

Understanding and Governing Dependencies in Software Engineering

Kilian Wenker¹

¹Technische Hochschule Nürnberg Georg Simon Ohm

Abstract

Background: In software engineering, the term *dependencies* commonly refers to code-level artifacts. However, dependencies in modern software systems extend beyond code to include reliance on data, services, tools, and environments used for development and operation. When such dependencies are not fully controlled by an organization, they introduce risks and constraints that require explicit governance and trade-off decisions. As the development and operation of software systems increasingly rely on such dependencies, understanding and governing them becomes a central software engineering challenge. *Objective:* The objective of this thesis is to advance the understanding of dependencies in the development and operation of software systems, to develop software engineering principles, methods, and supporting artifacts for identifying, assessing, and governing dependencies, and to empirically evaluate the resulting governance support. *Methods:* This thesis follows a design-science-oriented research approach, grounded in both conceptual and empirical investigation. A systematic mapping study and exploratory case studies will be conducted to investigate dependencies in software engineering and to develop an ontology of dependencies. This ontology will classify dependencies and relate them to risks, constraints, and governance concerns. Building on this ontology, governance support in the form of software engineering principles, methods, and supporting artifacts will be designed and empirically evaluated through case studies in realistic software engineering contexts. *Conclusion:* This thesis is expected to contribute a systematic understanding of dependencies in software systems and to provide software engineering approaches for their governance, thereby supporting more informed and autonomous development and operation.

Keywords

software engineering, dependency governance, external dependencies, digital sovereignty, design science research, case study, systematic mapping study

1. Introduction

Dependencies are a fundamental part of software engineering. They shape how software systems are designed, implemented, deployed, and operated. The term *dependencies* is commonly associated with code-level artifacts, such as libraries or reusable components. However, this understanding is too narrow for contemporary software engineering practice. Software systems also rely on data, services, tools, and operational environments that influence both development and operation. Many of these dependencies are external to the organizations that rely on them. They allow organizations to reuse components, accelerate development, reduce implementation effort, build on existing services, platforms, tools, and operational capabilities. At the same time, reliance on such dependencies introduces two central challenges.

First, external dependencies are not fully under the control of the organization. This introduces risks and constraints that may affect maintainability, portability, security, compliance, substitutability, operational resilience, and long-term autonomy. Consequently, dependency-related decisions are not merely technical implementation choices, but software engineering decisions involving trade-offs.

Second, dependencies may be implicit, fine-grained, and subject to change over time. They can be embedded in architectural decisions, configuration choices, data formats, APIs, toolchains, deployment environments, or operational conventions. Moreover, both the set of dependencies an organization has as well as the dependencies themselves are subject to change. They evolve as software systems change, external services are updated, tools are replaced, and operational environments shift. This

IDOESE'26: International Doctoral Symposium on Empirical Software Engineering, October 07, 2026, München, Germany

✉ wenkerki110316@th-nuernberg.de (K. Wenker)

ORCID 0000-0003-0145-6862 (K. Wenker)



© 2026 Copyright for this paper by its author. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

makes dependencies moving targets that are difficult to identify, assess, and govern systematically. The increasing use of autonomous or agent-based software systems may further complicate dependency governance. This is because such systems may introduce or rely on dependencies dynamically and in ways that are not fully transparent to developers and operators.

Since external dependencies offer major advantages while also introducing significant risks, their governance becomes a necessary software engineering concern.

Current software engineering research and practice often address dependencies in isolated contexts. Some dependency-related concerns, such as open-source license compliance, vulnerability management, and software composition analysis, are comparatively well understood and supported by established tools. However, these approaches only partially address the broader need to govern dependencies across development and operation. What is missing is a software engineering perspective that treats dependencies as a first-class concern across both development and operation.

We would like to contribute towards this vision of software engineering that manages dependencies responsibly by creating *governance support*. In this thesis, governance support refers to software engineering principles, methods, and supporting artifacts that help practitioners identify, assess, and govern dependencies.

The objective of this thesis is to advance the understanding of dependencies in the development and operation of software systems, to develop software engineering principles, methods, and supporting artifacts for identifying, assessing, and governing dependencies, and to empirically evaluate the resulting governance support.

This objective is addressed through the following four research questions:

- RQ1** What types of dependencies occur in the development and operation of software systems, and what risks, constraints, and governance concerns are associated with them?
- RQ2** How can dependencies be systematically identified and represented to support their assessment and governance?
- RQ3** Which software engineering principles, methods, and supporting artifacts can support practitioners in governing dependencies?
- RQ4** How useful, applicable, and understandable is the proposed governance support in realistic software engineering contexts?

To this end, we apply a design-science-oriented research approach grounded in conceptual and empirical investigation [1]. Following this approach, we will conduct a systematic mapping study and exploratory case studies to address RQ1 and to inform RQ2. The results will be synthesized into an ontology of dependencies in software engineering. This ontology is intended as an intermediate artifact that classifies dependencies and relates them to risks, constraints, and governance concerns. As an initial framing, such governance concerns might include visibility, ownership, traceability, accountability, substitutability, controllability, compliance, decision-making, and lifecycle management. This set is not assumed to be complete. The identification, refinement and structuring of such concerns is part of the planned research. Table 1 summarizes an initial framing of governance categories, concerns, and guiding questions. These questions provide an initial impression for relating dependency types to governance concerns and for motivating the identification and representation mechanisms addressed in RQ2. Building on the ontology, we will design software engineering principles, methods, and supporting artifacts for dependency governance to address RQ3, and evaluate the resulting governance support to address RQ4.

2. Background and Related Work

The existing literature addresses some aspects of the governance of dependencies in software engineering, such as Software Bill Of Materials (SBOM) [2], dependency structure [3], or supply chain security [4, 5]. These aspects are comparatively well supported by existing tools and standards [6, 7]. Related work has also examined on dependencies from the perspective of requirements engineering [8].

Table 1

Initial governance categories, concerns, and questions for dependency governance.

Category	Concern	Governance question
Awareness	Visibility	Do we know which dependencies exist?
	Traceability	Can we trace where and why a dependency is used?
Responsibility	Ownership	Who is responsible for a dependency?
	Accountability	Who is accountable if a dependency becomes problematic?
	Decision-making	Who decides whether a dependency is acceptable?
	Compliance	Does the dependency meet legal and security requirements?
Control and evolution	Substitutability	How can the dependency be replaced?
	Steering	How much influence does the organization have over the dependency?
	Lifecycle management	How is the dependency monitored, updated, replaced, or retired?

The insights and methods in these contributions can serve as a basis for the holistic perspective pursued in this thesis.

For the larger field of *digital sovereignty*, there is a range of works from various disciplines [9, 10]. These contributions tend to focus on understanding the motivations, goals and political implications of digital sovereignty. Our proposed research would add to the body of research on digital sovereignty from a different angle, as we wish to explore how software-centric organizations can strengthen digital sovereignty in practice.

In Europe, the realization of digital sovereignty is rapidly gaining interest from software-centric organizations [10, 11] and researchers. A recent symposium paper on *digital sovereignty in organizational end-to-end processes* [12] is further indication of the relevance of the subject. Their work differs from ours in its focus on business process management, political science, and sociotechnical research, whereas this thesis aims to develop software engineering support for dependency governance in practice.

Note that political science sees *digital sovereignty* as a term that has many, often contradictory definitions [13, 14]. This discourse often focuses on policies and strategies at the geopolitical level, but also includes the organizational and individual level [9]. For our research, we approach the issue of strengthening digital sovereignty for software-centric organizations from a software engineering perspective. For this context, we understand digital sovereignty as the ability of organizations to make informed and deliberate decisions about dependencies on externally controlled data, software, services, tools, and operational environments.

Overall, existing work provides important foundations for selected dependency-related concerns, especially software composition, SBOMs, license compliance, vulnerability management, and supply-chain security. However, these streams only address specific dependency types or governance concerns rather than dependencies as a broader software engineering concern across development and operation. Currently, there is no unified model that disambiguates the various types of dependencies an organization faces. Further, existing work on digital sovereignty motivates the relevance of this broader perspective, but does not yet provide software engineering methods for identifying, assessing, and governing dependencies in practice. To best support practitioners, we plan to pair heightened understanding of digital sovereignty with practical methods for governing dependencies effectively. This thesis addresses this gap by conceptualizing dependencies more broadly and by developing software engineering support for their governance.

3. Methods

For this thesis, we follow a design science methodology [1], structuring the work around problem investigation, treatment design, and treatment validation. This allows us to contribute to problem conceptualization as well as to the design and validation of governance support [15].

In the problem conceptualization part, we investigate dependencies in the development and operation of software systems, while in the design-oriented part, we develop and empirically evaluate governance support for identifying, assessing, and governing dependencies. We will conduct the literature study as a systematic mapping study [16] following established guidelines for secondary studies in software engineering [17, 18]. We aim to identify and structure existing concepts, classifications, risks, constraints, and governance concerns related to dependencies in software engineering. Since dependency governance is also shaped by industrial practice and regulation, we will include academic literature, selected grey literature, and regulatory texts. Grey literature may include standards, technical reports, tool documentation, and practitioner-oriented publications, guided by recommendations for multivocal literature reviews in software engineering [19]. Regulatory texts will be considered insofar as they inform dependency-related governance concerns, constraints, and decision criteria. Database search will be complemented by backward and forward snowballing [20].

Additionally, we will use exploratory multiple-case studies following established guidelines for case study research in software engineering [21]. Case studies are appropriate because dependencies occur in real development and operation contexts and are shaped by architectural decisions, toolchains, deployment environments, organizational practices, and operational constraints. Through these case studies, we will examine how dependencies arise across development and operation, how they are documented or remain implicit, how they change over time, and which risks, constraints, and governance concerns they introduce.

Based on these results of the systematic mapping study and the exploratory case studies, we will develop an ontology of dependencies in software engineering. To this end, we map dependencies onto a foundational ontology (e.g. DOLCE [22]). This ontology will serve as a well-founded model for classifying dependencies and relate them to risks, constraints, and governance concerns. It serves as an intermediate design artifact that structures the problem space and provides a foundation for subsequent treatment design. Additionally, it informs our definition of dependencies and thus helps refine the scope of the project.

Building on this ontology, we will design software engineering principles, methods, and supporting artifacts (e.g. software tools, guidelines) for dependency governance. These artifacts are intended to support practitioners in identifying, assessing, and governing dependencies in the development and operation of software systems.

For treatment validation, we will conduct evaluation case studies in realistic software engineering contexts. This is appropriate because dependency governance is context-dependent and cannot be fully assessed independently of the software systems, toolchains, deployment environments, and organizational practices in which dependencies occur. Within these case studies, focus groups [23] of experts will serve as qualitative empirical method for assessing the usefulness, applicability, understandability, and limitations of the proposed governance support. Additional data collection may include interviews, workshops, document analysis, and artifacts from development and operation contexts. The resulting qualitative data will be analyzed to identify perceived strengths, weaknesses, missing aspects, and contextual conditions for applying the proposed governance support.

Treatment design and treatment validation are expected to proceed iteratively, with validation results informing refinements of the governance support.

We consider open science a core responsibility of this thesis. Accordingly, research protocols, search strings, inclusion and exclusion criteria, data extraction forms, coding schemes, ontology versions, and supporting materials will be made openly available whenever possible. For industry-related data, openness will be balanced with confidentiality, privacy, and contractual obligations; where direct sharing is not possible, anonymized, aggregated, or sanitized artifacts will be provided to preserve transparency, traceability, and reuse.

4. Research Plan

The thesis is currently in the planning stage and the empirical work has not yet started. The first phase is planned to begin in autumn. The research is structured into three phases following the design science distinction between problem investigation, treatment design, and treatment validation [1].

Figure 1 outlines the planned research process: Phase 1 focuses on problem investigation and dependency conceptualization. It uses a systematic mapping study based on academic literature, selected grey literature, and regulatory texts, as well as exploratory case studies with industry partners. The outcome of this phase is an ontology of dependencies that relates dependency types to risks, constraints, and governance concerns. This ontology will allow us to specify for the rest of this research what the concept *dependency* includes (and what it does not). Phase 2 uses the ontology as the basis for treatment design. In this phase, software engineering principles, methods, and supporting artifacts for dependency governance will be designed. Phase 3 validates the resulting governance support through evaluation case studies in realistic software engineering contexts. The validation will focus on the usefulness, applicability, and limitations of the proposed governance support.

Although presented as phases, the process is not intended to be strictly linear. In particular, treatment design and treatment validation are expected to proceed iteratively, with validation results informing refinements of the governance support.

The feasibility of the planned case studies and validation activities is supported by letters of intent from eight industry partners. These partners are expected to provide access to realistic software engineering contexts, including software systems, practitioners, development and operation artifacts, and organizational practices related to dependency governance.

The applicability of the proposed research will depend on the applied methodologies, selection of industry partners, and the moment in time at which the research will be conducted. The latter is particularly relevant given the rapidly changing role of AI in software engineering, which may limit the transferability of the findings over time. We do not presume that the results will apply in all contexts. We will therefore document the studied contexts and the boundaries of our conclusions. Researcher and participant bias will be addressed through documented protocols, traceable analysis procedures, and consideration of contradictory evidence. Finally, the evaluation will support conclusions about the usefulness, applicability, and understandability of the proposed governance support, but not necessarily about causal effectiveness or statistical generalizability.

5. Conclusion

This research plan frames dependency governance as a software engineering challenge. Dependencies in modern software systems extend beyond code-level artifacts and may be external, implicit, fine-grained, and evolving, making them difficult to identify, assess, and govern systematically across development and operation.

The thesis is expected to contribute:

- A broader understanding of dependencies in software engineering, including dependencies that arise across development and operation.
- An ontology of dependencies that relates dependency types to risks, constraints, and governance concerns.
- Software engineering principles, methods, and supporting artifacts for identifying, assessing, and governing dependencies.
- Empirical insights from the systematic mapping study, exploratory case studies, and design-science-oriented evaluation in realistic software engineering contexts.

By providing these contributions, the thesis seeks to support more informed software engineering decisions and enable organizations to attain digital sovereignty through the governance of their dependencies in the development and operation of software systems.

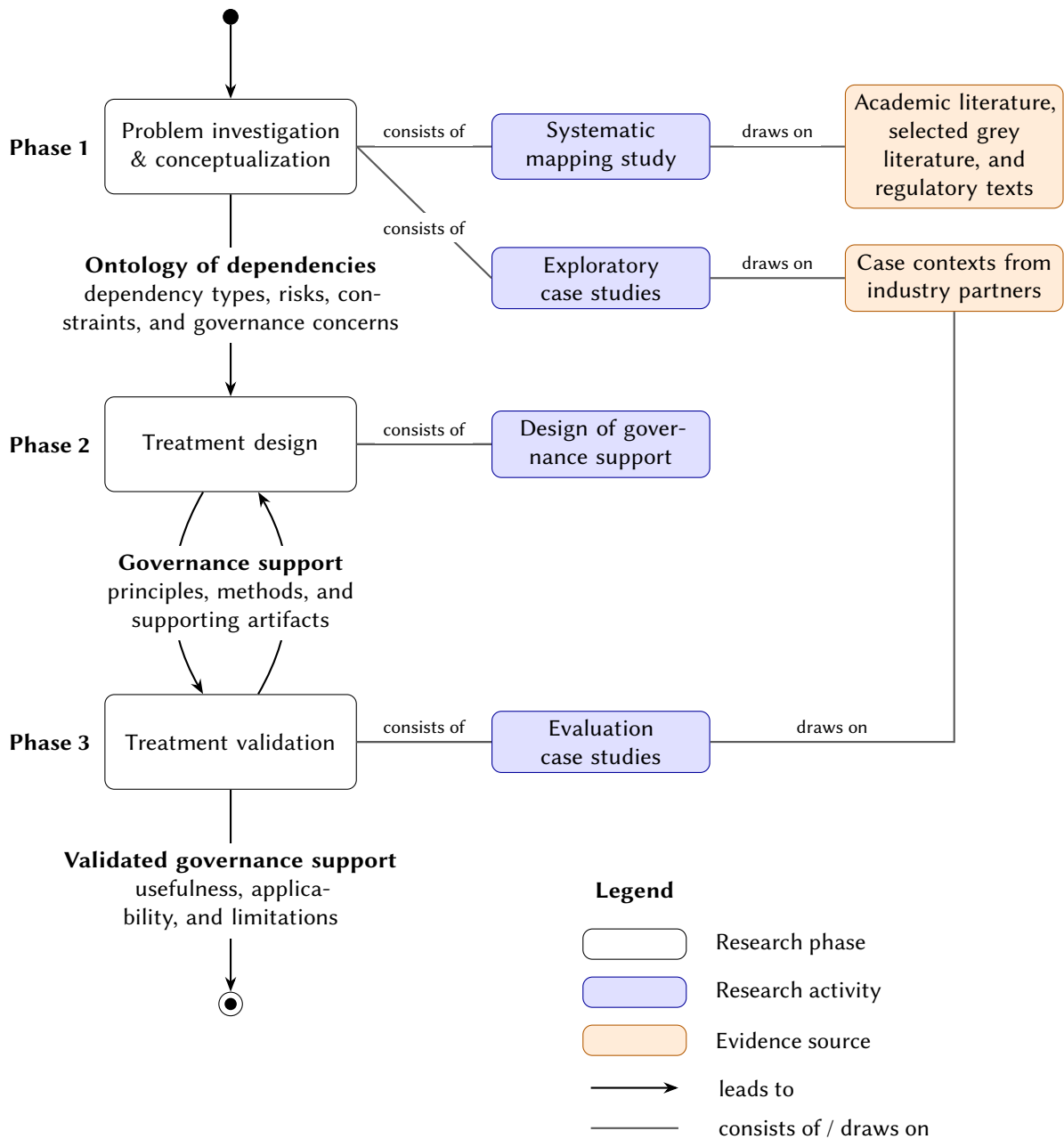


Figure 1: Planned research process structured according to problem investigation, treatment design, and treatment validation.

Declaration on Generative AI

During the preparation of this work, the author used GPT-5.6 Luna in order to: Check grammar and spelling, paraphrase and reword. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

- [1] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-43839-8.
- [2] G. Scanniello, M. Di Penta, S. Romano, R. Francese, S. Nocera, P. Cassieri, D. Bifulco, F. Zampetti, MSR4SBOM: Mining Software Repositories for Enhanced Software Bills of Materials, in: Proceed-

- ings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, Barcelona Spain, 2024, pp. 589–593. doi:10.1145/3674805.3695390.
- [3] D. Cui, X. Li, F. Liu, S. Wang, J. Dai, L. Wang, Q. Li, Towards Demystifying the Impact of Dependency Structures on Bug Locations in Deep Learning Libraries, in: Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, Helsinki Finland, 2022, pp. 249–260. doi:10.1145/3544902.3546246.
 - [4] N. Imtiaz, S. Thorn, L. Williams, A Comparative Study of Vulnerability Reporting by Software Composition Analysis Tools, in: Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, Bari Italy, 2021, pp. 1–11. doi:10.1145/3475716.3475769.
 - [5] A. C. Latsiou, A. R. Nygård, S. Katsikas, C. Lambrinoudakis, Never Trust - Always Verify: Assessing the Cybersecurity Trustworthiness of Suppliers in the Digital Supply Chain, *Procedia Computer Science* 254 (2025) 98–107. doi:10.1016/j.procs.2025.02.068.
 - [6] G7 Cybersecurity Working Group, Software Bill of Materials (SBOM) for Artificial Intelligence - Minimum Elements, Technical Report, Federal Office for Information Security (BSI), 2026. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/SBOM-for-AI_minimum-elements.html.
 - [7] OWASP, Dependency-Track Contributors, Dependency-Track, 2026. URL: <https://dependencytrack.org/>.
 - [8] A. Martakis, M. Daneva, Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners, in: IEEE 7th International Conference on Research Challenges in Information Science, 2013, pp. 1–11. doi:10.1109/rcis.2013.6577679.
 - [9] G. Glasze, A. Cattaruzza, F. Douzet, F. Dammann, M.-G. Bertran, C. Bômout, M. Braun, D. Danet, A. Desforges, A. Géry, S. Grumbach, P. Hummel, K. Limonier, M. Münßinger, F. Nicolai, L. Pétiñaud, J. Winkler, C. Zanin, Contested Spatialities of Digital Sovereignty, *Geopolitics* 28 (2023) 919–958. doi:10.1080/14650045.2022.2050070.
 - [10] F. G. Blancato, M. Carr, The Trust Deficit. EU Bargaining for Access and Control Over Cloud Infrastructures, *Journal of European Public Policy* 33 (2026) 936–967. doi:10.1080/13501763.2024.2441418.
 - [11] T. Y. Koh, J. Prenio, Managing Cloud Risk: Some Considerations for the Oversight of Critical Cloud Service Providers in the Financial Sector, number 53 in FSI insights on policy implementation, Bank for International Settlements, Financial Stability Institute, Basel, 2023. URL: <https://www.bis.org/fsi/publ/insights53.pdf>.
 - [12] A. Schneider, T. Schuster, M. Bosslau, L. Waidelich, Digital Sovereignty in Organisational End-to-End Processes: An Early-Stage Design-Oriented Research Roadmap, in: L. M. Camarinha-Matos, F. Ferrada (Eds.), *Technological Innovation to Tackle Societal Challenges*, volume 790, Springer Nature Switzerland, Cham, 2027, pp. 20–31. doi:10.1007/978-3-032-28939-1_2.
 - [13] E. Repetto, Concept ‘Stretching’ or Concept Innovation? A Review of the Usages of Sovereignty in the Digital Sovereignty Literature, *Policy & Internet* 17 (2025) e70011. doi:10.1002/poi3.70011.
 - [14] J. Pohle, R. Nanni, M. Santaniello, Unthinking Digital Sovereignty: A Critical Reflection on Origins, Objectives, and Practices, *Policy & Internet* 16 (2024) 666–671. doi:10.1002/poi3.437.
 - [15] E. Engström, M.-A. Storey, P. Runeson, M. Höst, M. T. Baldassarre, How Software Engineering Research Aligns With Design Science: A Review, *Empirical Software Engineering* 25 (2020) 2630–2660. doi:10.1007/s10664-020-09818-7.
 - [16] P. Ralph, S. Baltes, Paving the way for mature secondary research: the seven types of literature review, in: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 2022, pp. 1632–1636. doi:10.1145/3540250.3560877.
 - [17] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Technical Report, 2007.
 - [18] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic Mapping Studies in Software Engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software

Engineering, EASE '08, BCS Learning & Development Ltd., Swindon, GBR, 2008, pp. 68–77. doi:10.14236/ewic/EASE2008.8.

- [19] V. Garousi, D. Pfahl, J. M. Fernandes, M. Felderer, M. V. Mäntylä, D. Shepherd, A. Arcuri, A. Coşkunçay, B. Tekinerdogan, Characterizing Industry-Academia Collaborations in Software Engineering: Evidence from 101 Projects, *Empirical Software Engineering* 24 (2019) 2540–2602. doi:10.1007/s10664-019-09711-y.
- [20] C. Wohlin, Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering, in: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14*, ACM Press, 2014, pp. 1–10. doi:10.1145/2601248.2601268.
- [21] P. Runeson, M. Host, A. Rainer, B. Regnell, *Case Study Research in Software Engineering*, 2012. doi:10.1002/9781118181034.
- [22] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E. M. Sanfilippo, L. Vieu, DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering, *Applied Ontology* 17 (2022) 45–69. doi:10.3233/AO-210259.
- [23] J. Kontio, J. Bragge, L. Lehtola, *The Focus Group Method as an Empirical Tool in Software Engineering*, Springer London, 2008, pp. 93–116. doi:10.1007/978-1-84800-044-5_4.