Knowledge Management Processes in Software Reuse: A Rapid Review

Emilly Lopes, Carlos Eduardo Barbosa[1,2], Matheus Argôlo[1], Lucas Nóbrega[1], Luiz Felipe Martinez[1], Claudia Werner[1] and Jano de Souza[1]

[1]Systems Engineering and Computer Science Program of the Alberto Luiz Coimbra Institute for Graduate Studies and Research in Engineering, Federal University of Rio de Janeiro, Brazil
[2]Naval Systems Analysis Centre, Brazilian Navy, Rio de Janeiro, Brazil

emlopes@cos.ufrj.br
eduardo@cos.ufrj.br
matheusargolo@cos.ufrj.br
lucasnobrega@cos.ufrj.br
felipem@cos.ufrj.br
werner@cos.ufrj.br
jano@cos.ufrj.br

Abstract: Software development is a knowledge-intensive activity, and managing knowledge effectively is essential to ensuring productivity and quality. Software reuse (SR) is a strategic approach that improves efficiency and return on investment, but sustaining reuse programs remains challenging. Organisations often struggle to manage the knowledge embedded in software assets such as code, designs, and test cases. Knowledge management (KM) processes—identification, acquisition, sharing, storage, codification, creation, application, and evaluation—offer a promising lens for addressing these challenges. Yet, recent work has paid limited attention to their explicit role in SR. This exploratory work analyses the recent literature on SR from a KM perspective, seeking to understand how the main knowledge processes support SR. To this end, a rapid literature review was conducted to identify studies that explicitly discuss KM aspects of SR. The review applied inclusion criteria focused on studies that explicitly discuss KM for SR, and exclusion criteria that removed unavailable studies, review-only studies, and studies that did not answer the research questions. Each selected study was then mapped, categorised, and analysed using a protocol aligned with the research questions, considering the KM processes addressed, the types of assets reused at each reuse level (specification reuse, design reuse, code reuse, application system reuse, and test reuse), and the research method employed. By structuring the evidence around these dimensions, the review helps characterise how knowledge processes are discussed in relation to different forms of reuse, different reusable artefacts, and both operational and strategic reuse concerns. Rather than consolidating a mature theory, the results indicate tentative paths for connecting KM and SR research. In particular, the analysis indicates a predominance of storage and codification, followed by application and sharing. At the same time, the findings highlight the need to deepen research on knowledge evaluation, test reuse, and the organisational conditions that help sustain software reuse initiatives over time.

Keywords: Knowledge management, Software engineering, Software reuse, Rapid review

1. Introduction

Software development is a knowledge-intensive activity, since software projects depend on human expertise, coordination, communication, and the reuse of previous experience (Panagiotou and Mentzas, 2011). In this context, Software Reuse (SR) can be understood not only as a technical practice, but also as a form of knowledge reuse. Software artefacts such as code components, requirements documents, functional designs, and test cases embody knowledge that can be preserved, shared, adapted, and applied in future projects (Spoelstra, 2010).

SR has both strategic and operational dimensions. At the strategic level, organisations need to sustain reuse programs, align reuse with business objectives, and create conditions for long-term adoption. At the operational level, developers need mechanisms to identify, understand, evaluate, and adapt reusable artefacts during the software lifecycle. These concerns are related to known SR challenges, such as sustaining reuse programs, integrating reuse into organisational strategy, supporting professionals, and improving software processes (Frakes and Kang, 2005; Antovski and Imeri, 2013).

Despite this connection, KM and SR are often discussed through different vocabularies. For a KM audience, SR offers a practical setting in which knowledge reuse appears through artefacts, repositories, development methods, documentation, and evaluation practices. However, recent literature still provides limited integrated discussion of KM processes explicitly related to SR.

Therefore, this work conducts an exploratory Rapid Review (RR) to identify how Knowledge Management Processes (KMP) are addressed in studies on SR. The aim is not to consolidate a mature theory, but to map how

the selected studies connect KM processes, reuse levels, and the sustainability of reuse initiatives, indicating tentative paths for future KM–SE research.

2. Theoretical Background

This section presents the KM and SR concepts that support the review and the coding of the analysed studies.

2.1 Knowledge Management Processes

KM may be analysed through people, processes, and technology (Edwards, 2011). This study focuses on the process dimension because SR depends on repeatable activities for locating, representing, transferring, evaluating, and applying reusable knowledge. We treat KMP as interrelated activities for managing knowledge assets (Hosseingholizadeh, 2014).

The coding model combines six processes commonly used in KM studies—acquisition, storage, codification, sharing, application, and creation (Costa & Monteiro, 2016)—with two additional processes: identification and evaluation. Identification distinguishes the mapping of knowledge already available inside the organisation from acquisition from external sources. Evaluation was added because reuse also requires checking whether an asset, component, design, requirement, or test artefact is suitable for a new context. Table 1 summarises the eight KMP used in the analysis.

Table 1: Knowledge management processes and their main activities

Process	Main Activities
Identification	Mapping (focus on organisation's internal resources)
Acquisition	Capture (focus on organisation's external resources)
Sharing	Dissemination, distribution, transfer, assimilation
Storage	Preservation, Retention, Update
Codification	Indexing, encoding, filtering, externalisation
Creation	Construction, development, innovation
Application	Use, action, internalisation.
Evaluation	Review, validation, verification

2.2 Software Reuse

Software Reuse (SR) refers to building new systems from existing software artefacts, including source code, models, requirements, architectures, test plans, and complete systems (Castro and Werner, 2021). SR is not restricted to copying code. It also includes approaches such as component-based development, software product lines, model-driven development, and domain engineering (Barros-Justo et al., 2019). For a KM audience, these approaches can be understood as different ways of making development knowledge available for future use.

A reusable artefact only produces value when it can be discovered, understood, trusted, adapted, and applied in another project. This links SR directly to KM processes. At the strategic level, organisations need policies, incentives, metrics, and long-term support for reuse programs. At the operational level, professionals need repositories, documentation, semantic descriptions, feedback mechanisms, and evaluation criteria to decide whether an artefact can be reused. These concerns are associated with known challenges in SR, including sustaining reuse programs, organisational integration, technology transfer, and process support (Frakes and Kang, 2005; Antovski and Imeri, 2013).

For coding purposes, this review adopted Younoussi & Roudies' (2015) five reuse levels: specification reuse, such as requirements documents; design reuse, such as architectures or component designs; code reuse, such as source code or subroutines; application-system reuse, in which complete systems are deployed in new environments; and test reuse, such as test plans and test cases.

3. Methodology

This section summarises the RR protocol adopted to analyse KMP in SR. An RR was selected because it enables a structured synthesis in less time than a full systematic review, which is suitable for an exploratory paper on a fragmented research intersection. The review was carried out in June 2025, following Cartaxo et al. (2020),

Kitchenham & Charters (2007), and PRISMA 2020 reporting principles (Page et al., 2021). The objective was to provide a transparent exploratory sample rather than an exhaustive systematic review.

3.1 Rapid Review Planning

The RR was guided by three research questions focused on KM processes, reuse levels, and the contributions and methods of the selected studies:

- RQ1: What are the main KM processes studied, considering their relationship with software reuse?
- RQ2: What levels of reuse are primarily studied that involve KM processes?
- RQ3: What are the main contributions and research methods addressed in the analysed studies?

Scopus was used as the search source. The search was conducted in June 2025 using title terms related to reuse, software or assets, and knowledge management or processes. The search was limited to articles and conference papers published in English between 2015 and 2025. Title-based searching was adopted to accelerate the review, consistent with the RR scope. The final search string is shown in Table 2.

Table 2: Query string used in this work

"TITLE (""reus*"" AND (""software"" OR ""asset"")) AND (""Knowledge"" AND (""Management"" OR ""Process*"")) AND PUBYEAR > 2014 AND PUBYEAR < 2026 AND (LIMIT-TO (DOCTYPE , ""cp"") OR LIMIT-TO (DOCTYPE , ""ar"")) AND (LIMIT-TO (LANGUAGE , ""English""))"

3.2 Conducting the Rapid Review

The Scopus search returned 176 results. After abstract screening, 22 articles were pre-selected for explicitly addressing KM aspects in SR. Seven were excluded because full texts were unavailable, two because their contribution was limited to literature review, and three after full-text reading because they did not answer the research questions. Thus, 10 publications from Scopus were included. Table 3 show the inclusion and exclusion criteria.

Backward snowballing was then conducted using the reference lists of these 10 studies. After title and abstract screening, six additional studies were read in full and five were included. The final sample comprised 15 studies. Each study was coded using a structured extraction form covering metadata, main contribution, KM processes, reuse levels, and research methods. Figure 1 summarises the selection process.

Table 3: Inclusion and exclusion criteria

Inclusion	Exclusion
(I-1) Include only studies that explicitly discuss knowledge management aspects of software reuse	(E-1) Exclude articles that do not explicitly discuss knowledge management aspects of software reuse
(I-2) Include only studies in English	(E-2) Exclude unavailable studies (e.g., paid or inaccessible)
(I-3) Include only studies published between 2015 and 2025	(E-3) Exclude studies whose contribution is limited to literature review, without the development of theoretical or methodological proposals or empirical applications
	(E-4) Exclude studies that do not answer the research questions

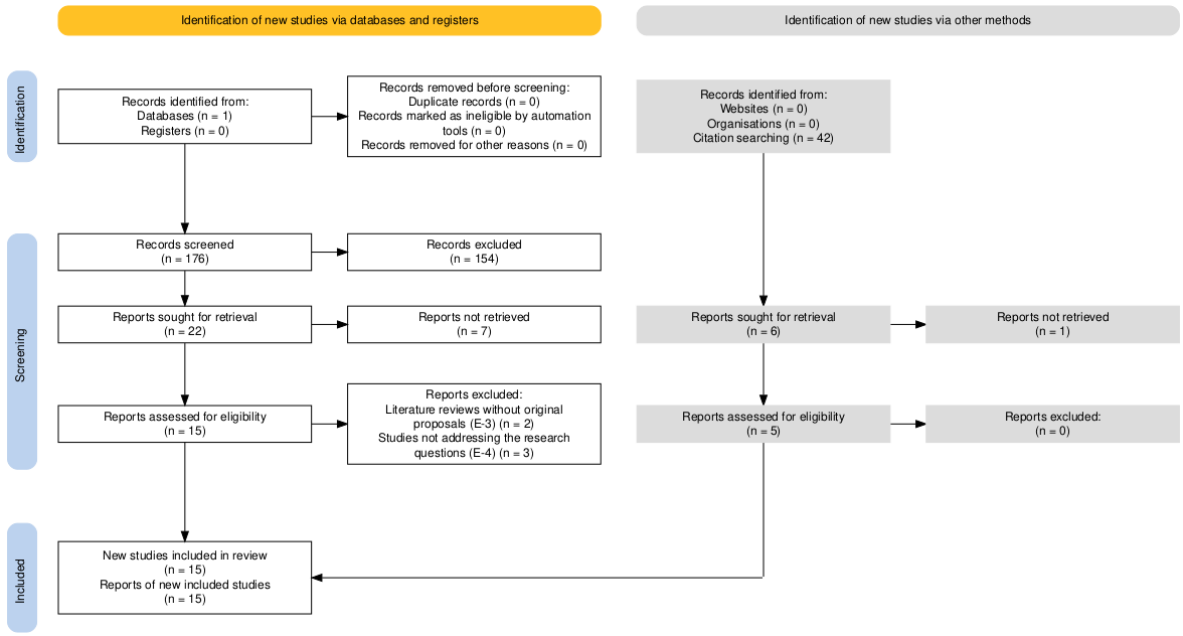

Figure 1: Rapid review flow diagram (Haddaway et al., 2022) following PRISMA 2020 (Page et al., 2021)

4. Results

Based on the discussion presented thus far on KMP and SR, this section presents the main results of the analysis performed on the sample of 15 selected articles. Regarding the KMP identified in the studies, the main knowledge processes addressed were: Codification (n=15), Storage (n=15), Application (n=14), Sharing (n=13), Creation (n=12), Identification (n=11), Evaluation (n=11), and Acquisition (n=7). Regarding the levels of reuse discussed most frequently, it is noteworthy that only one study presented Application Systems Reuse (n=1), while the most frequently identified level of reuse was Specification Reuse (n=12), followed by Code Reuse (n=11) and Design Reuse (n=8), and finally Test Reuse (n=5), which was identified in one third of the studies.

From a research method perspective, the most explored methods in the literature were empirical analysis using controlled experiments (n=6) and Proof of Concept (n=6). It was also noted that literature review methods (n=4) are commonly used to complement the studies presented. Furthermore, survey (n=2), exploratory case studies (n=1), design sprints (n=1), design science research - DSR (n = 1), qualitative manual analysis, evidence-based software engineering (n=1), Keep All Objects Satisfied - KAOS (n=1), and data triangulation (n=1) were identified as research methods in the analysed studies. A summary of the main results of the analysis is presented in Table 4.

Table 4: Knowledge management processes, reuse levels and research methods

Ref.	Knowledge Management Processes	Reuse Levels	Research Methods
Putraprata ma et al. (2021)	Identification, Creation, Evaluation, Codification, Sharing, Storage	Design, Code	Literature Review, Data Triangulation, Design Sprint
Polenghi et al. (2022)	Identification, Acquisition, Creation, Evaluation, Codification, Sharing, Storage, Application	Specification	Literature Review, Empirical Analysis using Controlled Experiment
Jabbar et al. (2019)	Identification, Creation, Evaluation, Codification, Sharing, Storage, Application	Specification, Design, Code	Empirical Analysis using Controlled Experiment
Carlier et al. (2023)	Identification, Creation, Codification, Sharing, Storage, Application	Specification, Design, Code	Proof of Concept
Hamdi et al. (2022)	Identification, Creation, Evaluation, Codification, Storage, Application	Specification, Code	Empirical Analysis using Controlled Experiment

Ref.	Knowledge Management Processes	Reuse Levels	Research Methods
Chen et al. (2022)	Identification, Acquisition, Creation, Evaluation, Codification, Sharing, Storage, Application	Specification, Design, Code, Test	Exploratory Case Study
Melo et al. (2020)	Identification, Acquisition, Evaluation, Codification, Sharing, Storage, Application	Code	Literature Review, Proof of Concept
Hu et al. (2021)	Identification, Acquisition, Codification, Sharing, Storage, Application	Test	Proof of Concept
Maccanti et al. (2016)	Identification, Acquisition, Creation, Codification, Sharing, Storage, Application	Specification, Design, Code, Test	Proof of Concept
Elgedawy (2015)	Identification, Creation, Codification, Sharing, Storage, Application	Specification, Design, Code, Test	Proof of Concept
Fahmideh & Beydoun (2018)	Acquisition, Creation, Evaluation, Codification, Storage, Application	Specification, Design, Code, Application system	Literature Review, Design Science Research (DSR), Evidence-Based Software Engineering (EBSE), KAOS (Keep All Objects Satisfied)
Campos et al. (2016)	Acquisition, Evaluation, Codification, Sharing, Storage, Application	Specification, Code	Empirical Analysis using Controlled Experiment, Qualitative Manual Analysis
Rahman et al. (2017)	Acquisition, Evaluation, Codification, Sharing, Storage, Application	Specification, Code	Empirical Analysis using Controlled Experiment
Fraga et al. (2019)	Creation, Evaluation, Codification, Sharing, Storage, Application	Specification, Design	Empirical Analysis using Controlled Experiment, Survey
Vasanthapriyan et al. (2017)	Identification, Creation, Evaluation, Codification, Sharing, Storage, Application	Specification, Test	Proof-of-Concept, Survey

Beyond the frequency counts, the selected papers suggest three recurring ways in which KM is embedded in SR. First, several studies treat reuse as a problem of representation and retrieval. Ontologies, knowledge graphs, semantic repositories, and portals are used to organise reusable knowledge so that developers can find and apply it later (Putrapratama et al., 2021; Polenghi et al., 2022; Fraga et al., 2019; Vasanthapriyan et al., 2017; Hu et al., 2021; Elgedawy, 2015). This helps explain the high incidence of codification and storage.

Second, some studies focus on operational support for developers. Requirements traceability, code search, and API recommendation approaches use prior artefacts, user feedback, or community knowledge to support current development tasks (Hamdi et al., 2022; Melo et al., 2020; Rahman et al., 2017; Campos et al., 2016). These studies connect acquisition, evaluation, and application, especially in code and specification reuse.

Third, a smaller group discusses organisational or strategic conditions for reuse. Chen et al. (2022) highlight reusable assets and knowledge sharing in a company context; Maccanti et al. (2016) organise documentation and standards through a KM framework; Jabbar et al. (2019) link KM practices to SCRUM-based reuse; and Fahmideh & Beydoun (2018) reuse empirical knowledge to support cloud adoption decisions. These studies are useful for discussing reuse sustainability because they show that reuse depends not only on artefacts, but also on processes, coordination, and decision support.

5. Discussion

The findings should be interpreted as exploratory indications rather than as evidence of a consolidated KM–SR research field. This positioning is important because the selected studies use different vocabularies, methods, and technical contexts. Nevertheless, the sample suggests that SR can be read as a concrete manifestation of knowledge reuse in software development. In KM terms, there is little value in identifying, acquiring, or storing knowledge if it cannot later be reused. In SR terms, reusable artefacts only become useful when the knowledge embedded in them can be found, understood, evaluated, adapted, and applied.

The predominance of codification and storage indicates that many SR studies approach KM mainly at the operational level. The recurring use of ontologies, knowledge graphs, semantic repositories, portals, and documentation frameworks shows a concern with making knowledge explicit, searchable, and technically reusable. This is valuable, but it also reveals a limitation: codified knowledge does not guarantee reuse by itself. Reuse also depends on sharing practices, evaluation criteria, trust in the artefact, and feedback from use in real projects.

The results also indicate a distinction between operational and strategic perspectives. Operationally, studies on code search, API recommendation, requirements traceability, test data reuse, and semantic retrieval support developers in specific tasks. Strategically, fewer studies address how organisations sustain reuse over time through governance, incentives, standards, decision support, and alignment with software process improvement. This imbalance helps explain why SR remains connected to sustainability challenges. A repository or tool may support reuse locally, but sustainable reuse requires organisational routines that connect people, processes, and technology.

The lower incidence of knowledge acquisition from external sources also deserves attention. Some studies use Stack Overflow or community knowledge, but most still emphasise internal knowledge assets. This suggests a possible research path concerning how organisations evaluate and integrate external knowledge into reuse programs. Similarly, the limited attention to test reuse and application-system reuse suggests that KM–SR research remains concentrated in requirements, design, and code. Test artefacts, however, are knowledge-rich assets that can support quality and learning across projects.

Overall, the main contribution of this review is not to establish definitive trends, but to indicate possible bridges between KM and SR. Future studies could investigate how KM strategies define what counts as reusable knowledge, how evaluation processes influence trust in reusable assets, how operational tools feed organisational learning, and how feedback loops sustain reuse programs. These questions may help KM researchers engage with SR as an empirical setting for studying knowledge reuse, while also helping SE researchers make the knowledge dimension of reuse more explicit.

6. Conclusion

This paper presented an exploratory RR on how KMP are addressed in studies on SR. The review analysed 15 studies according to eight KM processes and five reuse levels. The results show that codification and storage are the most frequent processes, followed by application and sharing. Specification and code reuse are the most frequent reuse levels, while test reuse and application-system reuse are less explored.

The review suggests that SR can be understood as a practical context for studying knowledge reuse. Many studies focus on operational mechanisms, such as repositories, ontologies, knowledge graphs, traceability approaches, and code search tools. Fewer studies address the strategic conditions needed to sustain reuse over time, such as governance, incentives, standards, evaluation routines, and organisational learning. This distinction between operational and strategic perspectives offers a tentative path for connecting KM and SE research.

The study is limited by the use of Scopus only, title-based searching, English-language accessible studies, and single-researcher coding. These choices are consistent with the RR approach but limit coverage and generalisation. Future studies should include multiple databases, broader search fields, and multiple coders. Further research is also needed on knowledge acquisition from external sources, evaluation of reusable assets, test reuse, application-system reuse, and the role of KM processes in sustaining reuse programs.

Ethics declaration: Ethical clearance was not required for the development of this research.

AI declaration: AI tools such as Grammarly and ChatGPT were used solely for language revision. The authors' analysis and interpretations are their own.

References

Antovski, L., Imeri, F., 2013. Review of software reuse processes—International Journal of Computer Science Issues (IJCSI) 10.

Barros-Justo, J.L., Benitti, F.B.V., Matalonga, S., 2019. Trends in software reuse research: A tertiary study. Comput. Stand. Interfaces 66. https://doi.org/10.1016/j.csi.2019.04.011

Campos, E.C., De Souza, L.B., Maia, M. de A., 2016. Searching crowd knowledge to recommend solutions for API usage tasks. Journal of Software: Evolution and Process 28, 863–892.

Carlier, S., Naessens, V., De Backere, F., De Turck, F., 2023. A Software Engineering Framework for Reusable Design of Personalized Serious Games for Health: Development Study. JMIR Serious Games 11, e40054. https://doi.org/10.2196/40054

Cartaxo, B., Pinto, G., Soares, S., 2020. Rapid Reviews in Software Engineering.

Castro, D., Werner, C., 2021. Systematic Mapping on Software Reuse Teaching, in: 2021 12th International Conference on Information and Communication Systems (ICICS). pp. 257–264. https://doi.org/10.1109/ICICS52457.2021.9464556

Chen, X., Badampudi, D., Usman, M., 2022. Reuse in Contemporary Software Engineering Practices – An Exploratory Case Study in A Medium-sized Company. e-Informatica Software Engineering Journal 16, 220110. https://doi.org/10.37190/e-Inf220110

Costa, V., Monteiro, S., 2016. Key knowledge management processes for innovation: a systematic literature review. VINE Journal of Information and Knowledge Management Systems 46, 386–410. https://doi.org/10.1108/VJIKMS-02-2015-0017

Edwards, J., 2011. A process view of knowledge management: it ain't what you do, it's the way that you do it. Electronic Journal of Knowledge Management 9, 297–306.

Elgedawy, I., 2015. USTA: An Aspect-Oriented Knowledge Management Framework for Reusable Assets Discovery. Arabian Journal for Science and Engineering 40, 451–474. https://doi.org/10.1007/s13369-014-1428-5

Fahmideh, M., Beydoun, G., 2018. Reusing empirical knowledge during cloud computing adoption. Journal of Systems and Software 138, 124–157.

Fraga, A., Llorens, J., Génova, G., 2019. Towards a methodology for knowledge reuse based on semantic repositories. Information Systems Frontiers 21, 5–25.

Frakes, W.B., Kang, K., 2005. Software reuse research: status and future. IEEE Transactions on Software Engineering 31, 529–536. https://doi.org/10.1109/TSE.2005.85

Haddaway, N.R., Page, M.J., Pritchard, C.C., McGuinness, L.A., 2022. PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis. Campbell systematic reviews 18, e1230.

Hamdi, M.S., Ghannem, A., Kessentini, M., 2022. Requirements traceability recovery for the purpose of software reuse: an interactive genetic algorithm approach. Innovations in Systems and Software Engineering 18, 193–213. https://doi.org/10.1007/s11334-021-00418-2

Hosseingholizadeh, R., 2014. Managing the knowledge lifecycle: A integrated knowledge management process model. 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE) 102–110.

Hu, C., Ma, S., Yang, W., Sun, Z., Deng, F., Yang, Y., 2021. Software Test Data Reuse Based on Domain Ontology Construction, in: 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C). pp. 279–284.

Jabbar, T., Hafeez, Y., Kiani, A.A., Anwar, N., Javaid, J., 2019. Use of Knowledge Management and SCRUM techniques to increase the reusability in software development, in: 2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS). pp. 1–5. https://doi.org/10.1109/MACS48846.2019.9024803

Kitchenham, B., Charters, S., others, 2007. Guidelines for performing systematic literature reviews in software engineering.

Maccanti, S., Al-Jaroodi, J., Sirinterlikci, A., 2016. Knowledge management framework for software reuse, in: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). pp. 155–160.

Melo, G., Oliveira, T., Alencar, P., Cowan, D., 2020. Knowledge reuse in software projects: Retrieving software development QA posts based on project task similarity. PLoS ONE 15.

Page, M.J., McKenzie, J.E., Bossuyt, P.M., Boutron, I., Hoffmann, T.C., Mulrow, C.D., Shamseer, L., Tetzlaff, J.M., Moher, D., 2021. Updating guidance for reporting systematic reviews: development of the PRISMA 2020 statement. Journal of clinical epidemiology 134, 103–112.

Panagiotou, D., Mentzas, G., 2011. Leveraging Software Reuse with Knowledge Management in Software Development. International Journal of Software Engineering and Knowledge Engineering 21, 693–723. https://doi.org/10.1142/S0218194011005414

Polenghi, A., Roda, I., Macchi, M., Pozzetti, A., Panetto, H., 2022. Knowledge reuse for ontology modelling in Maintenance and Industrial Asset Management. Journal of Industrial Information Integration 27, 100298. https://doi.org/10.1016/j.jii.2021.100298

Putrapratama, Y.B., Adjandra, W., Wiraguna, A., Sensuse, D.I., Safitri, N., 2021. Knowledge Reuse Evaluation in Software Development: A Case Study on a Startup Company, in: 2021 Sixth International Conference on Informatics and Computing (ICIC). pp. 1–7. https://doi.org/10.1109/ICIC54025.2021.9632904

Rahman, M.M., Roy, C.K., Lo, D., 2017. Rack: Code search in the IDE using crowdsourced knowledge, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE, pp. 51–54.

Spoelstra, W., 2010. Reusing software assets in agile development organizations - a management tool: a case at a medium sized software development organization (Master's thesis). University of Twente.

Vasanthapriyan, S., Tian, J., Zhao, D., Xiong, S., Xiang, J., 2017. An ontology-based knowledge sharing portal for software testing, in: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, pp. 472–479.

Younoussi, S., Roudies, O., 2015. All about software reusability: A systematic literature review. Journal of Theoretical and Applied Information Technology 76, 64–75.