The Future of Software Transparency: Bridging Understanding, Measurement, and Practice

Gregorio Dalia*, Annibale Panichella†, Andrea Di Sorbo*, Gerardo Canfora*, Corrado A. Visaggio‡
*University of Sannio, Italy
†Delft University of Technology, The Netherlands
‡University of Foggia, Italy
g.dalia@studenti.unisannio.it, A.Panichella@tudelft.nl, {disorbo, canfora}@unisannio.it, corrado.visaggio@unifg.it

***Abstract*—Although the study of software transparency has deep roots in software engineering, a shared definition and practical application in real-world development contexts remain elusive. Through an in-depth analysis of the academic and industrial landscape, this article provides an overview of the current state of knowledge on software transparency, outlining a path to a deeper understanding of the subject for both developers and researchers. The challenge of software transparency involves not only establishing a formal, widely accepted understanding within the community, but also measuring and quantifying it in production environments. To this end, we survey academics and developers to evaluate an innovative approach to defining transparency and present a vision of a new framework for its quantification.**

I. Introduction

Software is everywhere, powering the economy and running health systems. However, users rarely know what happens to their data, and developers seldom disclose how their products actually work. This opacity introduces risks, including algorithmic bias, security vulnerabilities, and eroded trust, especially in critical domains such as healthcare, finance, or autonomous vehicles, where opaque behavior can undermine trust or even cause harm. Therefore, greater transparency would foster safer adoption, responsible innovation, and public trust in technology. Transparency is a widely discussed concept in many disciplinary contexts, and its etymology, from the Latin *trans-parere*, "to see through", well conveys the idea of accessibility and clarity. In the field of software engineering (SE), transparency has progressively assumed the status of a non-functional requirement [26], placing itself fully among the desirable properties for reliable, responsible and secure systems. However, software transparency remains a partially unexplored concept in the scientific literature [25]. Indeed, transparency is an aspiration rather than a concrete, measurable goal. Our work stems from the observation of a growing and urgent demand for transparency, perceived not only by the academic community, but also by regulatory and industrial contexts. Institutional efforts such as the EU Cyber Resilience Act [7] and the US Executive Order on cyber security [3] closely linked transparency to the security of the supply chain and trust in the entire software ecosystem. In the academic world, however, the formalization of transparency has produced theoretical models [26], [20], whose practical applicability has been limited, especially w.r.t. measurement. These models typically conceptualize transparency as a non-functional requirement, laying formal foundations within requirement engineering and identifying intersections with usability, reliability, and security, offering a first conceptual framework of reference. While these efforts provide valuable insights, they fall short of equipping practitioners with tools to design and assess real-world systems transparency.

This contribution focuses on the transparency of the software product, deliberately excluding the dimension related to the development process, which has been studied more extensively in the literature [21]. This focus on the product dimension is motivated by its relative under-exploration and lack of specific conceptual and operational tools. Drawing from the original definition of transparency, we argue that software transparency should encompass both structural visibility (understanding of the architecture and components of the system) and behavioral intelligibility (understanding of the actions of the systems at runtime).

Our work has a two-fold objective. On the one hand, we want to provide an extended and more operational vision of software transparency, capable of addressing the concrete needs of developers and integrators. On the other hand, we directly tackle the open challenge of measuring transparency, responding to the needs that emerged from the literature [12] that criticize the current reliance on indirect proxies and abstract indicators. Our proposal introduces a new conceptual framework that integrates, to the current ones, structural and behavioral dimensions, and is designed for practical applicability. We have preliminarily evaluated this framework through a survey involving academic and industrial stakeholders. The results reveal a high level of interest and confirm that our direction is perceived as promising. However, we acknowledge that our contribution is a first step toward building a new generation of tools for software transparency governance, as our framework is not yet definitive, requiring further validation, field experiments, and refinement, which can be ensured through close collaboration between academia and industry.

II. Background and motivation

Software transparency is a deeply rooted topic in the software engineering literature [19], and despite a growing interest in recent years [25], it continues to represent an open challenge, both conceptually and operationally, especially since it

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00325

is rarely treated as a primary and measurable attribute of software. Since the first theoretical reflections, it has emerged that transparency is not limited to access to the source code, in fact, as Camp observed [4], making the code public does not imply that its behavior is interpretable or that its structure is clear. One of the main pioneering contributions in this area is the SIG (Softgoal Interdependency Graph) framework proposed by Leite et al. [26], which represents a first structured attempt to model transparency through five dimensions: accessibility, usability, informativeness, understandability and verifiability. The SIG has been adopted and cited in numerous subsequent studies [34], [10], [33], [18], [22], becoming the reference point for software transparency. However, its theoretical nature, combined with the presence of 32 components and subcomponents, has limited its adoption in production.

Pfleeger [23] extended the theme of transparency with the idea of recognizing transparency as a point of intersection between cybersecurity and software engineering, introducing the concept of "undesirable code." This term refers to software behaviors that, while not directly violating formal security or privacy requirements, are problematic or ambiguous with respect to user expectations. A notable example is the Volkswagen emissions scandal [30], which demonstrated that formal compliance does not equate to transparency, and highlighted the need for more interpretable and accountable software systems. This perspective is further strengthened by the concept of "equivocal behavior" [8], according to which a software can exhibit ambiguous behaviors that are not directly harmful but that become critical if not correctly communicated and interpreted. Transparency, therefore, does not only concern the structure of the software but also the way in which this structure is documented and transmitted.

The systematic analysis of the literature carried out by Ofem et al. [20] attempted to systematize this field, identifying 106 factors associated with transparency, then synthesized into 21 to make them more suitable for practical application. Despite this effort at simplification, their operationalization remains complex, and adoption in industrial contexts is limited. This is consistent with the observations of Brennan et al. [2], according to which transparency tends to be treated as a derivative concept, subordinated to other objectives, and rarely as a stand-alone metric. Zacarias et al. [33] investigated the target of software transparency, showing how the concept should address both end users, to improve the conscious use of software, and developers, who must evaluate the quality and reliability of third-party software components. This dual perspective implies that transparency should be communicable and measurable along the entire software supply chain.

The central challenge motivating our work is not only how transparency is defined, but how it can be measured. As Tu et al. [31], [32] highlight, existing metrics are often insufficient or impractical for real-world use. Several proposals, such as Portugal et al .[25], using the "GQO" strategy, and Spagnuelo et al .[28], based on IEEE 1061, define metrics, but rely on complex, non-scalable manual evaluations. Pfleeger [23] observed that even automatic tools are inadequate to detect ambiguous or malicious behaviors that do not break explicit rules but elude the system's intention. In support of this claim, Schell's study on the Multics system [27] demonstrates how a deliberately inserted backdoor eluded detection by experts despite manual analysis and full access to the code.

The work of Isong et al. [12] represents one of the most recent efforts to address the problem of measuring transparency in a pragmatic way. The framework focuses on accessibility, usability, understandability, modifiability and reusability, but explicitly acknowledges that it does not cover the entire spectrum of transparency. Finally, Ofem [21] further systematized the existing methods, highlighting how most approaches rely on interviews, controlled experiments and theoretical models, with little empirical validation.

In summary, two main challenges emerge: (i) the lack of a shared and operationally usable framework that allows to measure software transparency in real environments; and (ii) the inadequacy of current approaches to support concrete decisions in the industrial field and in the software supply chain. These evidences motivate our proposal: the vision of a new framework for measuring software transparency that bridges methodological fragmentation and supports both scientific and industrial applications.

III. Vision on future Framework

As highlighted by Zacarias et al. [33], software transparency must serve a dual purpose: on the one hand, providing end-users with a clear and accessible understanding of software behavior; on the other hand, providing developers with effective tools to assess the quality and reliability of third-party components, especially in the context of integration and supply chain security. In parallel, studies such as those by Hosseini et al. [11] and Leite and Cappelli [26] have emphasized the time and resource costs of implementing transparency, reinforcing the need for automatable and sustainable solutions. While previous efforts have focused on dimensions such as accessibility and informational power, their heavy reliance on manual evaluations limits their scalability. These limitations highlight the urgency of a paradigm shift: a framework is needed that focuses directly on the software, its construction and behavior, without requiring explicit involvement of developers or relying on information extracted from production processes. This is the basis of the approach we propose in this section, where we outline an innovative vision of a framework for measuring software transparency, designed to overcome current constraints of scalability, subjectivity, and incompleteness.

To effectively address the challenges outlined above, we hypothesized an approach aimed at integrating and extending existing frameworks present in the literature, enriching them with new dimensions that explicitly consider both the structural and behavioral aspects of the software. On the structural side, a proposal widely recognized by both the scientific community and international regulatory bodies is represented by the use of software bills of materials (SBOM). An SBOM consists of a complete and detailed inventory of

all the software components used in the construction of an application or system. In recent years, SBOMs have acquired increasing attention [9], [29], [1], also by virtue of institutional initiatives that encourage, or even impose, their adoption [7], [3]. However, despite their potential, the large-scale adoption of SBOMs is still hampered by several critical issues. Among them are the lack of mature tools for their generation, especially in contexts where the inventory has to be built *a posteriori*, and the difficulty of integrating these tools into existing industrial workflows [9]. These limitations confirm that SBOMs, while representing a promising structural basis for software transparency, are still a topic of strong interest for both academic research and industrial innovation.

In contrast to structural analysis, behavioral transparency remains underexplored. Recent work [6] has shown that, for both developers, who may use software components in their products, and end users, the system often remains a black box that does not adequately inform them about all the actions it performs for its operation. The problem of understanding software behavior has very deep origins [5], introducing the challenge of dynamic analysis, which involves inspecting execution traces to understand software interactions, input handling, and responses. As noted by Dalia et al. [8], the same behavior, such as the use of a specific encryption method or access to certain system resources or information, may be perfectly acceptable in one context but gray and inadvisable in another. Hence, the common denominator should be awareness, as the same software could be used in different contexts.

To ground our proposal in concrete needs and stakeholder perspectives, we conducted a survey, designed according to relevant guidelines in the software engineering domain [24], [16], [17], [13], [14], [15], involving both academics and industry professionals (developers, and experts in security and software engineering). The goal was to assess the relevance of the dimensions considered by the current framework and our proposed extension. After refining the survey questions based on the feedback of our research team and PhD students in SE, we invited selected participants from academia and industry to complete the online questionnaire. Since adequate knowledge about the survey topics was required, the participants were selected based on their work and research activities.

The survey involved 23 experts, almost evenly split between academia (47.8%) and the industrial sector (52.2%). Among the participants of the latter group, 50% identified themselves as "Professional Developer", while the remainder held highly specialized technical roles, such as "Security Engineer" and "SOC Analyst". The questionnaire, structured according to a 5-point Likert scale (1 = minimum value, 5 = maximum value), allowed us to collect information on the level of competence of the interviewees: approximately 70% declared a high qualification in software development (value ≥ 4), while 65% indicated an equivalent level in the IT security domain. In the first section of the survey, the participants were asked to evaluate the currently prevailing approach for measuring software transparency, based on five fundamental dimensions: *accessibility* (i.e., how easy it is to access software-related information and artifacts), *usability* (i.e., how easy and effective it is to use the available information and tools), *understandability* (i.e., how well the documentation, behavior, and logic can be understood), *modifiability* (i.e., how easily the software or its components can be changed), and *reusability* (i.e., how easily components or resources can be reused elsewhere). We presented a scenario in which these dimensions had already been analyzed and made available through a qualitative numerical estimate on a scale of 1–10. 65.2% of the participants expressed a positive evaluation (value $\geq$ 4), confirming how the current framework is well accepted by experts. Nevertheless, some critical issues were raised. Among these were the absence of information regarding the maintenance status and versioning of components, a lack of data concerning the supplier's reputation, and operational difficulties in concretely evaluating the proposed dimensions, which were often considered too generic or not directly measurable. Additionally, some participants noted the lack of explicit safety-related indicators and the absence of a specific dimension dedicated to auditability. The survey participants were also asked to rate the importance of each of the five dimensions individually. All were judged positively by the majority of the sample (values ≥ 4), with the exception of the *usability* dimension, which received mostly intermediate ratings (average value 3) and, in some cases, negative (13% assigned the minimum value).

In the second section of the survey, participants were presented with a new hypothetical scenario to assess the usefulness of structured metadata in selecting and integrating software components. They were asked to imagine a situation in which a software library was to be integrated into a development environment or adopted as a commercial solution (Commercial-Off-The-Shelf, COTS). To support the decision, a JSON-formatted metadata file was provided, designed to promote information transparency by describing structural and behavioral properties of the component (a concrete example of such a file was accessible via hyperlink). The metadata file included: (i) general data, including the component name, typology, functional description, and intended context of use; (ii) a Software Bill of Materials (SBOM) compliant with the CycloneDX standard; (iii) a list of known vulnerabilities affecting the included components; (iv) MITRE ATT&CK techniques identified during sandbox-based behavioral analysis; and (v) a set of ambiguous behaviors [8], defined as actions that are not explicitly malicious but may be critical depending on the application context.

The approach was judged positively by 82.6% of participants (value ≥ 4), confirming the perceived usefulness and relevance of the initiative in both academic and industrial fields. The individual metadata components were also received favorably, with ratings equal to or above 4 on the Likert scale in most cases. Specifically, 82.6% of respondents appreciated the inclusion of the Software Bill of Materials (SBOM), 86.9% expressed a positive opinion regarding the list of known vulnerabilities, 73.9% found the reported MITRE techniques valuable, and 69.5% considered the detection of

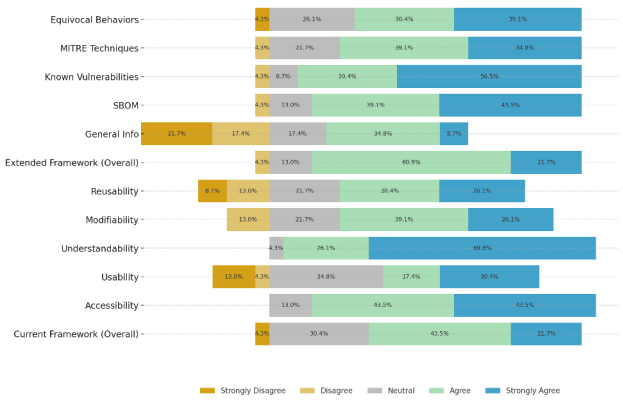

Fig. 1. Survey feedback on the two approaches.

equivocal behaviors to be useful. An exception is represented by the section related to general information, which was rated positively by 43.5% of the sample, while 39.1% assigned a value ≤ 2, highlighting a perception of limited usefulness or incompleteness of this descriptive category. All results are shown in Figure 1

Overall, 73.9% of the participants considered the newly introduced dimensions to be essential and comprehensive for enhancing the current methodology for assessing software transparency. Alongside this positive evaluation, several suggestions emerged regarding potential future extensions of the framework. Participants emphasized the importance of enabling a direct mapping between source code and known vulnerabilities, as well as including detailed information about APIs linked to MITRE techniques uncovered through dynamic analysis. Furthermore, they suggested incorporating clear data on software component licenses, indicators of compliance with relevant industry standards, and metadata elements such as code signing information, audit logs, and mechanisms for verifying the identity of developers. Some participants also stressed the importance of clearly specifying the origin of the documentation provided, whether it was provided by the original software supplier, through a declared and transparent methodology, or whether it is produced by an independent third entity. Furthermore, the need for such information to be generated through automated processes, minimizing human intervention, was reiterated. This is in order to avoid operational overloads and reduce the risk of systematic errors.

In light of the feedback collected during the survey and the insights from recent literature, in particular, from the work of Ofem et al. [20], the structuring and validation of a new framework for measuring software transparency is particularly relevant and promising. This framework, conceived as an extension of current approaches, aims at a more efficient and concrete operationalization of the concept of transparency, making it assessable in an objective, replicable, and automated way and allowing stakeholders an effective and operational assessment of transparency based exclusively on the product and on automated approaches. Based on the evidence gathered, we invite the scientific community to engage in a broader reflection on the future of this research area. We encourage to further investigate the extension of the transparency measurement model through the new dimensions proposed, which have received strong support from our participants. These dimensionsaim to support stakeholders in real contexts, providing objective and scalable support for a more informed decision-making process during the software selection and adoption phase.

As a possible evolution of the framework and the metadata file associated with the product, we propose the introduction of a structured checklist of transparency requirements. This checklist, designed as a practical and concise companion to technical documentation, would be derived from the dimensions and metadata described in the framework. This could allow the actors involved (in particular integrators and auditors) to quickly verify the presence or absence of key elements for transparency, simplifying the evaluation activities and encouraging the adoption of solutions compliant with the required transparency requirements. Finally, we highlight the need of extending the current analysis to larger and/or domain-specific samples, with the goal of validating and testing the framework in concrete industrial scenarios, in particular in continuous integration (CI/CD) and Secure Development processes.

IV. Conclusion

In this vision paper, we advocated for a more comprehensive approach to software product transparency, aimed to address requests for further investigation and development from the literature. More specifically, we analyzed the concept of software transparency itself, highlighting its broad scope of application and use, before focusing on a product-oriented perspective, which appears to have not reached an adequate state of maturity. In particular, we considered the central problem related to the transparency of a software product, i.e., the challenge of measurement. After reviewing the proposed models and methods, we pointed out their limitations and poor practical applicability.

We then explored several possible improvements to the current framework, integrating machine-readable and automatically generated documentation, and presented these ideas to a group of experts from both academia and industry. The results of the survey confirmed our hypothesis: that the current model, relying on indirect and largely manual assessment, is neither comprehensive nor scalable. It leads to assessments that are nebulous, subjective, and resource-intensive. In contrast, the participants welcomed the proposed new dimensions and methodology, identifying promising directions for future research and development. While we are aware that the extended model we propose is far from being mature and ready for adoption, we believe that the findings identified by this paper represent a first step toward building secure and transparent software systems. The question is no longer whether software transparency matters, but how we can make it practical, scalable, and integral to the software life cycle and this work initiates to chart that path.

ACKNOWLEDGMENT

The paper was partially funded by the project "PAAM – Privacy Aware Anti Malware" financed by the PRIN Program (Research Projects of National Interest) , Ministry of University and Research (MUR), Code of the Project: P20225J5YS, CUP of the project: F53D23009180001. We also acknowledge the PRIN 2022 PNRR project "FRINGE: context-aware FaiRness engineerING in complex software systEms" grant n. P2022553SL, funded by the European Union – NextGenerationEU through the Italian Ministry of University and Research.

REFERENCES

[1] J. Bonacci and R. Martin, "Software bill of materials (sbom) approach to iot security vulnerability assessment," in *International Conference on Information Technology-New Generations*. Springer, 2024, pp. 57–62.

[2] K. Brennan-Marquez and D. Susser, "Obstacles to transparency in privacy engineering," in *2016 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2016, pp. 49–52.

[3] S. Brief, "Executive order on improving the nation's cybersecurity," 2021.

[4] L. J. Camp, "Varieties of software and their implications for effective democratic government," in *Proceedings of the British academy*, vol. 135, 2006, pp. 183–185.

[5] G. Canfora, M. Di Penta, and L. Cerulo, "Achievements and challenges in software reverse engineering," *Communications of the ACM*, vol. 54, no. 4, pp. 142–151, 2011.

[6] L. Chazette, O. Karras, and K. Schneider, "Do end-users want explanations? analyzing the role of explainability as an emerging aspect of non-functional requirements," in *2019 IEEE 27th international requirements engineering conference (RE)*. IEEE, 2019, pp. 223–233.

[7] P. G. Chiara, "The cyber resilience act: the eu commission's proposal for a horizontal regulation on cybersecurity for products with digital elements: An introduction," *International Cybersecurity Law Review*, vol. 3, no. 2, pp. 255–272, 2022.

[8] G. Dalia, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "It's acting odd! exploring equivocal behaviors of goodware," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, pp. 2192–2215, 2025.

[9] G. Dalia, C. A. Visaggio, A. Di Sorbo, and G. Canfora, "Sbom ouverture: What we need and what we have," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024, pp. 1–9.

[10] F. Guitton, A. Oehmichen, É. Bossé, and Y. Guo, "Honest computing: Achieving demonstrable data lineage and provenance for driving data and process-sensitive policies," *arXiv preprint arXiv:2407.14390*, 2024.

[11] M. Hosseini, A. Shahri, K. Phalp, and R. Ali, "Four reference models for transparency requirements in information systems," *Requirements Engineering*, vol. 23, pp. 251–275, 2018.

[12] B. Isong, P. Ofem, and F. Lugayizi, "Towards a framework for improving transparency in the software engineering process," in *2022 12th International Conference on Software Technology and Engineering (ICSTE)*. IEEE, 2022, pp. 19–28.

[13] B. Kitchenham and S. L. Pfleeger, "Principles of survey research part 4: questionnaire evaluation," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 3, pp. 20–23, 2002.

[14] ——, "Principles of survey research: part 5: populations and samples," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 5, pp. 17–20, 2002.

[15] ——, "Principles of survey research part 6: data analysis," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, pp. 24–27, 2003.

[16] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research part 2: designing a survey," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 1, pp. 18–20, 2002.

[17] ——, "Principles of survey research: part 3: constructing a survey instrument," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 2, pp. 20–24, 2002.

[18] J. C. Leite, "A never-ending story: Revisiting requirements major misunderstandings," *Cadernos do IME-Série Informática*, vol. 48, pp. 9–27, 2023.

[19] T. Mudge, R. Volz, and D. Atkins, "Hardware/software transparency in robotics through object level design," in *Robotics and Industrial Inspection*, vol. 360. SPIE, 1983, pp. 216–223.

[20] P. Ofem, B. Isong, and F. Lugayizi, "On the concept of transparency: A systematic literature review," *IEEE Access*, vol. 10, pp. 89 887–89 914, 2022.

[21] ——, "Stakeholders' transparency requirements in the software engineering process," in *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2022, pp. 1–6.

[22] T. d. M. Parracho, R. O. Zacarias, M. C. d. R. Seruffo, and R. P. dos Santos, "I didn't find what i wanted-how do developers consume information in software ecosystems portals?" in *Proceedings of the XIX Brazilian Symposium on Information Systems*, 2023, pp. 143–150.

[23] C. P. Pfleeger, "Looking into software transparency," *IEEE Security & Privacy*, vol. 14, no. 1, pp. 31–36, 2016.

[24] S. L. Pfleeger and B. A. Kitchenham, "Principles of survey research: part 1: turning lemons into lemonade," *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 6, pp. 16–18, 2001.

[25] R. L. Quintanilla Portugal, P. Engiel, H. Roque, and J. C. Sampaio do Prado Leite, "Is there a demand of software transparency?" in *Proceedings of the XXXI Brazilian Symposium on Software Engineering*, 2017, pp. 204–213.

[26] J. C. Sampaio do Prado Leite and C. Cappelli, "Software transparency," *Business & Information Systems Engineering*, vol. 2, pp. 127–139, 2010.

[27] R. Schell, "Note on malicious software," *Technology Review and Update*, 2000.

[28] D. Spagnuelo, C. Bartolini, and G. Lenzini, "Metrics for transparency," in *Data Privacy Management and Security Assurance: 11th International Workshop, DPM 2016 and 5th International Workshop, QASA 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings 11*. Springer, 2016, pp. 3–18.

[29] T. Stalnaker, N. Wintersgill, O. Chaparro, M. Di Penta, D. M. German, and D. Poshyvanyk, "Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 2024, pp. 1–13.

[30] R. L. Trope and E. K. Ressler, "Mettle fatigue: Vw's single-point-of-failure ethics," *IEEE Security & Privacy*, vol. 14, no. 1, pp. 12–30, 2016.

[31] Y.-C. Tu, E. Tempero, and C. Thomborson, "Evaluating presentation of requirements documents: Results of an experiment," in *Requirements Engineering: First Asia Pacific Requirements Engineering Symposium, APRES 2014, Auckland, New Zealand, April 28-29, 2014. Proceedings*. Springer, 2014, pp. 120–134.

[32] ——, "An experiment on the impact of transparency on the effectiveness of requirements documents," *Empirical Software Engineering*, vol. 21, pp. 1035–1066, 2016.

[33] R. O. Zacarias, R. F. Gonçalves, and R. P. dos Santos, "Investigating transparency in software ecosystems," in *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, 2023, pp. 132–141.

[34] R. O. Zacarias, R. P. d. Santos, and P. Lago, "Exploring transparency as a sustainability goal in software ecosystems," in *Proceedings of the 12th ACM/IEEE International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems*, 2024, pp. 45–52.