Enabling Systematic Microservices Reuse Through DevOps*

Gabriel Morais[1,2]

[1]Université du Québec à Rimouski (UQAR), 1595, boulevard Alphonse-Desjardins, Lévis (Québec), G6V 0A6, Canada
[2]Desjardins, 100, rue des Commandeurs, Lévis, (Québec), G6V 5C1, Canada

Abstract

The increasing popularity of Microservices Architecture (MSA) has resulted in a proliferation of microservices within organizations. However, this growth has led to challenges in effectively managing variants and reusing microservices, which can result in increased development time and costs. Microservices are considered reusable components by design, although a comprehensive and structured reuse-based process is required to systematically consider microservices' reuse opportunities when composing microservices architectures. This research aims to address these challenges by exploring microservices reuse practices in the context of DevOps. DevOps is the principal development methodology used in MSA, influencing what, when, and how microservices are reused and often applying unorganized and opportunistic reuse. The objectives of this research are to (1) identify, assess and organize the elements guiding effective reuse in MSA, (2) design methods to build context-aware systematic reuse processes, and (3) simplify systematic microservices reuse by leveraging DevOps automation philosophy. The proposed methodology for this research includes design science research, method engineering, and empirical industrial validation. The expected outcomes of this research are a microservices reuse reference model and methods for building modular, non-intrusive, and systematic microservices' reuse processes compliant with variable adoption of DevOps practices. We evaluate the effectiveness of these reuse models, methods and mechanisms in reducing the complexity of microservices' reuse in an industrial context.

Keywords

Microservices Architecture Composition, Microservices Reuse, Systematic Reuse, DevOps

1. Introduction

Software reuse is a key principle in software engineering that involves using existing artifacts to develop new software systems instead of starting from scratch [1], making it a cost-effective and reliable method for creating large software systems. It became a recognized concept during the 1968 NATO Software Engineering Conference [1] and a popular approach in the 2000s due to the growth of the open-source movement and the availability of reusable software at low costs [2]. It can be viewed from different perspectives, including development "for reuse," i.e., creating reusable artifacts, and development "with reuse," i.e., developing applications or systems based on existing reusable assets [3]. In this research project, we investigate reuse from the developing "with reuse" perspective and use the term "reuse" to refer to it.

Proceedings of the Doctoral Consortium Papers Presented at the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023), June 12–16, 2023, Zaragoza, Spain
gabrielglauber.morais@uqar.ca (G. Morais)

Modularity is a fundamental approach to software reuse. In 1972, David Parnas [4] introduced the concept of *modularization*, a technique that breaks down a complex software system into smaller and more manageable pieces, which are self-contained, increasing software understandability and maintainability. In 1991, Latour [5] introduced the separation of concerns in the design of reusable ADA components. Therefore, in addition to being modular, reusable components should be "embodied" by minimal content representing "exactly one concern."

Modularization and *single concern* are the building blocks of the *Microservices Architecture* (MSA) [6], which is the arrangement of *microservices* to compose distributed, resilient and scalable systems. A *microservice* is a small, autonomous service focusing on "doing one thing well" [6] and communicating through lightweight mechanisms, providing higher modularity and well-scoped functionality with a single business purpose [7]. It emerged from Agile communities [7] and is considered the confluence and evolution of component and service-based architectural approaches [8, 7]. MSAs have replaced monolithic architectures, and well-known industry cases like Amazon, Netflix and Spotify have adopted them for developing large-scale systems [9]. The adoption of DevOps [10] and Continuous Integration and Continuous Deployment (CI/CD) [11] practices often supports their development [7]. MSA is currently considered an enabler and a challenge in software reuse [12]. Some systematic reuse approaches to MSAs have been proposed [13, 14], but they have yet to consider organizational concerns and lack industrial validation.

The general objective of this research project is to improve reuse in the development life cycle of microservices architectures by designing modular systematic reuse processes that satisfy high compatibility and limited invasiveness requirements concerning organizations' development practices so that they can leverage existing microservices, handle microservices variant management, and avoid duplication. Here, organizational requirements relate to development processes, practices and culture (e.g., DevOps), legal framework, and reuse practices (e.g., ad-hoc and systematic reuse), influencing microservices architectures composition.

Building practical reuse techniques for MSA composition includes various challenges:

1. Criteria for identifying exchangeable microservices are yet to be defined [15].
2. Microservices' decentralized management [6] and the diversity in adopting DevOps practices [16, 17] make it challenging to define unified processes.
3. The mismatch of information needs and availability induced by DevOps documentation practices [18] is critical for supporting microservices retrieval and selection processes.
4. Technology diversity in MSA [6] and DevOps [17] complexifies extensive automation.
5. Moving from widely adopted ad-hoc [19] to organized and planned reuse can be intrusive and create a psychological impediment [20].

This research focuses on solving the above challenges by identifying the *essential elements* and *how* they should be applied in the MSA life cycle. From solving the specific problem of improving MSA reuse in DevOps teams, we conceive a reuse reference model tailored for MSA composition and design methods to implement modular non-intrusive systematic reuse processes in various development contexts. Our industry-sponsored project allows us to validate its outputs in real-world conditions and contribute to investigating MSA in practice. Our sponsor is a bank insurance cooperative that has 7000 IT employees working on a portfolio of 1070 active microservices, supporting 1200 applications, and serving 7.5 million members and clients.

2. Research Goals and Questions

This project's general objective is divided into three research goals: (1) Developing a reuse reference model for MSAs composition (challenge 1); (2) allowing organizations to create context-aware reuse processes (challenge 2); and (3) simplifying systematic reuse in Microservices by leveraging DevOps automation philosophy (challenges 3, 4 and 5).

2.1. Microservices Reuse Reference Model

An empirically validated software reuse reference model considering technical and organizational concerns was proposed in 2000 by Rine and Nada [21]. We intend to create a similar model for MSA, compliant with DevOps. To meet this goal, we investigate the components, processes and practices involved in managing the reuse of microservices. Mainly, we extract the factors driving the identification of reuse opportunities and decisions, and how and when they act in the development process. We start building an initial frame based on a literature review, a survey, and a case study. Collaborating with microservices practitioners (around 25) from the sponsor, we refine it and investigate real-world reuse practices to identify enablers and barriers. This approach allows us to incrementally and empirically build the reference model by defining and evaluating elements for reuse in MSA, and strategies to overcome existing barriers.

2.2. Creating Context-aware Reuse Processes

Software Product Lines (SPL) [22] have been a popular way to implement systematic reuse in software development in the last 20 years [19, 12]. It is a method for systematically reusing software assets to create a portfolio of software products for a particular application domain. It is a planned, structured, and top-down approach where reusable assets and reuse processes are carefully considered and designed, producing a clear reuse strategy, processes and governance, allowing organizations to reuse at scale but at the cost of significant structural changes. Applying SPL approaches to MSA is an open research field [15] with few solution proposals [13, 14].

Current software development practices favour "less formal" ad-hoc reuse [19]. Despite the unorganized and opportunistic characteristics of these practices, large-scale reuse of software components is still prevalent. However, they bring new challenges, including compliance with system architecture, security concerns, and a lack of guidelines for choosing appropriate elements for reuse [19]. These challenges have led to the emergence of reuse guidelines for structuring ad-hoc reuse practices based on experiences [19]. Some automatic approaches for enhanced reuse of microservices in an ad-hoc approach have been proposed [23, 24] but cover few MSA facets, are technology dependent and lack industrial validation.

Whether systematic or ad-hoc, eventually, a convergence for adopting reuse in a structured way occurs, but such adoptions are context-based, leading to a large spectrum of adopted practices [25]. This variability in adoption is a critical concern that can be exacerbated in DevOps because organizations are transitioning to it [16, 17], resulting in different maturity levels, which restrains universal solutions. Therefore, a modular approach supported by a composition method has great potential to enhance the development of adapted systematic reuse processes. Thus, we seek to define a method for tailoring processes to variable software

development contexts. We identify *what*, *when*, *where* and *how* the steps of such processes should be implemented and define guidelines to get around incompatible situations.

2.3. Simplifying Microservices Reuse by Leveraging Automated Processes

Abstraction and *automation* are essential in any reuse technique [1]. They allow developers to understand and use reusable artifacts efficiently, reducing the *cognitive distance* [1] of reuse, which measures the intellectual effort required to use an existing component than to create a new one. Indeed, an effective reuse technique reduces the cognitive distance between the reuse opportunity and realization. In the reuse process, locating, comparing, and selecting reusable artifacts, which includes classification, retrieval, and exposition mechanisms, are mandatory and complex tasks [26]. Abstraction and automation can tackle this complexity [1].

DevOps automation philosophy [10] enables abstraction and automation-based reuse processes. It can simplify microservices reuse by implementing implicit operations, including systematically gathering information from existing microservices during CI/CD processes to build knowledge bases, introducing reuse validation gatings to avoid duplicates, and making recommendations systematically to assist developers in reuse decisions.

Recommender systems [27] can tackle the above challenges and leverage DevOps automated processes by providing users with personalized recommendations based on projections of reusable microservices. It can play a critical role in assisting users in making informed reuse decisions and finding microservices arrangements using one or various microservices that best match their needs and constraints. Using recommender systems to support software reuse in Agile development processes is not new [28]; however, such an approach for MSA composition in the DevOps context has yet to be explored. We aim to identify mechanisms for automatically locating and selecting reusable microservices and guidelines to add them to the development process with limited intrusiveness.

2.4. Research Questions

To achieve the research goals, we investigate the following research questions:

RQ.1 - Which factors influence reuse when composing microservices architectures?

RQ.2 - How should reuse opportunities and reusable parts be matched?

RQ.3 - When should microservices reuse decisions be taken?

RQ.4 - How to implement microservices systematic reuse in variable development contexts?

RQ.5 - How to enhance microservices reuse without adding complexity to developers and swapping existing processes?

We consider the variability of microservices development contexts as the differences in adopting DevOps practices and supporting tools. *RQ1* and *RQ2* contribute to developing the microservices reuse reference model, *RQ3* and *RQ4* to creating context-aware reuse processes, and *RQ5* to simplifying microservices reuse by leveraging automated processes.

3. Research Design

Design science research is an accepted research methodology in the software engineering domain [29]. It is an iterative methodology that comprises two activities: Artifact design and investigation, organized through five activities: Problem investigation (1), treatment design (2), validation (3) and implementation (4), and implementation evaluation (5) [29]. It is our primary methodology for designing the following artifacts:

1. The Microservices reuse reference model (MRRM)
2. A method to design systematic reuse processes tailored for DevOps
3. A blueprint for creating reuse recommendation generators that leverage DevOps processes

The project life cycle, outlined in Fig.1, follows the Agile Unified Process (AUP) [30], which comprises four phases: Inception, evaluation, construction and transfer. Inception defines the project scope, establishes a proposed architecture, and receives sponsor approval. Elaboration proves the proposed architecture to secure the sponsor's support, and construction produces and delivers artifacts following the sponsor's priority. Finally, transition validates the system for deployment in the production environment.

We rely on SCRUM [31] to manage the project, as it is the development management method adopted by the sponsor. We adopt a product increment approach; thus, each artifact to be designed is part of the final product. This setup helps the industrial sponsor and researcher track the progress of the research realization and enhances communication between them.

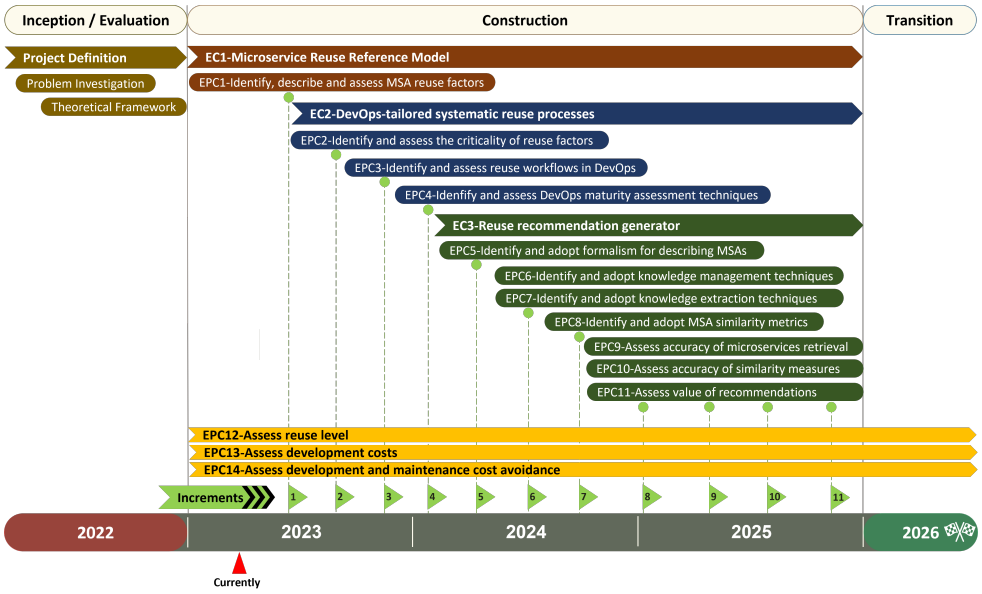

Figure 1: Research structure following the AUP and SCRUM.

3.1. Inception and Elaboration

In the *inception* and *elaboration* phases, we establish the social context, identify the design problem and requirements, establish an initial theoretical framework, conduct proof of concepts, and validate technical feasibility. The outcome of these phases is the research proposal.

3.2. Construction

The *construction* phase is the research realization. Here, we *design*, *validate*, *implement* and *evaluate* each artifact. This process is conducted through engineering cycles (EC) [29] supported by empirical cycles (EPC) [29], which respond to knowledge questions, including establishing the state-of-the-art and practice, finding and adapting techniques, and exploring and understanding a context or a problem.*Implementation* and *evaluation* tasks follows the technical action research (TAR) method [29], which involves evaluating the effect of an artifact prototype by experimenting with it in a real-world context, i.e., the sponsor's context.

3.2.1. EC1- Microservices Reuse Reference Model

The EC1 creates a reuse reference model for MSA composition, the MRRM. For the design of the MRRM, we combine design science research with the process model for empirically constructing reference models [32]. The reference model is gradually built from a frame reference derived from existing knowledge, which is used as a starting point for empirically refining the model by interacting with domain experts through enquiries. Reference models can be modelled using ontologies; thus, we will use the Ontology of Microservices Architecture Concepts (OMSAC) [33] to formalize the MRRM. If needed, we will extend OMSAC.

The empirical cycle EPC1 identifies the factors influencing microservices reuse practices and their relations to form an initial reference model frame, which will be enriched through experts' feedback. During this cycle, we prepare the experts' enquiries, by designing an initial form, enquiry plan and criteria to select participants. Then, we extend the initial reference model through refinement loops. During this process, it could be necessary to adapt the enquiries.

3.2.2. EC2- Method for Creating DevOps-tailored Systematic Reuse Processes

In the second engineering cycle, EC2, we define methods for creating systematic reuse processes based on the MRRM elements. For this purpose, we combine design science research with situational method engineering [34]. We use the paradigm-based method engineering [35], a meta-model-based approach, to construct a method for creating modular systematic reuse processes from the MRRM. Furthermore, we rely on the extension-based method engineering [35] for adapting systematic reuse processes to the context of DevOps. First, we focus on generic reuse processes not bound to a particular DevOps maturity level. Before defining this method, we need to establish the criticality of the MRRM elements (EPC2), and identify and assess reuse workflows in the standard DevOps development process (EPC3). Finally, in the empirical cycle EPC4, we identify and assess DevOps maturity standards and techniques to define criteria and tools for tailoring generic reuse processes to match DevOps maturity levels.

3.2.3. EC3- Blueprint for Reuse Recommendation Generator

The last engineering cycle, EC3, creates a blueprint for a reuse recommendation generator. First, we must define a formalism to represent the microservices architectures (EPC5). Then, we identify and adopt techniques for knowledge management (EPC6), knowledge extraction (EPC7) and similarity measurements (EPC8). The criteria to compare and classify microservices is

established during EPC8 by investigating the state of the art and practice through literature reviews and practitioners' enquiries. Empirical cycles EPC9, EPC10 and EPC11 assess the accuracy of the microservices retrieval, similarity, and value of the recommendations.

3.2.4. Evaluation

Finally, we evaluate the artifacts' effects in the TAR scope through empirical cycles EPC12, EPC13, and EPC14. We iterate between client cycles when we implement the artifacts prototypes in the sponsor context (35 teams), and empirical cycles when we analyze the evaluation data. We evaluate the effects of the artifacts on the reuse level [36] (EPC12), development costs [36] (EPC13), and development and maintenance cost avoidance [36] (EPC14). These measures are taken at different periods during the project realization, once at the beginning of the construction phase. Then, before and after each product increment, and finally in the transition phase.

3.3. Transition

This final phase is the technology transfer to the sponsor; the researcher does not intervene in the sponsor's teams' adoption and implementation of the artifacts. However, in this phase, we continue evaluating the effects and adoption of the artifacts in the long run (six months). We also conduct a qualitative enquiry to assess the usability and utility of the designed artifacts.

4. Current Status

We started this project by exploring challenges in microservices reuse faced by the sponsor, which allowed us to define the design problem and requirements. First, we established an initial theoretical framework from a literature review of reuse practices in software engineering and a systematic mapping study of reuse practices in MSA composition (to be published). Then, we used this knowledge to guide our discussions with the sponsor. Finally, we scoped the design problem, identified the project goals and formalized the associated requirements, which we translated into the three design artifacts presented previously.

We conducted two proofs of concepts to obtain the sponsor's acceptance and secure her support. First, we assessed the feasibility of our approach in the sponsor's technological environment. We investigate the data sources available and how they could be explored to build knowledge about existing microservices. Second, we investigate how to measure microservices' similarity. These proofs of concept unveiled the concern of formalizing the extracted information. We explored using an extended version of OMSAC to model microservices architectures and implemented similarity measurements from different perspectives using existing criteria and new ones from a first practitioner enquiry on similarity criteria in microservices (to be published). These novelties extend a previous work [37]. However, modelling microservices using OMSAC is a challenge for a non-ontologist; consequently, we developed a framework to guide the automation of OMSAC-based models' creation from existing documents. We implemented this framework for extracting microservices descriptions, from OpenAPI definitions, and dependencies, from Docker-compose files. We presented our results in a paper accepted to the WorldCIST'23 conference [38]. Despite this improvement, the challenge of creating a

universal parser for OMSAC remains. Consequently, we added the empirical cycle EPC5 to investigate other formalisms and compare them to OMSAC.

The output of this initial phase is an in-depth understanding of developing MSA in a real-world DevOps context. The diversity of practices and tools, and the structural role of CI/CD pipelines, unveiled new research perspectives. Indeed, federating available data to build a microservices knowledge base is a technological challenge which should be addressed case-by-case. However, building modular systematic reuse processes to be integrated into automated CI/CD pipelines is an attractive hypothesis to structure reuse in DevOps.

We are currently working on defining the reuse reference model and starting the incremental and iterative construction of the MRRM in collaboration with domain experts, including developers (15), tech leads (5) and systems (5) architects from the sponsor, and researchers from different organizations (5).

5. Conclusion and Expected Contributions

This paper presented our proposal for a modular, non-intrusive and systematic microservices' reuse process tailored to comply with DevOps practices currently employed in the MSA development life cycle. The systematic reuse process is derived from mandatory and optional elements of a defined Microservices reuse reference model, the MRRM, and tailored to the target DevOps context. We propose using recommender systems deployed into existing automatic integration and deployment pipelines to limit intrusiveness in the developer tasks, generalize the reuse opportunities detection and assist in reuse decisions.

By developing such a reuse approach, we contribute to research and practice in the following way: First, the MRRM provides a common understanding of reuse in MSA composition. Second, the modular construction of systematic reuse processes provides tools to accommodate structured reuse and various DevOps-adopted practices. It could provide insights to explain how DevOps influences current reuse practices in MSA. Third, achieving a non-intrusive process throughout DevOps automated pipelines can contribute to understanding their structural role in the MSAs' life cycle. Fourth, choosing a formalism to describe MSAs and the criteria to retrieve and compare them could be effective solutions to open challenges in microservices reuse. Finally, the artifacts developed in this project will eventually be evaluated in an industrial context contributing to filling the gap of empirical industrial validations in the MSA domain.

Acknowledgments

I want to thank Prof. Dr. Mehdi Adda (mehdi_adda@uqar.ca) from the Université du Québec à Rimouski, Canada, and Prof. Dr. Dominik Bork (dominik.bork@tuwien.ac.at) from the Technische Universität Wien, Austria, for the supervision of this thesis.

References

[1] C. W. Krueger, Software reuse, ACM Computing Surveys (CSUR) 24 (1992) 131–183.
[2] I. Sommerville, Software Engineering, Always learning, 10. ed., Pearson, 2016.

[3] J. L. Barros-Justo, F. B. Benitti, S. Matalonga, Trends in software reuse research: A tertiary study, Computer Standards & Interfaces 66 (2019) 103352.
[4] D. L. Parnas, On the criteria to be used in decomposing systems into modules, Communications of the ACM 15 (1972) 1053–1058.
[5] L. Latour, A methodology for the design of reuse engineered ada components, in: Proceedings of the first international symposium on Environments and tools for Ada, 1991, pp. 103–113.
[6] S. Newman, Building microservices, " O'Reilly Media, Inc.", 2021.
[7] O. Zimmermann, Microservices tenets, Computer Science-Research and Development 32 (2017) 301–310.
[8] E. Yuan, Architecture interoperability and repeatability with microservices: an industry perspective, in: Proceedings of the 2nd International Workshop on Establishing a Community-Wide Infrastructure for Architecture-Based Software Engineering, 2019, pp. 26–33.
[9] J. Soldani, D. A. Tamburri, W.-J. Van Den Heuvel, The pains and gains of microservices: A systematic grey literature review, Journal of Systems and Software 146 (2018) 215–232.
[10] M. Httermann, DevOps for developers, Apress, 2012.
[11] J. Humble, D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, 2010.
[12] R. Capilla, B. Gallina, C. Cetina, J. Favaro, Opportunities for software reuse in an uncertain world: From past to emerging trends, Journal of Software: Evolution and process 31 (2019) e2217.
[13] M. R. Setyautami, H. S. Fadhlillah, D. Adianto, I. Affan, A. Azurat, Variability management: Re-engineering microservices with delta-oriented software product lines, in: Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A-Volume A, 2020, pp. 1–6.
[14] W. D. Mendonça, W. K. Assunção, L. V. Estanislau, S. R. Vergilio, A. Garcia, Towards a microservices-based product line with multi-objective evolutionary algorithms, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.
[15] W. K. Assunção, J. Krüger, W. D. Mendonça, Variability management meets microservices: six challenges of re-engineering microservice-based webshops, in: Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A, 2020, pp. 1–6.
[16] L. E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, C. Lassenius, Devops in practice: A multiple case study of five companies, Information and Software Technology 114 (2019) 217–230.
[17] GitLab, "the gitlab 2022 global devsecops survey," thriving in an insecure world, https://about.gitlab.com/developer-survey/#developers, 2022. (Accessed on 08/29/2022).
[18] U. Van Heesch, T. Theunissen, O. Zimmermann, U. Zdun, Software specification and documentation in continuous software development: a focus group report, in: Proceedings of the 22nd European Conference on pattern languages of programs, 2017, pp. 1–13.
[19] N. Mäkitalo, A. Taivalsaari, A. Kiviluoto, T. Mikkonen, R. Capilla, On opportunistic software reuse, Computing 102 (2020) 2385–2408.
[20] D. C. Schmidt, Why software reuse has failed and how to make it work for you, C++ Report 11 (1999) 1999.

[21] D. C. Rine, N. Nada, An empirical study of a software reuse reference model, Information and Software Technology 42 (2000) 47–65.
[22] P. Clements, L. Northrop, Software product lines, Addison-Wesley Boston, 2002.
[23] S.-P. Ma, C.-Y. Fan, Y. Chuang, I.-H. Liu, C.-W. Lan, Graph-based and scenario-driven microservice analysis, retrieval, and testing, Future Generation Computer Systems 100 (2019) 724–735.
[24] C.-a. Sun, J. Wang, Z. Liu, Y. Han, A variability-enabling and model-driven approach to adaptive microservice-based systems, in: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2021, pp. 968–973.
[25] M. Antkiewicz, W. Ji, T. Berger, K. Czarnecki, T. Schmorleiz, R. Lämmel, S. Stănciulescu, A. Wąsowski, I. Schaefer, Flexible product line engineering with a virtual platform, in: Companion Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 532–535.
[26] R. Prieto-Diaz, P. Freeman, Classifying software for reusability, IEEE software 4 (1987) 6.
[27] F. Ricci, L. Rokach, B. Shapira, Recommender Systems: Techniques, Applications, and Challenges, Springer US, New York, NY, 2022, pp. 1–35.
[28] F. McCarey, M. Ó. Cinnéide, N. Kushmerick, Rascal: A recommender agent for agile reuse, Artificial Intelligence Review 24 (2005) 253–276.
[29] R. J. Wieringa, Design science methodology for information systems and software engineering, Springer, 2014.
[30] S. Ambler, The agile unified process (aup), http://www.ambysoft.com/unifiedprocess/agileUP.html, 2005. (Accessed on 09/07/2021).
[31] K. Schwaber, J. Sutherland, The scrum guide, Scrum Alliance 21 (2011) 1–38.
[32] F. Ahlemann, H. Gastl, Process model for an empirically grounded reference model construction, in: Reference modeling for business systems analysis, IGI Global, 2007, pp. 77–97.
[33] G. Morais, M. Adda, Omsac-ontology of microservices architecture concepts, in: 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), IEEE, 2020, pp. 0293–0301.
[34] S. Brinkkemper, Method engineering: engineering of information systems development methods and tools, Information and software technology 38 (1996) 275–280.
[35] J. Ralyté, R. Deneckère, C. Rolland, Towards a generic model for situational method engineering, in: Advanced Information Systems Engineering: 15th International Conference, CAiSE 2003 Klagenfurt/Velden, Austria, June 16–20, 2003 Proceedings 15, Springer, 2003, pp. 95–110.
[36] W. Frakes, C. Terry, Software reuse: metrics and models, ACM Computing Surveys (CSUR) 28 (1996) 415–435.
[37] G. Morais, D. Bork, M. Adda, Towards an ontology-driven approach to model and analyze microservices architectures, in: Proceedings of the 13th International Conference on Management of Digital EcoSystems, 2021, pp. 79–86.
[38] G. Morais, M. Adda, H. Hadder, D. Bork, x2omsac – an ontology population framework for the ontology of microservices architecture concepts, in: WorldCist'23 - 11st World Conference on Information Systems and Technologies, To Appear, p. in press.