Quebec, Canada
Email: pierre.lasou@bibl.ulaval.ca

Tomasz Neugebauer
Digital Projects & Systems Development Librarian
Concordia University Library
Montreal, Quebec, Canada
Email: tomasz.neugebauer@concordia.ca

Pamela Carson
Web Services Librarian
Concordia University Library
Montreal, Quebec, Canada
Email: pamela.carson@concordia.ca

Received: 18 Sept. 2024 **Accepted**: 15 Jan. 2025

Data Availability: Lasou, P., & Neugebauer, T. (2024a). Community patterns of scholarly communication open source software generated by YOSHI [dataset]. Borealis. https://doi.org/10.5683/SP3/4MEDXO

Data Availability: Lasou, P., & Neugebauer, T. (2024b). Community smells detection on scholarly communication open source software using csDetector [dataset]. Borealis. https://doi.org/10.5683/SP3/34MYPI

DOI: 10.18438/eblip30630

Abstract

Objective – In the interest of helping libraries make evidence based decisions about open-source software (OSS), the objective of this research is to establish whether tools that automate the evaluation of OSS project communities could be used specifically on scholarly communications OSS (SC-OSS) projects to provide actionable insights for libraries to guide strategic decision making and corrective interventions.

Methods – Seven OSS project communities were selected for evaluation, chosen from widely used scholarly communications software applications used in Canada for repositories, journal hosting, and archives. While all aspects of OSS projects may be evaluated at the project or network/ecosystem level, addressing the actors, software, or orchestration (Linåker et al., 2022), community evaluation that looks at the interaction patterns between project contributors is the practical focus of this research paper since there are multiple human factors that librarians who may not be software developers can impact. We identified a community analysis tool called csDetector (Almarimi et al., 2021) from the software engineering literature. This tool was chosen based on two main criteria: 1) ability to analyze data from GitHub repositories (the code sharing platform used by all selected SC-OSS projects) and 2) capacity to automatically produce results without manual intervention. Since some of the seven OSS projects were spread across multiple GitHub repositories, a total of 11 datasets from GitHub, each containing three months' worth of data, were analyzed using csDetector.

Results – The results produced by csDetector are interesting though not without limitations. The tool is complex and requires the user to have software development skills to use it effectively. It lacked sufficient documentation, which made interpreting the results challenging. The analysis from csDetector, which identifies community smells (i.e., types of organizational and social dysfunction within software projects [Tamburri et al, 2015, 2021a]), suggests that these SC-OSS project communities are experiencing knowledge sharing difficulties, weak collaboration practices, or other member interaction dysfunctions that can eventually permanently affect community health. Having a software tool that can take metrics from GitHub and detect community smells is a valuable way to illustrate problems in the project's community and point the way to remedying dysfunction.

Conclusion – While the OSS community analysis tool csDetector currently presents several hurdles before it can be used, and results generated come with caveats, it can be part of an approach to support evidence based decision-making pertaining to SC-OSS in libraries. The information provided can be worth monitoring (especially social network metrics such as centrality) and their results, particularly for community smells, identify problems that may be addressed by non-developers. Awareness of community smells in OSS can provide a deeper understanding of OSS sustainability as it provides a language to identify suboptimal social dynamics.

Introduction

As the open scholarship movement gains momentum (Martin, 2022) as governments commit to working in the open (Government of Canada, 2021) and government-funded research outputs are mandated to be openly available (Nelson, 2022), it continues to be important for academic libraries to be evaluating, adopting, and investing in open infrastructure for scholarly communications (Bilder et al., 2020; Cousijn et al., 2021). Key components for open infrastructure include open repositories: digital libraries of scholarly outputs, which, for the most part, are built using open-source software (OSS). There has long been an affinity between libraries and OSS (Chudnov, 1999; Corrado, 2005, 2021, 2023; Engard, 2010; Lease Morgan, 2002; Rhyno, 2004; Singh, 2020; Skog, 2023; Thacker and Knutson, 2015). Although OSS has many benefits over commercial software such as added control, and the opportunity to impact development, it also comes with risks. Unfortunately, several OSS systems used in libraries have become obsolete (Breeding, 2017; Gray, 2023). Rosen and Grogg (2021) found that project sustainability was a factor considered over financial advantages or user benefits when making decisions about OSS. Engard (2010) and Baker (2020) noted the importance of paying attention to an OSS project's community health and activity as it is linked to the project's sustainability. Involvement in OSS enables libraries to shape software development, collaborate with others, and stay competitive with proprietary options (Colt, 2023; Singh, 2020; Thacker et al., 2014).

The academic library community is advocating for OSS, emphasizing the need for financial and social contributions to sustain open-source digital infrastructure (Lewis, 2017; Martin, 2018; McIlwain, 2023; Skog, 2023). While some argue libraries do not contribute enough (Askey, 2008), research shows they do participate, although not universally (Thacker and Knutson, 2015). Thacker and Knutson (2015), in their survey completed by 77 ARL-member libraries, found that libraries contribute code/developer time, money, hosting, testing, and requirements to OSS projects used in libraries, mainly scholarly communications-related software like institutional repositories, digital preservation, and digital asset management software, and to a lesser extent to OSS not primarily used for scholarly communications (such as link resolvers, streaming media, and course reserves). OSS evaluation in academic libraries is still largely at the stage of comparing advertised features (Collister, 2023; Guimont et al., 2023; Open Society Institute, 2024; Thacker et al., 2014), which does not address how OSS differs from commercial software in that its users can affect its evolution by contributing resources, particularly to OSS communities.

Baker (2020) rated journal management OSS using QualiPSo's Open Maturity Model (OMM), which includes community evaluation questions about developer and user activity through a manual process. Colt (2023) explored automated methods for assessing library OSS, particularly communities, noting that, while GitHub provides access to some metrics, the Cauldron platform could be used to generate models from these metrics to indicate OSS project health and sustainability. While Baker (2020) and Colt (2023) introduced software engineering methods for evaluating OSS in libraries, this paper explores the software engineering literature for further tools and techniques, particularly focusing on evaluating OSS communities. The aim of this paper is to see if there is a way to easily identify opportunities for libraries to contribute to the health and growth of the open-source communities that build and maintain OSS critical for scholarly communications.

Literature Review

OSS Project Health and Sustainability

As organizations increasingly rely on OSS for digital infrastructure, they must plan for risk management and resource allocation. Monitoring OSS project health can help assess sustainability and guide resource investment (Germonprez et al., 2018). Link and Germonprez (2018) discovered through engaged scholarship and interviews that there was no single defined process for assessing OSS project health given the number of health-related factors, and then-current processes could be "unstructured, vague, subjective, and relying on intuition" (p. 3). The participants in their study struggled with interpreting meaning from metrics because context was so important for accurate interpretation. Goggins et al. (2021a) noted widespread "useful but haphazard assessments" (p. 106) for OSS as an impetus for creating standardized metrics in the Community Health Analytics in Open Source Software (CHAOSS) project. Techniques and metrics for assessing OSS project health are abundant. Linåker et al. (2022) proposed a framework with 107 health characteristics divided into three categories and 15 themes, distinguishing between project-level and network-level analyses.

Quantitative Analysis of Trace Data

OSS projects and project management from issue tracking to communications leave digital artifacts—trace data—that are visible and analyzable on both the open web (e.g., on GitHub) or by members of the project's community online. Trace ethnography, a methodology that examines documents and documentary traces, is suited to learning about distributed sociotechnical systems such as OSS communities (Geiger & Ribes, 2011). Trace data from repositories, issue trackers, and communication tools are usually what researchers analyze to ascertain OSS project health through network modelling or other statistical analysis techniques, including, more recently, machine learning. Although, some argue that trace data alone are insufficient (Geiger & Ribes, 2011; Germonprez et al., 2018; Goggins et al., 2021b; Link & Germonprez, 2018). Quantitative studies of OSS projects and communities which analyze trace data and networks are abundant in the software engineering literature (Almarimi et al., 2020, 2021, 2023; Aman et al., 2017; Çetin & Tüzün, 2022; Dey & Woods, 2022; Ferreira et al., 2019; McClean et al., 2021; Onoue et al., 2016; Oriol et al., 2023; Palomba & Tamburri, 2021; Raman et al., 2020; Tamburri et al., 2019c, 2021a, 2023; van Meijel, 2021; Voria et al., 2022; Yang et al., 2023).

Qualitative and Mixed Methods Research

Some studies combined qualitative and quantitative methods together, particularly to validate findings. For example, Avelino et al. (2016) used an algorithm on GitHub data to estimate the "truck factor" (which measures knowledge concentration and resilience to developer turnover) for software projects then checked findings by surveying developers involved in the projects. Tamburri et al. (2013a) used grounded theory with a systematic literature review to define a set of OSS communities. Based on these community definitions, Tamburri et al. (2019c) created YOSHI, a tool used to quantitatively analyze GitHub trace data and map a project to a community pattern. As part of the foundation of the CHAOSS project, multiple research methods were used to understand OSS communities (CHAOSS, n.d.-a; Germonprez et al., 2018; Goggins et al., 2021b). The results of this extensive multi-year mixed-methods research informed the creation of metrics and models for CHAOSS (CHAOSS, n. d.-c).

Importance of Community Analysis

While OSS project health may be monitored by analyzing everything from source code quality and complexity to software popularity and project activity, there is an area of research about community health: an analysis of the community members' communication and culture as well as the overall community structure, as an indicator of project sustainability.
Nagappan et al., in their influential 2008 paper, found that statistical models predicting software's proneness to failure were more accurate when these models included social and organizational metrics in addition to other technical software-oriented metrics. Bettenburg and Hassan (2010) also found that adding social data could increase the effectiveness of a software defect prediction model previously based

on only source code and process measures, showing that the traces of social interactions are valuable data for determining software health.

OSS development, given its reliance on collaboration, is inherently a dynamic socio-technical phenomenon (Ducheneaut, 2005). Cataldo et al. (2008) stated that a software development project is a socio-technical system, requiring congruence between the social and technical aspects for success, and found that their models were more accurate when they included measures of congruence between the technical aspects and social aspects. Palomba et al. (2018) determined that community dynamics impact source code quality issues. Tamburri et al. (2021b), in a systematic literature review to determine a grounded theory of success and failure factors for software engineering projects, determined that people- and process-related factors impact project sustainability.

Community Patterns

The structure and the resulting communication patterns of a community developing software is reflected in the software itself (Conway, 1968; Kwan et al., 2012). Different community structures, or community patterns, are linked with different effects on the project's health (Tamburri et al., 2023). Tamburri et al. (2013b) used a grounded theory approach to analyze software engineering research papers and found 13 different organizational social structures and divided them into metatypes of community, network, group, and team. Tamburri et al. (2019c) then created a systematic approach to identify community attributes in GitHub project data and classify project communities automatically into community patterns (defined as "sets of known organisational and social structure types and characteristics with measurable core attributes" (p. 1369) with a software they created called YOSHI (which stands for "Yielding Open-Source Health Information"). YOSHI, later updated by van Meijel (2021) who created YOSHI 2, examined six key characteristics: community structure, geodispersion, longevity, engagement, formality, and cohesion.

Community Health Affects Software Quality

There are two main lines of inquiry on OSS community health and its impact on software quality and sustainability. The first line of inquiry is about generating standard metrics and models about OSS projects and project ecosystems, including community health (Link & Germonprez, 2018; Goggins et al., 2021b). The other line of inquiry uses social network analysis and diagnoses community smells (i.e., types of organizational and social dysfunction within software projects) (Tamburri et al., 2015, 2021a) leading to the creation of multiple tools that detect community smells such as CodeFace4Smells (Tamburri et al., 2021a), Bus Factor Explorer (Klimov et al., 2023), Kaiaulu (Paradis et al., 2024; Paradis & Kazman, 2022), and ones that use machine learning such as csDetector (Almarimi et al., 2021) and CADOCS (Voria et al., 2022).

First Line of Inquiry: CHAOSS: Metrics and Models for Understanding Project (and Community) Health

The CHAOSS project is a current attempt to produce standard metrics and models to indicate project and community health. Informed by findings from multi-year engaged field studies (Germonprez et al., 2018; Goggins et al., 2021b), there are 89 CHAOSS metrics used in 17 metrics models using data about the project, including community-related data. CHAOSS (n.d.-c) metrics and metrics models, when visualized in dashboard services built with either GrimoireLab (Gonzalez-Barahona et al., 2022) or Augur are meant to inform project community managers and OSS program managers in organizations (Goggins et al., 2021b). GrimoireLab can use data from more than 30 sources; Augur uses data only from GitHub or

GitLab repositories or organizations, but Augur can import thousands of datasets from GitHub or GitLab repositories or organizations.

Colt (2023) analyzed DSpace SC-OSS using Cauldron (a dashboard for GrimoireLab) to generate CHAOSS Starter Project Health model metrics. However, not all metrics are available in Cauldron, and some would need to be measured qualitatively to complete a model. For example, the Community Welcomingness model contains 14 metrics, some of which can be gathered from trace data (e.g., whether there is a Code of Conduct in GitHub), but the Inclusive Leadership metric in this model requires qualitative research. In fact, qualitative data is required by multiple CHAOSS metrics (and models), increasing the time and work required.

Second Line of Inquiry: Community Smells: Social Network Analysis and Dysfunction Diagnosis

The second line of inquiry emerges from the concept of "social debt" in software engineering, defined as "the additional cost occurring when strained social and organisational interactions get in the way of smooth software development and operation" (Tamburri et al., 2015, p. 1). Given that OSS projects are socio-technical phenomena (Cataldo et al., 2008; Ducheneaut, 2005; Nagappan et al., 2008), the social, or community, health of OSS projects is a contributing factor to project sustainability. Tamburri et al. (2013a) posited that social debt within a software project is incurred by detrimental socio-technical decisions (e.g., decisions influencing both human and technical aspects). Social debt is important, though challenging, to identify and monitor but key to understanding project and community health.
Tamburri et al. (2015) defined a set of nine community smells by using an exploratory case study combined with grounded theory. They defined community smells as "sets of organisational and social circumstances with implicit causal relations . . . if reiterated over time cause social debt, in the form of mistrust, delays, uninformed or miscommunicated architectural decision-making," (p. 7). By providing a language for "hunches" about pervasive OSS project problems, Tamburri et al. (2015) provided a framework for naming the problems and mapped a set of mitigation strategies. Consequences of unaddressed community smells include employee turnover (Tamburri et al., 2015), bad software architecture decisions (Tamburri et al., 2019a), instability within the project's community, and even project collapse (Tamburri et al., 2020).

Tamburri et al. (2021a), in a mixed methods study, operationalized the detection of four community smells. They adapted an existing tool, CodeFace (Joblin et al., 2015), that identified communities and generated graphs of developer interactions and communications from GitHub and mailing list trace data. They created the network metrics and methods needed for detecting the community smells and made a new tool called CodeFace4Smells. Sixty OSS project communities were analyzed, with developers confirming the findings through surveys.

Once a set of community smells was defined by Tamburri et al. (2015), other research projects attempted to create software tools to automate the identification of these community smells. According to Paradis et al. (2024), these tools have two main tasks: first, to create graph representations using the trace data, and second, to compute the community smell metrics from these graphs.

Some projects address specific community challenges, such as the truck factor risk, which measures knowledge concentration and resilience to developer turnover (Avelino et al., 2016). This risk assessment is complicated due to the lack of standardized metrics (Ricca et al., 2011) and varying algorithm accuracy for determining truck factor and identifying key developers (Ferreira et al., 2019). Tools like those developed by Cosentino et al. (2015) and Klimov et al. (2023) help calculate and visualize the truck factor, while Çetin and Tüzün (2022) created an algorithm to identify key developers and assess knowledge distribution, considering various developer activities beyond code commits.

Palomba and Tamburri (2021) used the same set of 60 OSS project communities used by Tamburri et al. (2021a) but used a machine learning approach to predict community smells using metrics about socio-technical congruence, communicability, and turnover. Almarimi et al. (2020, 2021, 2023) also used a machine learning approach, including sentiment analysis, and built a software tool called csDetector that detects a project's community smells based on GitHub trace data. csDetector detects the most community smells of all the currently available tools. Community smells are explained in more detail in the Results section of this paper.

Relationship Between Community Patterns and Community Smells

De Stefano et al. (2020), in a preliminary study, used the YOSHI software to detect community patterns in 25 OSS project communities, then used CodeFace4Smells to extract a list of community smells and finally used machine learning (specifically, associated rule mining) to detect which community patterns and smells co-occurred. van Meijel (2021), expanding on De Stefano et al.'s (2020) work, adapted the YOSHI software in an attempt to find relationships between community patterns and community smells. For example, both studies found that "formal group" community patterns were linked with the "bottleneck" (when a single member of the community is frequently involved in community interactions) and "lone wolf effect" (when there are defiant contributors who work in isolation from other contributors) community smells. However, the generalizability of these studies is limited. De Stefano et al. (2020) relied solely on the Apriori machine learning algorithm and did not survey community members to confirm the findings. van Meijel (2021) surveyed community members to confirm the results of the YOSHI 2 analysis, but respondents disagreed with the results.

Aims

In the interest of helping libraries make evidence based decisions about open-source software (OSS), the objective of this research is to establish whether tools that automate the evaluation of OSS project communities could be used specifically on scholarly communications OSS (SC-OSS) projects to provide actionable insights for libraries to guide strategic decision making and corrective interventions.

Research Questions

RQ1: Which tools are appropriate for evaluating SC-OSS? What conclusions about community type and organizational characteristics can be made about these SC-OSS communities using existing OSS evaluation tools? Are these communities affected by any community smells? These questions are answered in the Literature Review and Results sections.

RQ2: The existing tools used in this study are producing results based on quantitative and qualitative criteria. What are the main assumptions, definitions, and dependencies used by these tools? These questions are answered in the Methodology and Results sections.

RQ3: Can such tools be useful for libraries to make strategic decisions related to their participation in SC-OSS communities? Answers will be provided in the Discussion section.

Methods

SC-OSS provides a unique opportunity for software evaluation, given that OSS is the primary type of software used in scholarly communications (CARL Open Repositories Working Group, 2022) and data generated during the SC-OSS project's lifespan are readily available. While multiple approaches and tools exist for assessing aspects of OSS projects, the aspects evaluated in this research paper are the community structure and community dysfunction (i.e., community smells). We initially identified two tools that can analyze OSS in the software engineering literature that were suitable for automated SC-OSS evaluation: YOSHI 2 for community types and csDetector for community smells. These two tools were used to generate and analyze datasets from 11 GitHub repositories representing three-month snapshots of community activity in seven SC-OSS projects. The results from YOSHI 2 did not provide any actionable insights, simply a characterization of the project's community type based on its members' interactions (e.g., most projects were communities of practice, and some were informal communities). Knowing a project's community type is a step toward gauging socio-technical congruence because it describes the project community's structure and dynamics (i.e., the social aspect) but without analyzing the technical aspect (e.g., software modularity and complexity, among others). However, some of the congruence measures used by Cataldo et al. (2008), namely geographic dispersion and patterns in communication, overlap with measures used to identify community smells. Finally, YOSHI 2 and csDetector results are too different to be complementary and any attempts to link specific community types to community smells and confirm findings qualitatively have not been completed (De Stefano et al., 2020; van Meijel, 2021). For this reason, YOSHI 2 data were not included in our results, but remain publicly available (Lasou & Neugebauer, 2024a).

The rationale for setting a maximum of three months' worth of data comes from Traag et al. (2013), as cited in Tamburri et al. (2019a), who determined that this timespan was a significant scale for effective analysis of community structures.

SC-OSS Project Selection

In libraries, OSS is used to support three main categories of scholarly communication services:

1. institutional repositories and research data repositories,
2. journal hosting, and
3. archiving.

This research paper focuses on SC-OSS used by academic libraries in Canada (see Table 1). The institutional repository software used in Canada is mainly DSpace (33%, n=31), EPrints (21%, n=19), and Islandora (18%, n=17) (CARL Open Repositories Working Group, 2022). Borealis (https://borealisdata.ca/), the Canadian research data repository, is based on Dataverse software. The most popular OSS used by Canadian academic libraries for journal hosting and publishing is Open Journal Systems (OJS) (Betz et al., 2023). Finally, OSS for archiving is less commonly used, but Archivematica is the major software used in this category (Barnes et al., 2022).

Table 1
SC-OSS Used by Academic Libraries in Canada and Selected for Evaluation

Provider	**Software**	**Service type**	**GitHub repository link**
Artefactual	Archivematica	Archiving	https://GitHub.com/artefactual/archivematica
University of Southampton	EPrints 3.4	Repository	https://GitHub.com/eprints/eprints3.4
Islandora Foundation	Islandora	Repository	https://GitHub.com/Islandora/islandora
LYRASIS	DSpace	Repository	https://GitHub.com/DSpace/DSpace
LYRASIS	DSpace-Angular	Repository	https://GitHub.com/DSpace/dspace-angular
LYRASIS	DSpace 7 REST Contract	Repository	https://GitHub.com/DSpace/RestContract
Samvera Community	Hyku, the Hydra-in-a-Box Repository Application	Repository	https://GitHub.com/samvera/hyku
Samvera Community	Hyrax: A Digital Repository Framework	Repository	https://GitHub.com/samvera/hyrax
Institute for Quantitative Social Science (IQSS)	Dataverse	Repository	https://GitHub.com/IQSS/dataverse
Public Knowledge Project (PKP)	PKP Documentation Hub	Journal hosting	https://GitHub.com/pkp/pkp-docs
Public Knowledge Project (PKP)	PKP Web Application Library	Journal hosting	https://GitHub.com/pkp/pkp-lib

To analyze SC-OSS communities, the focus of this research paper was to use existing tools that automatically collect and process data from GitHub rather than use frameworks that would require manual or qualitative processing and analysis (Andrade & Saraiva, 2017; Linåker et al., 2022). While all aspects of OSS projects may be evaluated at the project or network level, addressing the actors, software, or orchestration (Linåker et al., 2022), community evaluation, looking at the interaction patterns between project contributors, is the focus of this research paper since there are multiple human factors that librarians who may not be software developers can impact. The identification of community smells (i.e., community dysfunction) was of particular interest since naming a problem is the first step in its remediation, and, according to Labianca and Brass (2006), "negative relationships may have greater power than positive relationships to explain workplace outcomes" (p. 596).

Community Smell Detection

In their systematic review on community smells, Caballero-Espinosa et al. (2023) documented 30 different smells. Of those 30, nine were found to be widespread (Tamburri et al., 2015).

Several automated tools exist to identify community smells, and the number of community smells each tool can determine varies widely (see Table 2 below). For this research paper, csDetector was selected to do the community smell analysis since it detects the greatest number of community smells (up to 10). The developers of csDetector describe threats to the validity of the model that is used to detect community smells as well as the mitigation measures that they took (Almarimi et al., 2020). For example, the model training relies on a source of truth that is based on the authors' manual identification of community smells in 74 projects, which is prone to a certain degree of error. To mitigate, the authors relied on established definitions and guidelines, excluding projects for which there was no total agreement by the three authors (Almarimi et al., 2020). The authors also validated the model using bootstrapping statistical methods (a technique that repeatedly resamples data) to assess, among other measures, accuracy, precision, and recall for each smell, averaging at 0.86, 0.87, and 0.82, respectively (Almarimi et al., 2021).

Table 2
Number of Community Smells Detected by Existing Automated Tools in the Software Engineering Research Literature

Tool	**Community smells detected**	**Source code**	**Reference**
CodeFace4Smells	1. Organizational Silo 2. Black Cloud Effect 3. Lone Wolf Effect 4. Bottleneck or Radio Silence	https://github.com/maelstromdat/CodeFace4Smells/	(Tamburri et al., 2021a)
TruckFactor	1. Truck Factor or Bus Factor	https://GitHub.com/HelgeCPH/truckfactor	(Ferreira et al., 2019)
csDetector	1. Organizational Silo Effect 2. Black-Cloud Effect 3. Prima-Donnas Effect	https://GitHub.com/Nuri22/csDetector	(Almarimi et al., 2021)

	4. Sharing Villainy 5. Organizational Skirmish 6. Solution Defiance 7. Radio Silence 8. Truck Factor Smell 9. Unhealthy Interaction 10. Toxic Communication		
Kaiaulu	1. Organizational Silo 2. Missing Link 3. Radio Silence	https://GitHub.com/sailuh/kaiaulu	(Paradis & Kazman, 2022)

Results

Our goal is to explain how csDetector works, as well as the results it provides about community smells, to determine whether csDetector could be used to provide insights about SC-OSS project community characteristics, not to single out any SC-OSS project or community in particular.

csDetector

The csDetector software was built by Almarimi et al. (2021). It detects 10 community smells (see Table 2). To detect smells, it uses an extended genetic programming-based ensemble classifier chain to process exclusively GitHub data (Almarimi et al., 2023). csDetector analyzes the content of users' comments and can determine the tone (positive or negative). This process is called sentiment analysis and is conducted on issue comments, as well as commit and pull request comments. Results are anonymized as specified by Almarimi et al. (2023).

Running the Software

csDetector is open-source research prototype software written in Python, supported in a limited way by the development efforts of individual researchers (Nuri22, 2021). Other than an update to the licence and README file in 2024, the project's last code commits date back to 2021. The csDetector README file suggests that one way to run the software is to use the precompiled standalone executable file for Windows. Unfortunately, this method returned a runtime error; however, by using the second method described in the README file ("by the command line"), we could successfully run the software.[1] We followed the instructions for the installation of the requirements and executed the devNetwork Python script. We needed to modify the GitHub GraphQL API request settings in the Python code so that only 10 pull requests and issues at a time were requested, as 100 (the default setting) was causing the API to fail. One of the parameters is a GitHub authorization token that can be generated from GitHub, giving access

[1] https://GitHub.com/Nuri22/csDetector/blob/c91b4848231f10838991c14e5daf5611474dc364/README.md#2-by-the-command-line

to the API. csDetector also relies on ConvoKit[2] for parsing texts and politeness strategies used in conversations. ConvoKit, in turn, requires spaCy and its "en_core_web_sm".[3] SentiStrength is used by csDetector for classifying positive and negative sentiments in short texts. We used the models (one for each smell) that are supplied by default. Users can switch out the models by replacing files in a "models" directory.

We limited the data extraction from GitHub to a three-month period between May 3, 2023, and August 1, 2023, to limit the complexity of the analysis so that csDetector can complete within a reasonable amount of computation time for each repository. Another modification that was sometimes necessary was the default branch, which was hard coded in csDetector to "master," but in our case, many of the repositories were using "main."

The Samvera Hyku repository caused an issue in that, unexpectedly, it included a release with no authors/contributors (the GitHub API returns the value of "None"). The code needed to be modified so that this data would not break and interrupt the scripts with an error.

The two PKP repositories (Documentation Hub and Web Application Library), as well as the DSpace REST Contract, although not using releases, were processed without error. csDetector skipped 11 release-related metrics (such as NumberReleaseAuthors, NumberReleases, and ReleaseAuthorCount_count).

As DSpace REST Contract, PKP Documentation Hub, and Samvera Hyku do not use tags, three tag metrics were skipped for those repositories (TagCommitCount_count, TagCommitCount_mean, and TagCommitCount_stdev).

Community Smells

For each OSS assessed, csDetector results consist of a set of 190 metrics (e.g., number of authors, community centrality, pull request, commits and issue count, release count) grouped in nine different metric dimensions from which smells are inferred:

1. communication,
2. community,
3. developer contributions,
4. formality,
5. geographic dispersion,
6. sentiment analysis,
7. social network analysis,
8. truck number, and
9. community members.

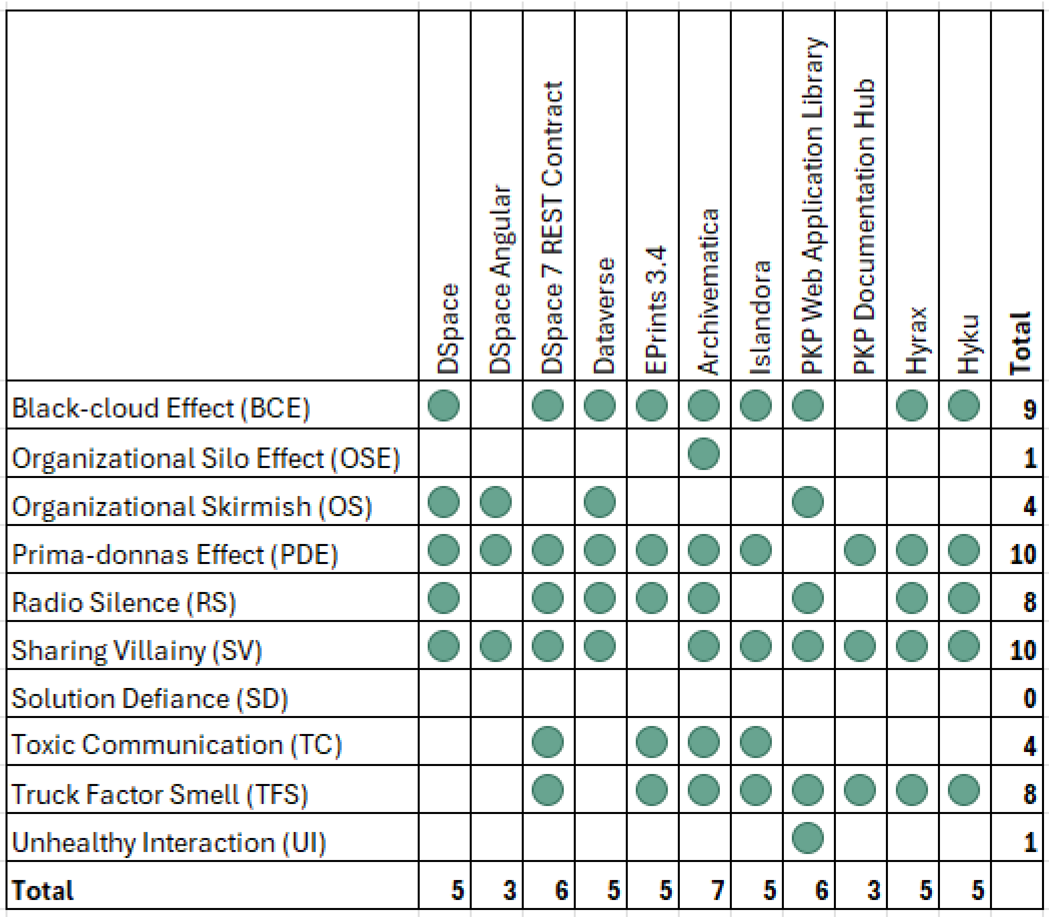
All the community smells and the respective repositories affected by them are shown in Figure 1.

csDetector also produces four graphs: 1) pull requests, 2) issues and pull request centrality, 3) issue graph, and 4) commit centrality, and a CSV file containing all metrics used. The community smells are

[2] https://convokit.cornell.edu/documentation/install.html
[3] From http://sentistrength.wlv.ac.uk/jkpop/, but we note that this URL is inaccessible in 2024.

only displayed directly on the command prompt windows. All results for metrics and graphs used for our studies have been publicly shared (Lasou & Neugebauer, 2024b).

One smell did not affect any SC-OSS communities: Solution Defiance. Its absence suggests that community members' backgrounds and cultural levels are homogeneous, thus avoiding division or conflicting opinion (Tamburri et al., 2015).



	DSpace	DSpace Angular	DSpace 7 REST Contract	Dataverse	EPrints 3.4	Archivematica	Islandora	PKP Web Application Library	PKP Documentation Hub	Hyrax	Hyku	**Total**
Black-cloud Effect (BCE)	●		●	●	●	●	●	●		●	●	**9**
Organizational Silo Effect (OSE)						●						**1**
Organizational Skirmish (OS)	●	●		●				●				**4**
Prima-donnas Effect (PDE)	●	●	●	●	●	●	●		●	●	●	**10**
Radio Silence (RS)	●		●	●	●	●		●		●	●	**8**
Sharing Villainy (SV)	●	●	●	●		●	●	●	●	●	●	**10**
Solution Defiance (SD)												**0**
Toxic Communication (TC)			●		●	●	●					**4**
Truck Factor Smell (TFS)			●		●	●	●	●	●	●	●	**8**
Unhealthy Interaction (UI)								●				**1**
Total	**5**	**3**	**6**	**5**	**5**	**7**	**5**	**6**	**3**	**5**	**5**	

Figure 1
Community smells in SC-OSS detected by csDetector using three months of GitHub activity data (May 3 to August 1, 2023). Community smells are explained in more detail below.

Two smells, Toxic Communication (TC) and Unhealthy Interaction (UI), strongly connected to the tone of the interactions between members affected some SC-OSS communities. They were detected by csDetector's sentiment analysis. Toxic Communication (TC) was diagnosed in four of the 11 communities. In communities affected by TC, communications are charged with negativity and there are many conflictual exchanges among members that can lead to developer stress or burnout (Almarimi et al., 2021). Only one community was affected by the Unhealthy Interaction (UI) smell (PKP Web Application Library). This smell occurs when communication exchanges between developers or community members are often very short, delayed, or spread over a long period of time (Raman et al., 2020).

Archivematica was the only community where the Organizational Silo Effect (OSE) smell was found. This smell signals communication problems and a high decoupling of tasks where members are isolated from one another and interaction among them seems limited. OSE can cause time waste, insignificant collaboration, and code duplication (Caballero-Espinosa et al., 2023).

The Organizational Skirmish (OS) smell affected four repositories. Communities facing OS deal with miscommunication or misunderstanding between members due to different expertise levels or organizational changes that can lead to delays (Tamburri et al., 2015).

Among the 10 smells, those affecting most SC-OSS communities were: Black-cloud Effect, (BCE), Prima-donnas Effect (PDE), Sharing Villainy (SV), Radio Silence (RS), and Truck Factor Smell (TFS).

Black-Cloud Effect (BCE)

Black-cloud Effect (BCE) was detected in nine of the 11 SC-OSS projects. BCE is strongly influenced by metrics from the social network analysis, developer contributions, and communication dimensions. The BCE emerges from a situation where community members have few opportunities to meet and share their experiences and, at the same time, few members can help close the knowledge gap between members. The resulting effects are that messages need to be repeated many times or are interpreted in the wrong way, also described as a "'black-cloud' of confusing back-and-forth messages [that] were constantly obfuscating reality" (Tamburri et al., 2015, p. 9).

Different strategies can be used to correct BCE smells. One of the most efficient solutions is to create an actively maintained and implemented communication plan to help structure communications between members (Catolino et al., 2020).

Prima-Donnas Effect (PDE)

The Prima-Donnas Effect (PDE) affected 10 of the 11 OSS communities. The most influential csDetector dimensions in PDE smells detection are social network analysis and sentiment analysis metrics. Communities affected by PDE will not readily welcome contributions or proposals from new members or other members. It is caused by inertia, both at the organizational level and code or software feature level, and may lead, for certain members, to non-inclusive behaviours such as condescension. PDE is also a sign of poor communication and collaboration practices among members (Caballero-Espinosa et al., 2023). Strategies to mitigate the PDE include community-based contingency planning, social wikis, and culture conveyor roles (Tamburri et al., 2015).

Sharing Villainy (SV)

Sharing Villainy (SV) was found in 10 of the 11 OSS communities. Like the BCE, it is caused by a lack of knowledge sharing opportunities (e.g., face-to-face meetings) causing the information shared among members to be outdated, unconfirmed, or wrong (Almarimi et al., 2023).

Mitigation strategies include the creation of a social wiki or ambassador roles to disseminate a homogeneous organizational culture (Tamburri et al., 2015).

Radio Silence (RS)

Radio Silence (RS) was detected in eight out of the 11 SC-OSS analyzed. csDetector's social network analysis dimension is the most influential for detecting RS. RS, also known as the bottleneck smell, reflects a formal organizational problem: the community is insufficiently structured and relies upon highly formal procedures maintained or controlled by few individual community members. For example, one member may interpose in every formal interaction and object to any initiatives to introduce changes (Tamburri et al., 2021a).

The most efficient strategies to resolve RS are to reinforce community cohesion by doing specific collective activities (e.g., brainstorming), practicing mentoring, and engaging with members to solve communication problems (Catolino et al., 2020).

Truck Factor Smell (TFS)

Eight out of the 11 SC-OSS analyzed were affected by the Truck Factor Smell (TFS). csDetector has a specific set of metrics to detect TFS based on issues, commits, and pull request interactions. TFS (also referred to as "bus factor") is an indicator used to determine the resiliency of a project if it encounters developer turnover, specifically, the number of software developers who would need to be "hit by a bus" to cause the project to stall (Jabrayilzade et al., 2022, p. 98).

TFS is a sign that knowledge critical to the project's sustainability is concentrated among few members. If there is developer turnover, knowledge will be lost, and the project will be at risk.

As part of its social network analysis, csDetector produces network graphs related to members' activities on GitHub. Figure 2 shows the Pull Request (PR) centrality graph for the EPrints software. From all the graphs generated as part of our studies, EPrints exemplifies TFS in the most unambiguous way. Any interactions on PR during the three-month period analyzed are centralized on only one member.

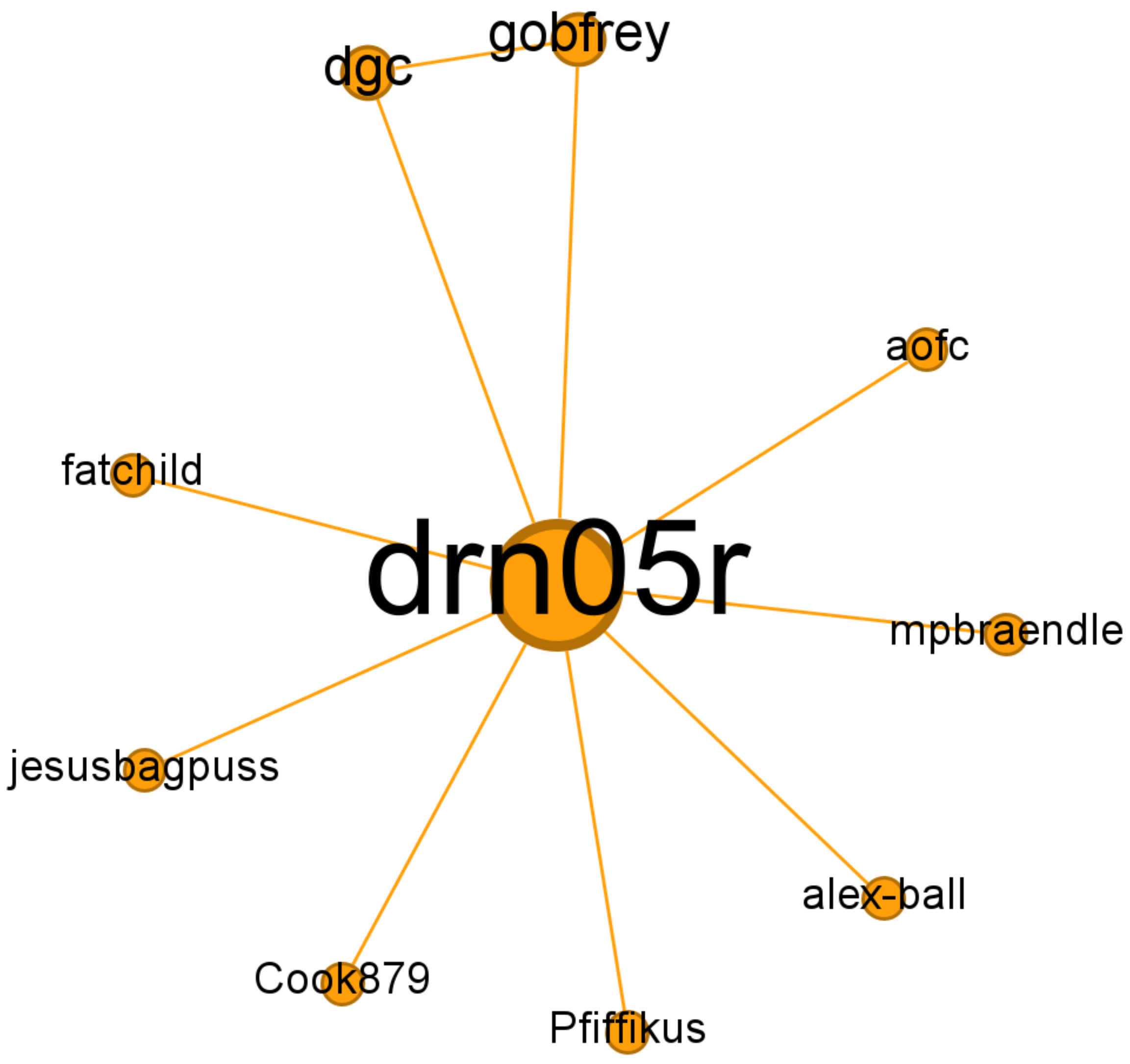


Figure 2
EPrints 3.4 csDetector pull request centrality graph visualization. The fact that all nodes connect to one central node demonstrates Truck Factor Smell (TFS). csDetector generates this data using GraphML File Format and a visualization of the graph in PDF. The above is the GraphML data from csDetector, visualized using Gephi - the Open Graph Viz Platform (Bastian, Heymann & Jacomy (2009).[4]

Truck factor, however, is a controversial indicator. Having too many developers involved in a project is also risky. The more people involved, the more communication paths required, adding complexity to communication patterns and potentially introducing miscommunication and resulting code problems (Nagappan et al., 2008). Avelino et al. (2016) created an approach to analyze 133 popular projects in

[4] https://github.com/gephi/gephi

GitHub to determine the truck factor for each, and later surveyed developers to confirm findings. Interestingly, they found that most systems had (in 2016) a truck factor of 1 (34%, n=45) or 2 (31%, n=42) including the very popular OSS projects, Clojure and Elasticsearch.
Overall, SC-OSS communities, from the three-month analysis period, seemed to have two main problems. The first problem was limited knowledge sharing either due to a lack of effective communication (BCE) or knowledge hoarding among members (SV). The second problem was weak collaboration practices possibly due to overly detailed and overwhelmingly formal procedures (RS) leading to few opportunities for onboarding newcomers and limiting the generation of new ideas (PDE).

Discussion

While the results generated by csDetector are interesting, this tool has limitations in the context of evaluating SC-OSS for academic libraries. Software development skills are required to run csDetector. Documentation is sparse. While some documentation about metrics is provided in an article about the first release (Almarimi et al., 2020, table 10), more than 50 metrics described in the GitHub repository (Nuri22, 2021) cannot be matched easily with the 190 metrics generated in the results. No information exists on the correlation of metrics and smells, nor the thresholds used.

Despite the challenges of using csDetector, this software can provide results worth monitoring, highlighting problems that are social in nature. The automated analysis of GitHub data and the use of metrics and machine learning models is also promising. The concept of community smells provides a useful vocabulary for defining OSS community dysfunctions and, therefore, solutions. For example, the Black Cloud Effect smell indicates weak user interaction and trouble with knowledge sharing and can be addressed by building a specific communication plan.

The data used and processed by csDetector opens a completely new dimension of indicators or characteristics. From the metrics used by csDetector, Almarimi et al. (2023) have emphasized that some metrics are more influential than others to detect community smells. The most significant in this regard are the “standard deviation of the number of developers per time zone and per community, and the social network betweenness, closeness and density centrality and the ratio of issues with negative sentiments, anger words and polite comments in PR [pull request] and issue discussions” (p. 2). Much of the data processed by tools like csDetector is originating from code sharing platforms. Some data helps determine how organized the community is (e.g., is the repository using milestones? Tags to flag issues?). Activities that circle around three GitHub features (issues, commits, and pull requests) can give a sense of engagement of community members and be used as criteria to measure OSS activities and vitality.

Many of the tools uncovered in our literature review, including csDetector, cannot be considered sustainable. In this regard, a project like CHAOSS (n. d.-a) looks much more promising. The community health indicators defined for this project were community based and the software used to produce the data, GrimoireLab, is open source (Gonzalez-Barahona et al., 2022).

The CHAOSS project has focused on building indicators to measure OSS community health and sustainability (CHAOSS, n. d.-a; Goggins et al., 2021b). However, like csDetector, it relies heavily on metrics based on software sharing platforms like GitHub. CHAOSS also contains more qualitative metrics that are similar in their objectives to csDetector sentiment analysis. For example, the CHAOSS, Psychological Safety, and Burnout models appear to align with two situations that widespread community smells, like Toxic Communication (TC), can cause within the communities we assessed. There is much overlap between CHAOSS metrics and csDetector, especially those focusing on community activities. However, CHAOSS also includes metrics beyond GitHub that are absent from csDetector: Documentation Accessibility, Discoverability and Usability. For all these reasons, CHAOSS seems to be more exhaustive for analyzing community health.

Using the CHAOSS Starter Project Health Model on the SC-OSS DSpace, designed to quickly assess the health of an OSS community, Colt (2023) demonstrated that metrics can be useful for validating or challenging perceptions about a project and for helping decision-makers determine if a specific OSS needs support.

With carefully selected and curated metrics, CHAOSS can generate lots of data. The real challenge is to determine the thresholds of each metric (e.g., how many days must an Issue Response Time be to be too much?) used to measure and determine specific OSS community health.

Limitations

The datasets and the methods used in our study have limitations. The repository selection was based on our experience as scholarly communication practitioners in Canada. There are others OSS used by libraries that would have been worth including, for example, for digital asset management: Omeka, or other journal publishing platforms such as Janeway. However, our goal was not to focus on a specific community but to determine the specificity of the results generated by csDetector.

Our assessment of OSS communities has been conducted over a brief period (three months). And we also voluntarily limited some configurations in part due to performance issues (i.e., limit the analysis to 100 comments per issue and pull requests). A longer analysis or temporal analysis (activities on a longer run or at different moments in time), sometimes pointed out as relevant to really have a sense of communities (Cánovas Izquierdo & Cabot, 2022; McClean et al., 2021), was out of the scope of our study.

The tools themselves have their own limitations. Only one specific GitHub branch can be processed. OSS can have multiple branches on which the community is active depending on software versions (i.e., OJS, EPrints). Moreover, only one GitHub repository at a time can be assessed, but a specific community can have multiple GitHub repositories (i.e., DSpace has three, two for developments DSpace and DSpace Angular and one for technical documentation for the REST API). This may prevent csDetector from providing a whole picture of a given community. In addition, the accuracy of tools such as csDetector are a function of the reliability of their natural language processing dependencies such as SentiStrength, ConvoKit, spaCy and its en_core_web model, in the specific context of GitHub data. Using these tools for analyzing issues and comments on GitHub can be effective, but there are specific considerations and potential limitations to keep in mind. GitHub comments and issues include domain-specific text and technical language, code snippets and domain jargon that might not be well represented in the training or lexical data of these tools. Although the authors of csDetector perform validation (accuracy, precision, recall, F1, AUC) of the training models used for smell detection resulting in relatively high average measures, the models ultimately rely on a source of truth that is based on the manual identification of community smells, which is prone to a certain degree of error. To mitigate, the authors of csDetector relied on inter-indexer reliability and established definitions and guidelines.

To predict community smells, csDetector relies only on GitHub data and do not use data from other platforms where non-programming activities occur (mailing lists, chat software like Slack or Mattermost, wikis). Cánovas Izquierdo and Cabot (2022) insist on the importance of non-programming activities in OSS communities, even suggesting that these activities could play a significant role for its sustainability. Based on GitHub data they define and analyze specific roles activities (developer, reviewer, merger, reporter, commenter, reactor). OSS community members tend to play multiple roles (Tatham, 2010). Community members intervene in various aspects of the OSS project they are involved in. The LYRASIS 2021 Open Source Software Report also highlighted that libraries are more inclined to allocate staff time for non-technical tasks (governance meetings, community feedback, or user testing) (Rosen & Grogg, 2021).

The analysis is also limited to the single project and does not consider the software dependencies or the network around the project. For example, thousands of websites and other software systems used OpenSSL, the ubiquitous OSS used for cryptography and secure communication. When the Heartbleed security bug in OpenSSL was discovered, it had a huge impact on OSS project health worldwide. This risk would not have been discovered through a trace data analysis of a single project’s GitHub repository and the community smells affecting OpenSSL (mainly, and arguably, truck factor) would not have been identified. Goggins et al. (2021a) argue that both the project and its ecosystem need to be analyzed to generate a complete picture of project health.

Linåker et al. (2022) concluded that it is important not to analyze an OSS project separate from its ecosystem and that a project’s software dependencies and links to other projects should be evaluated.

Also, analyzing GitHub trace data, although faster than a qualitative study, has been criticized by Goggins et al. (2021b) who stated that ongoing engagement with OSS projects over time provides better health and sustainability indicators. Geiger and Ribes (2011) argue that to understand trace data, researchers must be immersed in the group and actively investigate “otherwise backgrounded actors, software, and data” in tandem (p. 6).

Future Research

We did not validate the tools we used or seek to confirm their results with the scholarly communication communities we assessed. To investigate further SC-OSS communities, it would be relevant to conduct a qualitative analysis on each community to determine if concepts of community smells or community types are known and how they may be used to ensure OSS sustainability. This secondary qualitative analysis would serve as an additional test of the validity of the csDetector smell detection models, and potentially an expansion of its training dataset that would include software repositories specific to the scholarly communications domain.

A significant part of OSS communities' activities is not captured through GitHub and would need more research, especially to assess interactions occurring on communications tools such as mailing list and chat channels (i.e., Slack or Mattermost) and activities around documentation tools such as wikis. The community governance structure and rules (Who makes decisions? How do participants make their voices heard?) is also a critical aspect to understand to provide a complete picture of OSS communities (De Noni et al., 2011).

Conclusion

The automated evaluation of OSS communities using trace data is a promising area of development, and existing tools can be used in the context of a research project. These tools could also be used in combination with other evaluation methods to inform decision making.

The detection of community smells rather than community types seems more promising since it points out problems in the community and can help libraries make strategic decisions, whether it is to choose a specific OSS over another or to get involved in a community to try to address its problems (RQ3).

Community smells can be identified with csDetector, which was able to detect five widely spread smells in the SC-OSS communities targeted in this paper. The community smells identified when we evaluated SC-OSS with csDetector suggest that there may be inefficient knowledge sharing and weak collaboration practices that may lead to few opportunities for onboarding newcomers, in turn limiting the generation of new ideas (RQ1). csDetector cannot be considered as ready-to-use software; it is more akin to an open-source research prototype, with limited support. It has a major constraint given that it can analyze only one branch of a GitHub repository, resulting in only a partial analysis of a community's activities if the OSS is on multiple branches (RQ2). In this regard, a project like CHAOSS may be more promising. Our paper shows that csDetector can be used as an experimental software by library staff to evaluate SC-OSS communities. We recommend that library staff continue to monitor developments in this area to assess OSS communities as the tools become easier to use.

To assure long term development and flourishment of SC-OSS communities, libraries must participate in their communities, either by contributing code or through non-programming activities. Member participation, especially engagement, is vital to OSS communities' sustainability.

Simple awareness of common types of social and organizational dysfunctions in SC-OSS can help us understand what conditions are needed for software sustainability. The concept of "community smells" can, at a very minimum, provide us with a vocabulary for diagnosing dysfunctional dynamics and point us toward remedies. Libraries, along with other cultural and scientific heritage institutions, have the *It Takes a Village* project (Arp & Forbes, 2018), a collection of best practices and tools for supporting the sustainability of OSS used in these sectors. While remedies to community smells are not explicitly covered in the *It Takes a Village* guidebook, it offers information on aspects such as how to increase the number of contributors, which may address "truck factor," or the concentration of work among one or few people, under "Resources." In the "Community Engagement" section, it provides help on formulating a communications and engagement strategy/plan, which may address community smells such as "black-cloud effect" and "radio silence."

Without strategic support from libraries and library staff, any open scholarly communication infrastructure will remain an ideal that may not be reached. OSS communities are dynamic ecosystems and interactions can change with the flow of people leaving or onboarding it. csDetector, and similar tools, open a completely new dimension of OSS communities' characteristics to libraries. Interactions and communications problems, such as community smells, can lead to social debt, which in its turn will generate technical debt and, ultimately, jeopardize the sustainability of the project.

Author Contributions

Pierre Lasou: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Writing - original draft **Tomasz Neugebauer:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Writing - original draft **Pamela Carson:** Conceptualization, Writing - original draft, Writing - review & editing

Acknowledgements

Thank you to Francisco Berrizbeitia for his insights on machine learning.

References

Almarimi, N., Ouni, A., Chouchen, M., & Mkaouer, M. W. (2021). csDetector: An open source tool for community smells detection. In D. Spinellis, G. Gousios, M. Chechik, & M. Di Penta (Eds.), *ESEC/FSE 2021: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering,*1560–1564. https://doi.org/10.1145/3468264.3473121

Almarimi, N., Ouni, A., Chouchen, M., & Mkaouer, M. W. (2023). Improving the detection of community smells through socio-technical and sentiment analysis. *Journal of Software: Evolution and Process, 35*(6), Article e2505. https://doi.org/10.1002/smr.2505

Almarimi, N., Ouni, A., & Mkaouer, M. W. (2020). Learning to detect community smells in open source software projects. *Knowledge-Based Systems, 204,* Article 106201. https://doi.org/10.1016/j.knosys.2020.106201

Aman, H., Burhandenny, A. E., Amasaki, S., Yokogawa, T., & Kawahara, M. (2017). A health index of open source projects focusing on Pareto distribution of developer's contribution. *The 8th IEEE International Workshop on Empirical Software Engineering in Practice,* 29–34. https://doi.org/10.1109/IWESEP.2017.14

Andrade, S., & Saraiva, F. (2017). Principled evaluation of strengths and weaknesses in FLOSS communities: A systematic mixed methods maturity model approach. In F. Balaguer, R. Di Cosmo, A. Garrido, F. Kon, G. Robles, & S. Zacchiroli (Eds.), *Open source systems: Towards robust practices: 13th IFIP WG 2.13 International Conference, OSS 2017,* Proceedings, 34–46. https://doi.org/10.1007/978-3-319-57735-7

Arp, L. G., & Forbes, M. (2018). *It takes a village: Open source software sustainability.* LYRASIS. https://itav.lyrasis.org/guidebook/

Askey, D. (2008). We love open source software. No, you can't have our code. *Code4Lib, 5.* https://journal.code4lib.org/articles/527

Avelino, G., Passos, L., Hora, A., & Valente, M. T. (2016). A novel approach for estimating truck factors. *2016 IEEE 24th International Conference on Program Comprehension (ICPC),* 1–10. https://doi.org/10.1109/ICPC.2016.7503718

Baker, S. (2020). Assessing open source journal management software. *Journal of Electronic Publishing, 23*(1). https://doi.org/10.3998/3336451.0023.101

Barnes, M., Bell, E., Cole, G., Fry, J., Gatti, R., & Stone, G. (2022). *WP7 scoping report on archiving and preserving OA monographs (1.0)*. Zenodo. https://doi.org/10.5281/zenodo.6725309

Bastian, M., Heymann, S., Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*. https://gephi.org/users/publications/

Bettenburg, N., & Hassan, A. E. (2010). Studying the impact of social structures on software quality. *2010 IEEE 18th International Conference on Program Comprehension,* pp. 124-133. https://doi.org/10.1109/ICPC.2010.46

Betz, S., Nason, M., & Uhl, E. (2023). *Library publishing and hosting in Canada - Institutional responses to a 2022 questionnaire (Version 2)*. Borealis. https://doi.org/10.5683/SP3/SDFZUO

Bilder, G., Lin, J., & Neylon, C. (2020). *The principles of open scholarly infrastructure*. Retrieved on August 21, 2024, from https://doi.org/10.24343/C34W2H

Breeding, M. (2017). Kuali OLE (defunct). *Library Technology Reports, 53*(6), 25–26. https://journals.ala.org/index.php/ltr/article/view/6407/8456

Caballero-Espinosa, E., Carver, J. C., & Stowers, K. (2023). Community smells–the sources of social debt: A systematic literature review. *Information and Software Technology, 153,* 1–25. https://doi.org/10.1016/j.infsof.2022.107078

Cánovas Izquierdo, J. L., & Cabot, J. (2022). On the analysis of non-coding roles in open source development. *Empirical Software Engineering, 27*, 1–32. https://doi.org/10.1007/s10664-021-10061-x

CARL Open Repositories Working Group (2022). *Inventory of Canadian repository platforms.* Federated Research Data Repository. https://doi.org/10.20383/102.0538

Cataldo, M., Herbsleb, J. D., & Carley, K. M. (2008). Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and* Measurement, 2–11. https://doi.org/10.1145/1414004.1414008

Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., & Ferrucci, F. (2020). Refactoring community smells in the wild: The practitioner's field manual. *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS),* 25–34. https://doi.org/10.1145/3377815.3381380

Çetin, H. A., & Tüzün, E. (2022). Analyzing developer contributions using artifact traceability graphs. *Empirical Software Engineering, 27*, Article 77. https://doi.org/10.1007/s10664-022-10129-2

CHAOSS (n. d.-a). *About CHAOSS.* Retrieved on August 21, 2024, from https://chaoss.community/about-chaoss/

CHAOSS (n. d. -b). *CHAOSS software.* Retrieved on August 21, 2024, from https://chaoss.community/software/

CHAOSS (n. d.-c). *Metrics and metrics models.* Retrieved on August 21, 2024, from https://chaoss.community/kb-metrics-and-metrics-models/

CHAOSS (n. d.-d). *Practitioner guide: Introduction – things to think about when interpreting metrics.* Retrieved on August 21, 2024, from https://chaoss.community/practitioner-guide-introduction/

CHAOSS (n. d.-e). *Project engagement.* Retrieved on August 21, 2024, from https://chaoss.community/kb/metrics-model-project-engagement/

Chudnov, D. (1999). Open source software: The future of library systems? *Library Journal, 124*(13), 40–43.

Collister, L. (2023, December 8). Introducing Infra Finder. *Invest in Open Infrastructure.* Retrieved on April 3, 2024, from https://investinopen.org/blog/blog-introducing-infra-finder/

Colt, J. (2023). An introduction to using metrics to assess the health and sustainability of library open source software projects. *Code4Lib, 57.* https://journal.code4lib.org/articles/17514

Conway, M. E. (1968). How do committees invent? *Datamation, 14*(5), 28–31.

Corrado, E. M. (2005). The importance of open access, open source, and open standards for libraries. *Issues in Science and Technology Librarianship, 42*. https://doi.org/10.5062/F42F7KD8

Corrado, E. M. (2021). Revisiting the importance of open access, open source, and open standards for libraries. *Technical Services Quarterly, 38*(3), 282–292. https://doi.org/10.1080/07317131.2021.1934312

Corrado, E. M. (2023). Proprietary and open source software systems in libraries: A few considerations. *Technical Services Quarterly, 40*(3), 202–209. https://doi.org/10.1080/07317131.2023.2226434

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2015). Assessing the bus factor of Git repositories. *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER),* 499–503. https://doi.org/10.1109/SANER.2015.7081864

Cousijn, H., Hendricks, G., & Meadows, A. (2021). Why openness makes research infrastructure resilient. *Learned Publishing, 34*(1), 71–75. https://doi.org/10.1002/leap.1361

De Noni, I., Ganzaroli, A., & Orsi, L. (2011). The governance of open source software communities: An exploratory analysis. *Journal of Law and Governance, 6*(1), 1–18. https://vulj.vu.edu.au/index.php/jbsge/article/view/195/

De Stefano, M., Iannone, E., Pecorelli, F., & Tamburri, D. A. (2022). Impacts of software community patterns on process and product: An empirical study. *Science of Computer Programming, 214,* Article 102731. https://doi.org/10.1016/j.scico.2021.102731

De Stefano, M., Pecorelli, F., Tamburri, D. A., Palomba, F., & De Lucia, A. (2020). Splicing community patterns and smells: A preliminary study. *ICSEW '20: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 703–710. https://doi.org/10.1145/3387940.3392204

Dey, S., & Woods, W. (2022). LAGOON: An analysis tool for open source communities. *MSR '22: Proceedings of the 19th International Conference on Mining Software Repositories (MSR),* 717–721. https://doi.org/10.1145/3524842.3528504

Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work, 14*, 323–368. https://doi.org/10.1007/s10606-005-9000-1

Engard, N. C. (2010). *Practical open source software for libraries.* Chandos Publishing.

Ferreira, M., Mombach, T., Valente, M. T., & Ferreira, K. (2019). Algorithms for estimating truck factors: A comparative study. *Software Quality Journal, 27*, 1583–1617. https://doi.org/10.1007/s11219-019-09457-2

Geiger, R. S., & Ribes, D. (2011). Trace ethnography: Following coordination through documentary practices. *44th Hawaii International Conference on System Sciences,* 1–10. https://doi.org/10.1109/HICSS.2011.455

Germonprez, M., Link, G. J. P., Lumbard, K., & Goggins, S. (2018). Eight observations and 24 research questions about open source projects: Illuminating new realities. In K. Karahalios, A. Monroy-Hernández, A. Lampinen, & G. Fitzpatrick (Eds.), *Proceedings of the ACM on Human-Computer Interaction, 2*(CSCW), 1–22. https://doi.org/10.1145/3274326

Goggins, S. P., Germonprez, M., & Lumbard, K. (2021a). Making open source project health transparent. *Computer, 54*(8), 104–111. https://doi.org/10.1109/MC.2021.3084015

Goggins, S., Lumbard, K., & Germonprez, M. (2021b). Open source community health: Analytical metrics and their corresponding narratives. *IEEE/ACM 4th International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal),* 25–33. https://doi.org/10.1109/SoHeal52568.2021.00010

Goggins, S. P., Mascaro, C., & Valetto, G. (2013). Group informatics: A methodological approach and ontology for sociotechnical group research. *Journal of the American Society for Information Science and Technology, 64*(3), 516–539. https://doi.org/10.1002/asi.22802

Gonzalez-Barahona, J. M., Izquierdo-Cortázar, D., & Robles, G. (2022). Software development metrics with a purpose. *Computer, 55*(4), 66–73. https://doi.org/10.1109/MC.2022.3145680

Government of Canada. (2021). *Digital nations charter*. Retrieved on August 21, 2024 from https://www.canada.ca/en/government/system/digital-government/improving-digital-services/digital-nations-charter.html

Gray, K. L. (2023). Breathing life into Archon: A case study in working with an unsupported system. *Code4Lib, 57*. https://journal.code4lib.org/articles/17509

Guimont, C., Vaughn, M., & Ball, C. E. (2023). Finding the right platform: A crosswalk of academy-owned and open-source digital publishing platforms. *Knowledge Commons.* https://doi.org/10.17613/z27e-0z11

Jabrayilzade, E., Evtikhiev, M., Tüzün, E., & Kovalenko, V. (2022). Bus factor in practice. *ICSE-SEIP '22: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice,* 97–106. https://doi.org/10.1145/3510457.3513082

Joblin, M., Mauerer, W., Apel, S., Siegmund, J., & Riehle, D. (2015). From developer networks to verified communities: A fine-grained approach. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering,* 563–573. https://doi.org/10.1109/ICSE.2015.73

Klimov, E., Ahmed, M. U., Sviridov, N., Derakhshanfar, P., Tüzün, E., & Kovalenko, V. (2023). Bus Factor Explorer. *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE),* 2018–2021. https://doi.org/10.1109/ASE56229.2023.00015

Kwan, I., Cataldo, M., & Damian, D. (2012). Conway's law revisited: The evidence for a task-based perspective. *IEEE Software, 29*(1), 90–93. https://doi.org/10.1109/MS.2012.3

Labianca, G., & Brass, D. J. (2006). Exploring the social ledger: Negative relationships and negative asymmetry in social networks in organizations. *The Academy of Management Review, 31*(3), 593–614. https://doi.org/10.5465/amr.2006.21318920

Lasou, P., & Neugebauer, T. (2024a). Community patterns of scholarly communication open source software generated by YOSHI [dataset]. *Borealis*. https://doi.org/10.5683/SP3/4MEDXO

Lasou, P., & Neugebauer, T. (2024b). Community smells detection on scholarly communication open source software using csDetector [dataset]. *Borealis*. https://doi.org/10.5683/SP3/34MYPI

Lease Morgan, E. (2002). Possibilities for open source software in libraries. *Information Technology and Libraries, 21*(1), 12–15.

Lewis, D. W. (2017). *The 2.5% commitment* [working paper]. Retrieved on August 21, 2024, from http://doi.org/10.7912/C2JD29

Linåker, J., Papatheocharous, E., & Olsson, T. (2022). How to characterize the health of an open source software project? A snowball literature review of an emerging practice. *OpenSym 2022, Proceedings of the 18th International Symposium on Open Collaboration*, 1–12. https://doi.org/10.1145/3555051.3555067

Link, G. J. P., & Germonprez, M. (2018). Assessing open source project health. *24th Americas Conference on Information Systems 2018. AMCIS Proceedings 5,* 1–5.

Lumbard, K., Germonprez, M., & Goggins, S. (2024). An empirical investigation of social comparison and open source community health. *Information Systems Journal, 34*(2), 499–532. https://doi.org/10.1111/isj.12485

Martin, S. (2018). *The 2.5% commitment: 2.5% of whom?* [Presentation]. Retrieved on August 21, 2024, from https://hdl.handle.net/2022/22548

Martin, V. (2022). *The complete guide to open scholarship.* Bloomsbury Publishing.

McClean, K., Greer, D., & Jurek-Loughrey, A. (2021). Social network analysis of open source software: A review and categorisation. *Information and Software Technology, 130,* Article 106442. https://doi.org/10.1016/j.infsof.2020.106442

McIlwain, J. R. (2023). Towards an open source-first praxis in libraries. *Information Technology and Libraries*, *42*(4), 1–21. https://doi.org/10.5860/ital.v42i4.16025

Nagappan, N., Murphy, B., & Basili, V. (2008). The influence of organizational structure on software quality: An empirical case study. *ICSE '08: Proceedings of the 30th International Conference on Software Engineering*, 521–530. https://doi.org/10.1145/1368088.1368160

Nelson, A. (2022). *Ensuring free, immediate, and equitable access to federally funded research.* [Memorandum]. Office of Science and Technology Policy, Washington, D.C. Retrieved on August 21, 2024, from https://rosap.ntl.bts.gov/view/dot/65799/dot_65799_DS1.pdf

Nuri22. (2021). csDetector [Source Code]. https://github.com/Nuri22/csDetector

Onoue, S., Hata, H., Monden, A., & Matsumoto, K. (2016). Investigating and projecting population structures in open source software projects: A case study of projects in GitHub. *IEICE Transactions on Information and Systems, E99.D*(5), 1304–1315. https://doi.org/10.1587/transinf.2015EDP7363

Open Society Institute. (2004). *Guide to institutional repository software* (3rd ed.) [PDF]. https://www.budapestopenaccessinitiative.org/resources/guide-to-institutional-repository-software/

OpenDOAR. (2024, May). *OpenDOAR statistics.* Retrieved on May 19, 2024, from https://v2.sherpa.ac.uk/view/repository_visualisations/1.html

Oriol, M., Müller, C., Marco, J., Fernandez, P., Franch, X., & Ruiz-Cortés, A. (2023). Comprehensive assessment of open source software ecosystem health. *Internet of Things, 22*, Article 100808. https://doi.org/10.1016/j.iot.2023.100808

Palomba, F., Tamburri, D. A., Serebrenik, A., Zaidman, A., Fontana, F. A., & Oliveto, R. (2018). How do community smells influence code smells? *Proceedings of the 40th International Conference on Software Engineering,* 240–241. https://doi.org/10.1145/3183440.3194950

Palomba, F., & Tamburri, D. A. (2021). Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. *The Journal of Systems and Software, 171,* Article 110847. https://doi.org/10.1016/j.jss.2020.110847

Paradis, C., & Kazman, R. (2022). Building the MSR tool Kaiaulu: Design principles and experiences. In P. Scandurra, M. Galster, R. Mirandola, & D. Weyns (Eds.), *Software Architecture. ECSA 2021. Lecture Notes in Computer Science, 107–129*. https://doi.org/10.1007/978-3-031-15116-3_6

Paradis, C., Kazman, R., & Tamburri, D. (2024). Analyzing the Tower of Babel with Kaiaulu. *The Journal of Systems & Software, 210,* Article 111967. https://doi.org/10.1016/j.jss.2024.111967

Qiu, H. S., Lieb, A., Chou, J., Carneal, M., Mok, J., Amspoker, E., Vasilescu, B., & Dabbish, L. (2023). Climate Coach: A dashboard for open-source maintainers to overview community dynamics. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* https://doi.org/10.1145/3544548.3581317

Raman, N., Cao, M., Tsvetkov, Y., Kästner, C., & Vasilescu, B. (2020). Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. *ICSE-NIER '20: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results,*57–60. https://doi.org/10.1145/3377816.3381732

Rhyno, A. (2004). *Using open source systems for digital libraries.* Libraries Unlimited.

Ricca, F., Marchetto, A., & Torchiano, M. (2011). On the difficulty of computing the truck factor. In D. Caivano, M. Oivo, M. T. Baldassarre, & G. Visaggio (Eds.), *Product-focused software process improvement* (Vol. 6759, p. 337-351). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-21843-9_26

Rosen, H., & Grogg, J. (2021, August). *LYRASIS 2021 open source software survey report: Understanding the landscape of open source software in American libraries.* LYRASIS. https://doi.org/10.48609/n35f-5828

Singh, V. (2020). Applying participatory action approach to integrating professional librarians into open source software communities. *Journal of Librarianship and Information Science, 52*(2), 541–548. https://doi.org/10.1177/0961000619836724

Skog, A. (2023). Strategies for collaboration: ICOLC Open Collaboration in Library Consortia recommendations. *Journal of Library Administration, 63*(1), 101–110. https://doi.org/10.1080/01930826.2022.2146443

Tamburri, D. A., Blincoe, K., Palomba, F., & Kazman, F. (2020). "The canary in the coal mine…" A cautionary tale from the decline of SourceForge. *Software: Practice and Experience, 50*(10), 1930–1951. https://doi.org/10.1002/spe.2874

Tamburri, D. A., Di Nucci, D., Di Giacomo, L., & Palomba, F. (2019b) Omniscient DevOps analytics. In J. M. Bruel, M. Mazzara, B. Meyer (Eds.), *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018*, 48–59. https://doi.org/10.1007/978-3-030-06019-0_4

Tamburri, D., Kazman, R., & Fahimi, H. (2023). On the relationship between organizational structure patterns and architecture in Agile teams. *IEEE Transactions on Software Engineering, 49*(1), 325–347. https://doi.org/10.1109/TSE.2022.3150415

Tamburri, D. A., Kazman, R., Van Den Heuvel, W-J. (2019a). Splicing community and software architecture smells in Agile teams: An industrial study. In T. X. Bui (Ed.), *Proceedings of the 52nd Annual Hawaii International Conference on System Sciences*, 7037–7047. https://hdl.handle.net/10125/60140

Tamburri, D. A., Kruchten, P., Lago, P., & van Vliet, H. (2013a). What is social debt in software engineering? *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 93–96. https://doi.org/10.1109/CHASE.2013.6614739

Tamburri, D. A., Kruchten, P., Lago, P., & van Vliet, H. (2015). Social debt in software engineering: Insights from industry. *Journal of Internet Services and Applications, 6,* 1–17. https://doi.org/10.1186/s13174-015-0024-6

Tamburri, D. A., Lago, P., & van Vliet, H. (2013b). Organizational social structures for software engineering. *ACM Computing Surveys, 46*(1), 1–35. https://doi.org/10.1145/2522968.2522971

Tamburri, D. A., Palomba, F., & Kazman, R. (2021a). Exploring community smells in open-source: An automated approach. *IEEE Transactions on Software Engineering, 47*(3), 630–652. https://doi.org/10.1109/TSE.2019.2901490

Tamburri, D. A., Palomba, F., & Kazman, R. (2021b). Success and failure in software engineering: A followup systematic literature review. *IEEE Transactions on Engineering Management, 68*(2), 599–611. https://doi.org/10.1109/TEM.2020.2976642

Tamburri, D. A., Palomba, F., Serebrenik, A., & Zaidman, A. (2019c). Discovering community patterns in open-source: A systematic approach and its evaluation. *Empirical Software Engineering, 24*, 1369–1417. https://doi.org/10.1007/s10664-018-9659-9

Tamburri, D. A. (2024). YOSHI 2 [Source Code]. https://github.com/maelstromdat/YOSHI/tree/master/yoshi2

Tatham, E. (2010, November 23). Roles in open source projects. *OSS Watch.* http://oss-watch.ac.uk/resources/rolesinopensource

Thacker, J. C., Knutson, C. D., & Dehmlow, M. (2014, July). *Open source software.* SPEC Kit 340. Association of Research Libraries.

Thacker, C., & Knutson, C. (2015). Barriers to initiation of open source software projects in libraries. *Code4Lib, 29*. https://journal.code4lib.org/articles/10665

Traag, V. A., Krings, G., & Van Dooren, P. (2013). Significant scales in community structure. *Scientific Reports, 3*, Article 2930. https://doi.org/10.1038/srep02930 van Meijel, J. (2021). On the relations between community patterns and smells in open-source: A taxonomic and empirical analysis [Master's thesis, Eindhoven University of Technology]. https://research.tue.nl/en/studentTheses/on-the-relations-between-community-patterns-and-smells-in-open-so

Voria, G., Pentangelo, V., Della Porta, A., Lambiase, S., Catolino, G., Palomba, F., & Ferrucci, F. (2022). Community smell detection and refactoring in SLACK: The CADOCS project. *Proceedings 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME),* 469–473. https://doi.org/10.1109/ICSME55016.2022.00061

Wenger, E., McDermott, R. A., & Snyder, W. (2002). *Cultivating communities of practice: A guide to managing knowledge.* Harvard Business School Publishing.

Yang, R., Yang, Y., Shen, Y., Sun, H. (2023). An approach to assessing the health of opensource software ecosystems. In Y. Sun, T. Lu, Y. Guo, X. Song, H. Fan, D. Liu, L. Gao, & B. Du. (Eds.), *Computer Supported Cooperative Work and Social Computing. 17th CCF Conference, ChineseCSCW 2022. Communications in Computer and Information Science,* 465–480. https://doi.org/10.1007/978-981-99-2356-4_37