SERVICE COMPUTATION 2023

The Fifteenth International Conferences on Advanced Service Computing

ISBN: 978-1-68558-043-8

June 26 - 30, 2023

Nice, France

SERVICE COMPUTATION 2023 Editors

Anders Fongen, Norwegian Defence University College, Norway

SERVICE COMPUTATION 2023

Forward

The Fifteenth International Conferences on Advanced Service Computing (SERVICE COMPUTATION 2023), held on June 26 - 30, 2023, continued a series of events targeting computation on different facets.

The ubiquity and pervasiveness of services, as well as their capability to be context-aware with (self-) adaptive capacities posse challenging tasks for services orchestration, integration, and integration. Some services might require energy optimization, some might require special QoS guarantee in a Web-environment, while others a certain level of trust. The advent of Web Services raised the issues of self-announcement, dynamic service composition, and third party recommenders. Society and business services rely more and more on a combination of ubiquitous and pervasive services under certain constraints and with particular environmental limitations that require dynamic computation of feasibility, deployment and exploitation.

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the SERVICE COMPUTATION 2023 technical program committee, as well as the numerous reviewers. The creation of a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to SERVICE COMPUTATION 2023. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the SERVICE COMPUTATION 2023 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope SERVICE COMPUTATION 2023 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of computation. We also hope that Nice provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city.

SERVICE COMPUTATION 2023 Steering Committee

Paul Humphreys, Ulster Business School/University of Ulster, UK
Arne Koschel, Hochschule Hannover, Germany
Annett Laube, Bern University of Applied Sciences (BUAS), Switzerland
Eugen Borcoci, University "Politehnica" of Bucharest, Romania

Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Hannover, Germany
Ozgu Can, Ege University, Turkey

SERVICE COMPUTATION 2023 Publicity Chair

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

SERVICE COMPUTATION 2023

Committee

SERVICE COMPUTATION 2023 Steering Committee

Paul Humphreys, Ulster Business School/University of Ulster, UK
Arne Koschel, Hochschule Hannover, Germany
Annett Laube, Bern University of Applied Sciences (BUAS), Switzerland
Eugen Borcoci, University "Politehnica" of Bucharest, Romania
Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Hannover, Germany
Ozgu Can, Ege University, Turkey

SERVICE COMPUTATION 2023 Publicity Chairs

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

SERVICE COMPUTATION 2023 Technical Program Committee

Jocelyn Aubert, Luxembourg Institute of Science and Technology (LIST), Luxembourg
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Eugen Borcoci, University "Politehnica" of Bucharest, Romania
Uwe Breitenbücher, University of Stuttgart, Germany
Antonio Brogi, University of Pisa, Italy
Isaac Caicedo-Castro, Universidad de Córdoba, Colombia
Ozgu Can, Ege University, Turkey
Rong N. Chang, IBM T.J. Watson Research Center, USA
Dickson Chiu, The University of Hong Kong, Hong Kong
Leandro Dias da Silva, Universidade Federal de Alagoas, Brazil
Erdogan Dogdu, Angelo State University, USA
Monica Dragoicea, University Politehnica of Bucharest, Romania
Xinsong Du, University of Florida, USA
Sebastian Floerecke, University of Passau, Germany
Stefano Forti, University of Pisa, Italy
Sören Frey, Daimler TSS GmbH, Germany
Steffen Fries, Siemens Corporate Technology - Munich, Germany
Somchart Fugkeaw, Sirindhorn International Institute of Technology | Thammasat University, Thailand
Katja Gilly, Miguel Hernandez University, Spain
Victor Govindaswamy, Concordia University - Chicago, USA
Maki Habib, The American University in Cairo, Egypt
Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Germany
Bernhard Hollunder, Hochschule Furtwangen University - Furtwangen, Germany
Wladyslaw Homenda, Warsaw University of Technology, Poland
Tzung-Pei Hong, National University of Kaohsiung, Taiwan

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Towards Patterns for Choreography of Microservices-based Insurance Processes

Alexander Link
Henrik Meyer
Christin Schulze
Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hanover, Germany
Email: andreas.hausotter@hs-hannover.de

Andreas Hausotter
Arne Koschel

Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hanover, Germany
Email: arne.koschel@hs-hannover.de

***Abstract*—To avoid the shortcomings of traditional monolithic applications, the Microservices Architecture (MSA) style plays an increasingly important role in providing business services. This is especially true for the insurance industry with its sophisticated cross-domain business processes. Here, the question arises of how workflows can be implemented to grant the required flexibility and agility and, on the other hand, exploit the MSA style's potential. There are two competing approaches to workflow realization, orchestration, and choreography, each with pros and cons. Though choreography seems to be the method of choice in MSA, it comes with some challenges. As the workflow is implicit – it evolves as a sequence of events being sent around – it gets hard to understand, change, or operate the workflow. To manage the challenges of the choreography approach, we use BPMN 2.0 choreography diagrams to model the exchange of domain events between microservices, which represent 'participants' in terms of BPMN. We aim to execute choreography diagrams automatically. For this, we developed a set of choreography patterns that represent frequently occurring sequences. We present the pattern language and discuss two patterns, a One-Way Task pattern, and a Event-based Gateway – Deadline pattern. This paper is part of our ongoing research to design a microservices reference architecture for insurance companies.**

Keywords*—*Workflow; Choreography; BPMN; Patterns; Business Processes; Microservice.

I. INTRODUCTION

Business workflows and multistep business processes are typical for insurance companies; see, for example, the reference architecture for German insurance companies (VAA) [1]. They are complemented by general regulations, such as the European GDPR [2], as well as insurance-specific laws and rules regarding, for example, financial regulations, data protection, and security [3].

Recently, the Microservices Architecture (MSA) style [4] [5] and cloud computing [6] became more and more interesting for insurance companies. Traditionally, several technologies from monolithic mainframe applications, functional decomposition-based software, traditional Service-Oriented Architectures (SOAs), which often utilize some kind of Enterprise Service Bus (ESB), Business Process and Workflow Management Systems (BPMS, WfMS) for orchestration, and 3rd party software, such as SAP software, were and are used together in insurance business applications, which implement their business processes.

Taking all those typical cornerstones from (over time grown) insurances into account, the goal of our currently ongoing research [7] is to develop a 'Microservice Reference Architecture for Insurance Companies (RaMicsV)' jointly with partner companies from the insurance domain. Within our work, we also look at the question: 'how to implement (insurance) business workflows with microservices, which potentially utilize several logical parts from RaMicsV'?

Within the MSA style, the more decoupled choreography is favored for this purpose [4] [5]. This is in some contrast, however, for example, to SOAs, where such workflows are mainly implemented using orchestration [8]. For example, one of our partner companies utilizes Camunda [9], another one a Java/Jakarta EE-based workflow tool.

However, since co-existence of all approaches is a 'must have' for our insurance partner companies, RaMicsV aims to address the *combined usage* of more traditional approaches and the MSA style, the combination of choreography and orchestration naturally comes to mind. As evolution is a key demand for our business partners – they can and will not just 'throw away' their existing application landscape – concepts such as orchestration and tools such as an ESB, whose use within MSA style architectures are both clearly disputable, have to be integrated reasonably well into our approach.

We thus started to look at the *combination* of choreography and orchestration, including a look at insurance domain specifics, in our work from [10]. In the present article, we will now have a focus on choreography-based approaches for (insurance) business processes. Particularly, we will examine an initial set of emerged *choreography patterns* for this purpose, which we will model using choreography diagrams from the OMG BPMN 2.0 standard [11]. It should be noted that our goal is not a general implementation of choreographies, rather an implementation that orients itself toward real-world scenarios. Thus, we inspected multiple use cases from the insurance industry, one of which we will introduce later on.

In particular, we contribute in the present article our ongoing work and intermediate results about:

- The integration of the choreography within our RaMicsV;

 ISBN: 978-1-68558-043-8

- BPMN 2.0 choreography diagrams and the utilization of patterns;
- our pattern language for choreography patterns;
- two particular choreography patterns in depth, namely the One-Way Pattern and the Event-based Gateway – Deadline pattern;
- and finally insurance business use cases for those patterns.

The remainder of this article is structured as follows: After discussing related work in Section II, we briefly look at our current work within the RaMicsV context in Section III. Next, Section IV looks at BPMN 2.0 choreography diagrams with patterns. Section V then contributes our patterns usage and a pattern language for them, as well as two identified patterns. Moreover, Section VI looks at a usage of those patterns within an insurance business use case. Finally, Section VII summarizes our results and concludes with some outlook to future work, with more patterns to follow.

II. Related Work

The basis of our research builds on authors in the scope of microservices, such as the work from Newman [5], as well as Fowler and Lewis [12]. Within the design of our reference architecture, we profit from different microservices patterns, as they are discussed by Krause [13] and Richardson [4].

To model our business processes, we use OMG's BPMN 2.0 specification. Also, we use as groundwork about business processes and its development with BPMN the works from Allweyer [14] [15], Rücker and Freund [16].

For the basics of service composition types, orchestration and choreography, we chose to rely on Decker's approach [17]. It is important that we define the choreography in terms of workflows within a microservices architecture. Quite many publications discuss the benefits of the choreography as a composition between (micro-)services. In particular, in several cases the theoretical benefit is presented or the combination of different approaches with the choreography is shown, as discussed by Rücker in his blog [18].

This paper ties in with our previous work on realizing a choreography [10]. In our last paper, we experimented with the implementation of a choreography using BPMN. The first pattern 'Any Problem becomes a Service' appeared to be difficult, since the monolithic BPMN does not support the message exchange between different microservices.

In Mikalkinas' [19] approach, a BPMN choreography diagram is transformed into a BPMN collaboration diagram and then executed. After this transformation, the BPMN collaboration diagram is executed by an engine, in this case Camunda [9]. We intend to bypass this conversion and provide direct execution of the choreography diagram. Thereby, our goal is to explore an implementation without an engine, since this corresponds to an orchestration in the case of Camunda.

Milanović and Gasević also try to implement choreography via BPMN and REWERSE II Rule Markup Language in their work [20]. They developed a rule-based extension for BPMN to realize choreography, called rBPMN. Ortiz et al. describe a similar approach [21]: In their work, rules are also defined on how to react based on which events in a choreography. This work uses fragments of BPMN. In both approaches (only) parts of the BPMN are considered, and in each case, only collaboration diagrams.

Another related approach is Richardson's SAGA pattern and the Eventuate Framework [4] [22]. The pattern describes the splitting of a transaction into several small local transactions. The local transactions trigger each other by messages/events. The error handling could become interesting for our further work. The framework includes two manifestations: Tram and Local. Eventuate Tram [23] so far only implements an orchestrated SAGA, so it does not yet include a choreography. Eventuate Local [24] provides event sourcing to store events. It also offers functions to perform transactions, through a publish/subscribe realization. It maps the technical implementation of a transaction rather than the communication and composition between services.

We try to implement a choreography in a more straight way as a compositional approach between microservices. Our vision is to use the choreography for the complete communication and workflow. We define the choreography as a global approach to processing a workflow without the intervention of a controlling part. This approach was described by us in our previous paper [10] and is also defined by Decker [17].

To achieve this goal, we define patterns for BPMN choreography diagrams, which are supposed to be implemented automatically. To model our BPMN choreography diagrams we used the framework chor-js developed from Ladleif et al. [25]. We do not focus on the processes within the (micro-) services themselves, rather only on the communication between them and the infrastructure. The use of patterns should also mitigate to some degree the complexity that can arise in (extensive) choreography-based workflows. The developed patterns borrow in structure and approach from Barros et al. [26].

III. Service-based Reference Architecture for Insurance Companies

This Section presents our logical reference architecture for microservices in the insurance industry (RaMicsV) as initially started in [7].

RaMicsV defines the setting for the architecture and the design of a microservices-based application for our industry partners. The application's architecture will only be shown briefly, as it heavily depends on the specific functional requirements.

When designing RaMicsV, a wide range of restrictions and requirements given by the insurance company's IT management have to be considered. Regarding this contribution, the most relevant are:

- Enterprise Service Bus (ESB): The ESB as part of the SOA must not be questioned. It is part of a successfully

 ISBN: 978-1-68558-043-8

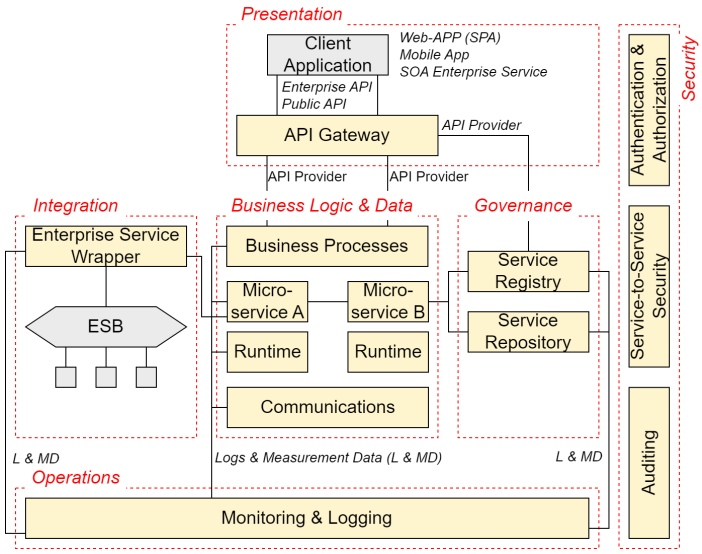

Figure 1. Building Blocks of the Logical Reference Architecture RaMicsV.

operated SOA landscape, which seems suitable for our industry partners for several years to come. Thus, from their perspective, the Microservices Architecture (MSA) style is only suitable as an additional enhancement and only a partial replacement of parts from their SOA or other self-developed applications.

- Coexistence: Legacy applications, SOA, and microservices-based applications will be operated in parallel for an extended transition period. This means that RaMicsV must provide approaches for integrating applications from different architecture paradigms – looking at it from a high-level perspective, allowing an 'MSA style best-of-breed' approach at the enterprise architectural level as well.
- Business processes are critical elements in an insurance company's application landscape. To keep their competitive edge, the enterprise must change their processes in a flexible and agile manner. RaMicsV must therefore provide suitable solutions to implement workflows while ensuring the required flexibility and agility.

Figure 1 depicts the building blocks of RaMicsV which comprises layers, components, interfaces, and communication relationships. Components of the reference architecture are colored yellow; those out of scope are greyed out.

A component may be assigned to one of the following *responsibility areas*:

- **Presentation** includes components for connecting clients and external applications such as SOA services.
- **Business Logic & Data** deals with the implementation of an insurance company's processes and their mapping to microservices, using various workflow approaches to achieve desired application-specific behavior.
- **Governance** consists of components that contribute to meeting the IT governance requirements of our industrial partners.
- **Integration** contains system components to integrate microservices-based applications into the industrial partner's application landscape.
- **Operations** consist of system components to realize unified monitoring and logging, which encloses all systems of the application landscape.
- **Security** consists of components to provide the goals of information security, i.e., confidentiality, integrity, availability, privacy, authenticity & trustworthiness, nonrepudiation, accountability, and auditability.

Components communicate either via HTTP(S) – using a RESTful API, or message-based – using a Message-Oriented Middleware (MOM) or the ESB. The ESB is part of the integration responsibility area, which itself contains a message broker (see Figure 1).

In the next Section, we will have a look at the choreography in general and BPMN 2.0 choreography in particular as a lead-in to this paper's contribution, located in the responsibility area *Business Logic & Data*.

IV. Choreography

This Section will present the core definition of choreography, as described in [10]. We briefly outline the use of BPMN, specifically the choice of BPMN 2.0 choreography diagrams.

A. Choreography

In a choreographed system, there exists no central coordinator, unlike in orchestration [27]. Decker [17] describes the definition of a choreography as a global view of how services cooperate and the interaction between participants. This proves to be a challenge when modeling and monitoring a workflow, as the workflow is mapped by the interaction between the participants. It follows that the responsibility of executing and processing the workflow is transferred to each participant [28].

While choreography may be combined with other patterns, like the event-driven architecture [29], we decided not to focus on technical implementations yet, but will eventually.

B. BPMN 2.0 choreography

BPMN 2.0 choreography is chosen as the modeling language, since BPMN is also used by our partners. In the BPMN specification exist at least three significantly different diagram types to describe processes:

- **Process** known as classic BPMN. It visualizes the entire process.
- **Collaboration** splits a classic process into multiple participants (or microservices). Each sub-process in a participant can be recognized, but also the message exchange between the participants.
- **Choreography** which visualizes only the exchange of messages between participants.

 ISBN: 978-1-68558-043-8

In contrast to our previous work [10], we now focus only on the implementation of BPMN 2.0 choreography diagrams [11], as they visualize the interaction between microservices. In these diagrams, a participant represents a microservice. We aim to execute business processes using a choreographed MSA. Choreography serves as a global composition pattern [17]. We start with a collaboration diagram to map the whole process, which we then transform into a choreography diagram to focus on the communication. The processes within the participants are out of scope as we focus on the means of communication.

To automatically implement the choreography with BPMN 2.0 choreography, we develop patterns that map frequently occurring sequences. It should be a wide selection of things that must, should or can occur. The pattern language and the yet-to-be-developed grammar will be used to create a tool that automatically accepts modeled choreography diagrams and generates the necessary infrastructure and message exchange.

V. Choreography Patterns

In this Section, we will present a pattern language, as well as two patterns from our list. The language intends patterns to be assembled to realize more extensive use cases. The patterns originate from real-world use cases.

A. Pattern Language

A pattern language is utilized to describe the patterns uniformly. It consists of the following elements (cf. [6]):

- **Identification number (ID)** of the pattern.
- **Name** of the pattern.
- **Figures** that visualize the pattern. Consisting of BPMN 2.0 choreography diagrams, BPMN collaboration diagrams, and UML Sequence diagrams.
- A **Description** which describes the use, content, and flow of the pattern.
- **Rules** and conditions under which the pattern may be used.
- A list of **used BPMN elements** from the choreography- and collaboration diagrams, as named in [11].
- **Used Patterns**, which this pattern builds upon.
- **Synonyms** and similar patterns from literature and industry.
- **Variations** where the core concept of the pattern stays the same.
- **Typical combinations** and patterns with high compatibility.
- Example **Use-Cases** from the industry.

B. One-Way Task

Now that the pattern language has been introduced, we start with the most atomic pattern, the *One-Way Task*.

- **ID:** BPMNChor01
- **Name:** One-Way Task
- **Figures:** See Figure 2, Figure 3, and Figure 4.

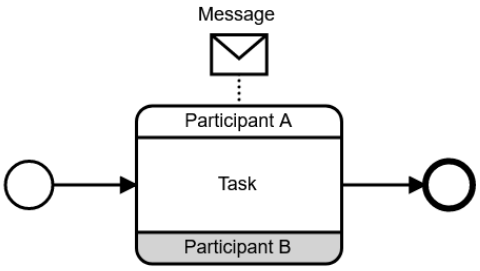

Figure 2. One-Way Task Choreography.

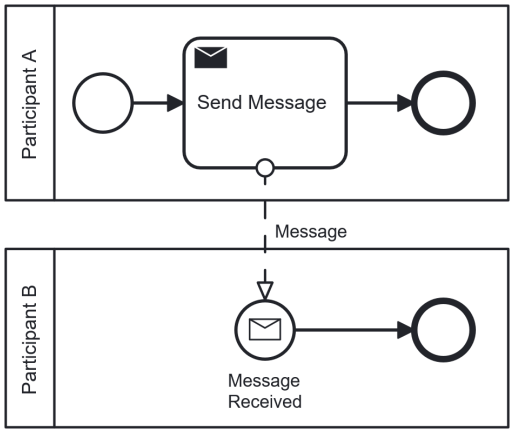

Figure 3. One-Way Task Collaboration.

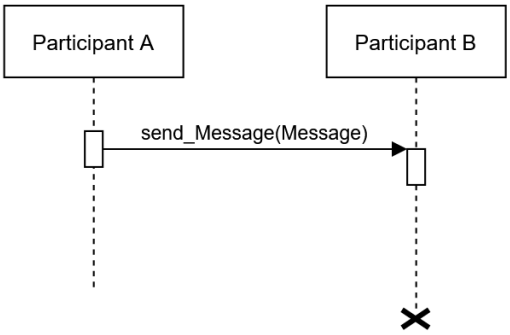

Figure 4. One-Way Task UML Sequence.

- **Description:** Participant A wants to deliver a message to Participant B. The initiator (A) sends the message to the receiver (B).
- **Rules:** None.
- **Used BPMN Elements:** startEvent (none), messageStartEvent, participant (pool), Message originating from the initiator, endEvent (none).
- **Used Patterns:** None, this pattern is atomic and depicts the minimum amount of interaction.
- **Synonyms:** Fire-and-Forget, One-Way Notification
- **Variations:** None.

 ISBN: 978-1-68558-043-8

- **Typical combinations:** Due to the atomic properties of this pattern, it may be combined with every other pattern.
- **Use-Case:** Sending an E-Mail or push-notification. For a longer scenario, see Section VI

This concludes the One-Way Task as the minimal way of communication, next we will introduce Event-based Gateway – Deadline pattern.

C. Event-based Gateway – Deadline

The *Event-based Gateway – Deadline* pattern describes a more complex, yet often occurring, scenario where the flow of a process is determined by a temporal aspect.

- **ID:** BPMNChor11
- **Name:** Event-based Gateway – Deadline
- **Figures:** See Figure 5, Figure 6, and Figure 7.
- **Description:** An answer only has a limited time frame to be received. Participant B receives a message from Participant A. Participant B has to answer within a given timeframe (N-Time) or else another workflow will be triggered. Participant A has the timing responsibility.
- **Rules:** Participant B has to initiate the answering message. A Two-Way communication is required.
- **Used BPMN Elements:** startEvent (none), messageStartEvent, participant (pool), Message, originating from the initiator, messageStartEvent, timerStartEvent, endEvent (none).
- **Used Patterns:** This pattern is based upon the *Sequence Flow – Two Participants* pattern (to be published) with the restriction that the receiving participant has to answer in the given timeframe.
- **Synonyms:** Asynchronous Request-Response
- **Variations:** None.
- **Typical combinations:** This pattern may be inserted into any request-response workflow when a timing-based component is needed.
- **Use-Case:** Setting a Deadline for paying an invoice. If the time is over, a reminder may be sent. For a longer scenario, see Section VI.

VI. Pattern Scenarios in Insurance Companies

To realize the pattern language of the two introduced patterns in Section V completely, this Section evaluates use cases of the patterns from the insurance industry.

We consider a typical process where a new insurance application is managed. The process *New Insurance Application* adopted from Freund and Rücker [16], but can also be taken directly from the insurance business model of our partners in the insurance industry, thus mapping a real-world use case. Due to the size of the process, it is only briefly described below and the parts containing the patterns are further explained.

In the process, a customer submits a new insurance application. If the request is rejected, this information is noted in the backend and the customer is informed. If the request is accepted, a policy is created. After creation, the policy is sent

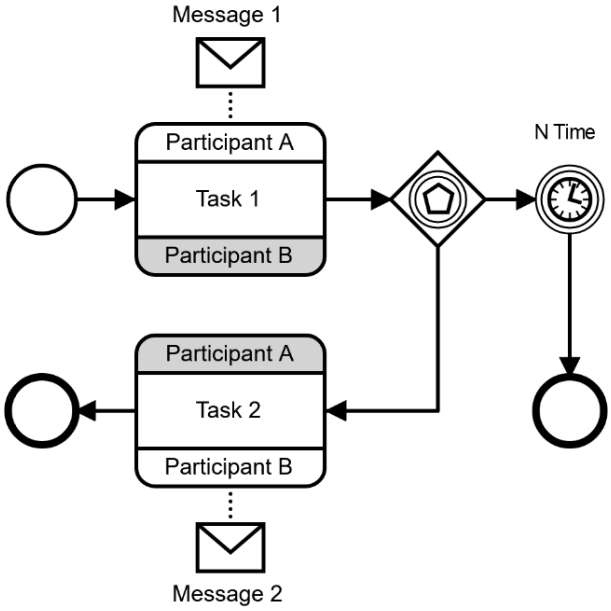

Figure 5. Event-based Gateway – Deadline Choreography.

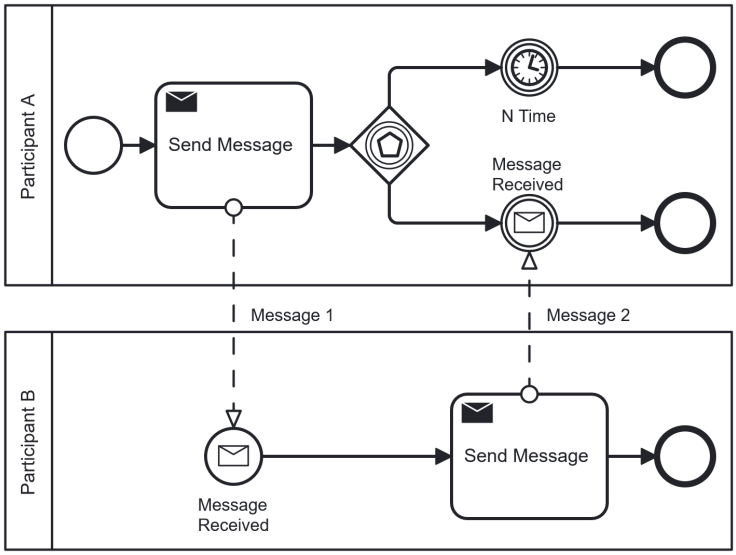

Figure 6. Event-based Gateway – Deadline Collaboration.

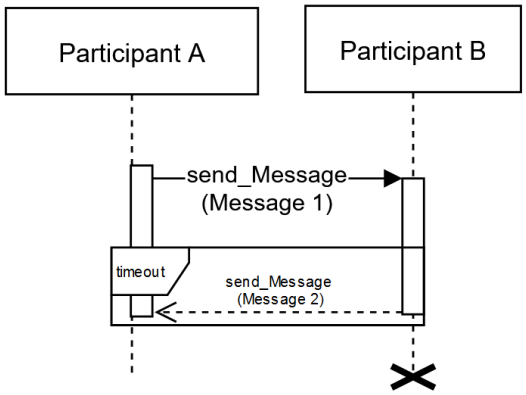

Figure 7. Event-based Gateway – Deadline UML Sequence.

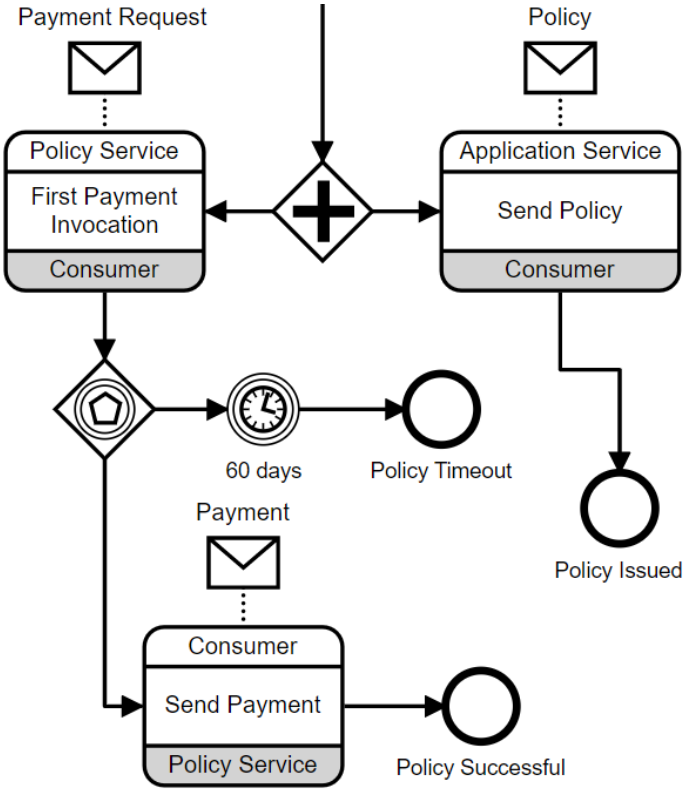

Figure 8. New Insurance Application Process – Cutout.

and the customer is requested to submit the first payment. If the payment is not made within 60 days, the request, and the policy are invalid. If the customer pays in time, the insurance is valid.

The parallel flow represents the examples of the use cases in the insurance industry. In this example, both use cases are separated by a parallel gateway, as shown in the Figure 8. The parallel gateway has not yet been introduced as a pattern; in this implementation, it visualizes the (almost) parallel flow of the two messages. In one path, the *One-Way Task* pattern is represented, by sending the policy to the customer. In the other path, the *Event-based Gateway – Deadline* pattern is utilized by the sending and receiving of the payment request.

In the right path (see Figure 8), the *One-Way Task* pattern is implemented. The application service sends the policy to the client. After sending, the task is completed and the path ends.

The *Event-based Gateway – Deadline* pattern is shown in the left path. The policy service sends the first payment request to the client. Then a timer is started. If the customer pays within 60 days, the policy, and the process are successful. If the customer does not pay within 60 days, a timeout occurs and the policy becomes invalid.

As shown with the payment request and the incoming payment in Figure 8, the *Event-based Gateway — Deadline* pattern contains the *One-Way Task* pattern. It shows that this fundamental pattern is the basis of the minimal communication for the choreography.

VII. Conclusion and Future Work

The effective modeling and implementation of business processes is of crucial importance for an insurance company. Coming from BPMN notation, there needs to be a concise way of realizing the modeled process in the MSA style using the choreography. In this article, we presented the beginning of our choreography pattern language as the first steps towards a clear realization approach with precise implementation rules to map from BPMN diagrams to the distribution of microservices. For realizing a pattern language, a grammar will be developed and evaluated in in future work.

Several more patterns are needed to cover a broader range of different business use cases in the insurance industry. We also plan to evaluate all theoretical patterns with our insurance industry partners to ensure practical use. In future work, we will thus present additional patterns and grammar, including usage examples for them. We will also aim to refine our choreography pattern language and evaluate its additional benefit through a concrete implementation.

References

[1] Gesamtverband der Deutschen Versicherungswirtschaft e.V. - General Association o.t. German Insurance Industry, “VAA Final Edition. Das Fachliche Komponentenmodell (VAA Final Edition. The Functional Component Model),” 2001.

[2] European GDPR, “Complete guide to GDPR compliance,” Online. Available: https://gdpr.eu/ [retrieved: 04, 2023].

[3] Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) - Federal Financial Supervisory (BaFin), “Versicherungsaufsichtliche Anforderungen an die IT (VAIT) (Insurance Supervisory Requirements for IT (VAIT)) vom 03.03.2023,” 2023, Online. Available: https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Meldung/2023/meldung_2023_03_03_Aktualisierung_VAIT.html [retrieved: 04, 2023].

[4] C. Richardson, *Microservices Patterns: With examples in Java*. Shelter Island, New York: Manning Publications, 2018.

[5] S. Newman, *Building microservices: designing fine-grained systems*. Sebastopol, California: O’Reilly Media, Inc., 2015.

[6] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns Fundamentals to Design, Build, and Manage Cloud Applications*. Springer Vienna, 2014.

[7] A. Koschel, A. Hausotter, R. Buchta, A. Grunewald, M. Lange, and P. Niemann, “Towards a Microservice Reference Architecture for Insurance Companies,” in *SERVICE COMPUTATION 2021, 13th Intl. Conf. on Advanced Service Computing*. IARIA, ThinkMind, 2021, pp. 5–9, Online. Available: https://www.thinkmind.org/articles/service_computation_2021_1_20_10002.pdf [retrieved: 04, 2023].

[8] A. Hausotter, A. Koschel, M. Zuch, J. Busch, and J. Seewald, “Components for a SOA with ESB, BPM, and BRM – Decision Framework and architectural Details,” *Intl. Journal od Advances in Intelligent Systems*, vol. 9, no. 3 & 4, pp. 287–297, 2016.

[9] “Workflow and decision automation platform,” Nov 2021, Online. Available: https://camunda.com/ [retrieved: 04, 2023].

[10] A. Koschel, A. Hausotter, R. Buchta, C. Schulze, P. Niemann, and C. Rust, “Towards the Implementation of Workflows in a Microservices Architecture for Insurance Companies – The Coexistence of Orchestration and Choreography,” in *SERVICE COMPUTATION 2023, 14th Intl. Conf. on Advanced Service Computing*. IARIA, ThinkMind, 2023, pp. 1–5, Online. Available: https://www.thinkmind.org/index.php?view=article&articleid=service_computation_2023_1_10_10002 [retrieved: 04, 2023].

[11] OMG, *Business Process Model and Notation (BPMN), Version 2.0*, Object Management Group Std., Rev. 2.0, January 2011, Online. Available: http://www.omg.org/spec/BPMN/2.0 [retrieved: 04, 2023].

 ISBN: 978-1-68558-043-8

[12] M. Fowler and J. Lewis, “Microservices a definition of this new architectural term,” 2014, Online. Available: https://martinfowler.com/articles/microservices.html [retrieved: 04, 2023].
[13] L. Krause, *Microservices: Patterns and Applications: Designing fine-grained services by applying patterns*. Lucas Krause, 2015.
[14] T. Allweyer, *Kollaborationen,Choreographien und Konversationen in BPMN 2.0 - Erweiterte Konzepte zur Modellierung übergreifender Geschäftsprozesse - Collaborations,Choreographies and Conversations in BPMN 2.0 - Advanced Concepts for Modeling Comprehensive Business Processes*. Fachhochschule Kaiserslautern, 2009.
[15] T. Allweyer, *Geschäftsprozessmanagement: Strategie, Entwurf, Implementierung, Controlling. - Business process management: strategy, design, implementation, controlling.* W3l GmbH, 2005.
[16] B. Rücker and J. Freund, *Praxishandbuch BPMN 2.0 - Practice Handbook BPMN 2.0*. Carl Hanser Verlag München Wien, 2014.
[17] G. Decker, O. Kopp, and A. Barros, *An Introduction to Service Choreographies*, vol. 50, no 2 ed. Information Technology, 2008.
[18] B. Rücker, “The Microservices Workflow Automation Cheat Sheet,” 2018, Online. Available: https://blog.bernd-ruecker.com/the-microservice-workflow-automation-cheat-sheet-fc0a80dc25aa[retrieved: 04, 2023].
[19] D. Mikalkinas, *Situation-aware Modelling and Execution of Choreography*. Stuttgart University, 2015.
[20] M. Milanović and D. Gasević, “Modeling service choreographies with rule-enhanced business processes,” in *2010 14th IEEE International Enterprise Distributed Object Computing Conference*, 2010, pp. 194–203.
[21] J. Ortiz, V. Torres, and P. Valderas, “A catalogue of adaptation rules to support local changes in microservice compositions implemented as choreographies of bpmn fragments,” 2023, Online. Available:https://riunet.upv.es/bitstream/handle/10251/181551/CatalogueOfAdaptationRules.pdf?sequence=1 [retrieved: 05, 2023].
[22] C. Richardson, “Eventuate Framework,” 2021, Online. Available: https://eventuate.io/[retrieved: 04, 2023].
[23] ——, “Eventuate Tram,” 2021, Online. Available: https://eventuate.io/abouteventuatetram.html[retrieved: 04, 2023].
[24] ——, “Eventuate Local,” 2023, Online. Available: https://github.com/eventuate-local/eventuate-local[retrieved: 04, 2023].
[25] J. Ladleif, A. von Weltzien, and M. Weske, “chor-js: A modeling framework for bpmn 2.0 choreography diagrams,” 2019.
[26] A. Barros, M. Dumas, and H. A.H.M, “Service interaction patterns,” 2005, pp. 302–318, Online. Available:http://www.workflowpatterns.com/documentation/documents/serviceinteraction_BPM05.pdf [retrieved: 05, 2023].
[27] C. Chen, “Choreography vs orchestration,” Online. Available: https://medium.com/ingeniouslysimple/choreography-vs-orchestration-a6f21cfaccae [retrieved: 04, 2023].
[28] B. Rücker, “The Microservices Workflow Automation Cheat Sheet,” Online. Available: https://blog.bernd-ruecker.com/the-microservice-workflow-automation-cheat-sheet-fc0a80dc25aa [retrieved: 04, 2023].
[29] ——, *Practical Process Automation - Orchestration and Integration in Microservices and Cloud Native Architectures*. O’Reilly, 2021.

 ISBN: 978-1-68558-043-8

Migration to Microservices: A Comparative Study of Decomposition Strategies and Analysis Metrics

Meryam Chaieb
Laval University
Quebec, QC, Canada
meryam.chaieb.1@ulaval.ca

Khaled Sellami
Laval University
Quebec, QC, Canada
khaled.sellami.1@ulaval.ca

Mohamed Aymen Saied
Laval University
Quebec, QC, Canada
mohamed-aymen.saied@ift.ulaval.ca

***Abstract*—The microservice architectural style has gained widespread popularity among developers due to its ability to provide numerous benefits, such as scalability, reusability and easy maintainability. However, transforming a monolithic application into a microservices-based architecture can be a complex and an expensive process. To address this challenge, we propose a novel method that leverages clustering to identify potential microservices from a monolithic application. Our approach uses a density-based clustering algorithm that considers the static analysis, structural and semantic relationships between the classes to establish a functionally coherent class partitioning. To evaluate our approach, we analyzed its hyperparameter sensitivity and compared it to two other well known clustering algorithms using various metrics on a Java applications. Our approach showed promising results, demonstrating its effectiveness and stability.**

Keywords-microservices architecture; static analysis; clustering; decomposition.

I. Introduction

The monolithic architectures is one of the most widely utilized architectures for software design. In the realm of software architecture, the monolithic architecture stands as a prominent approach where an application is built as a single, indivisible unit. It encompasses all essential functionalities and components within a unified codebase, thereby presenting a tightly coupled system. This architectural style often involves a centralized database, user interface, and business logic, rendering it self-contained and independent of external services. An exemplar of monolithic architecture, that we will use later in our evaluation process, can be observed in the context of the DayTrader [1] application, a virtual stock trading platform. In this monolithic setup, all trading functionalities, user management, and financial calculations are contained within a single application. While this approach simplifies development and deployment and despite being used since the early days of software systems, it can pose challenges when it comes to scalability, maintaining code integrity, and accommodating changes or updates in individual components [2]–[4].

Many methods have arisen throughout time to solve these issues, such as migrating to new technologies, managing independent services, and deploying more powerful servers. Despite the availability of these solutions, monolithic architectures are still limited by inherent drawbacks such as their large, complex, and often inefficient nature, which may hinder their ability to support advanced and more sophisticated technologies [5]–[10].

Microservices architecture, in the other side, is gaining in popularity and is projected to play a large role in developing scalable, easy to maintain software products by focusing on tightly defined, separated services inside a distributed system. The microservice architecture emerges as a contemporary approach where an application is built as a collection of small, independent services. These services are designed to be modular, self-contained, and focused on specific business functionalities. Unlike the monolithic architecture, microservices operate as autonomous units that communicate with each other through well-defined APIs. This architectural style enables teams to develop, deploy, and scale individual services independently, fostering flexibility and maintainability. A noteworthy example of the microservice architecture can be found in the Netflix streaming platform. In this setup, various microservices handle distinct tasks such as user authentication, content recommendation, billing, and media streaming. Each microservice can be developed, tested, deployed, and scaled independently, allowing Netflix to rapidly innovate, adapt to changing demands, and deliver a seamless streaming experience to its vast user base [11]. The transition from a monolithic design to a more durable and robust microservice architecture is based on the idea of finding contextually and functionally relevant modules and encapsulating them in a single service, while ensuring strong cohesion and low coupling between them. As Rosati pointed out in their research on the migration cost [12], transforming a mature monolithic software into microservices architecture may demand substantial investment in terms of time and cost. These difficulties have prompted academics to devise automatic decomposition methods that might ease the migration process.

The task of transitioning a monolithic application into a microservices architecture is treated as a clustering problem in the context of our project. Our suggested method entails a multi-step procedure that employs static examination of the source code and density-based clustering algorithm to divide the classes into multiple potential microservices that may be evaluated further. We conducted an in-depth review utilizing a variety of metrics to measure the efficacy and efficiency of our method.

 ISBN: 978-1-68558-043-8

The main contributions of our work are as follows:

1) The proposed approach combines density-based clustering and static analysis techniques to leverage the advantages of both methods. It considers the structural and semantic dependencies among classes in a given monolithic application.
2) A comparison between the resulting decomposition of the proposed algorithm and those of commonly used clustering algorithms in the field.

This paper is structured as follows: Section II details the proposed methodology, including the clustering algorithms used. In Section III, we discuss the findings of this effort and respond to different research questions. Section IV presents the related work in the field of monolithic migration to microservices. Section **??** outlines the threats to validity that were considered during the study. Finally, Section V, concludes the work and discuss future research directions.

II. Proposed Approach

The task of extracting microservices from a monolithic software is approached as a clustering problem, with the application's source code as input. Figure 1 outlines the phases involved in our research. Our primary goal in this effort is to achieve granularity at the class level.

Our technique begins with the extraction of semantic and structural information via static analysis of the source code. Then, it evaluates all potential combinations while selecting only one option from each semantic and structural preprocessing component. We feed these representations to each one of the clustering algorithms, resulting in three distinct decompositions that will be analyzed and compared .

A. Representation of the Monolithic Application

The monolithic application is represented as a set of Object Oriented Programming classes denoted as C_M=(c_1,...,c_Z), where Z represents the total number of classes. In this context, our approach aims to partition the original monolithic application into a set of K microservices M=(m_1,..,m_K). Each microservice, m_i=(c_a,...,c_p), represents a subset of the original classes. We aim to optimize the migration process, where each microservice is expected to be cohesive and loosely coupled, resulting in a more maintainable and scalable architecture.
The initial step, presented in the diagram in Figure 1, focuses on representing the monolithic application and extracting the necessary information to build the microservices. To do this, we begin by creating an encoding scheme for the monolith's classes to capture their structural and semantic links.

1) Structural encoding: Abstract Syntax Trees (ASTs) can be created after the source code has been parsed using a static analysis tool, such as "Understand" [9]. These ASTs are used to extract call relationships between classes in a form of an interaction graph. As described in task 2.1 of the diagram presented in Figure 1 the structural information can be encoded using three different options:

- $Call_{in}$, $Call_{out}$: Each class is represented as the sum of incoming and outgoing calls. Our strategy seeks to group classes that interact frequently within the same cluster in order to reduce coupling while promoting greater cohesion within the resulting microservices.
- Call frequencies: This option tries to build more coherent clusters by encoding classes in greater depth. We analyse the call frequencies between each pair of classes to capture a more nuanced understanding of class connections.
- Codependent calls: We consider call frequencies of classes that interacts with both classes to encode each pair of classes. To aid in understanding this concept, we will go through the example in Figure 2 . We have 4 classes: A, B, C, and D. The objective is to encode the pair of classes A and B. Class A is invoked five times by class B, three times by C, and once by D. In addition, class B is invoked twice by C and once by D. The encoding of the pair of classes A and B is the sum of incoming calls to A from the codependent classes C and D.
  The idea behind this technique is that classes that are frequently called together are usually used to handle the same functionality.

2) Semantic encoding: Assume we are dealing with monolithic software projects that were created in accordance with industry norms, the names of classes, methods, and variables are chosen based on functional principles in such projects, and thorough annotations are included to indicate their intended use. By including semantic information into the encoding process, we can determine the essential links between classes and the functionality they provide, facilitating the ability to combine them into coherent microservices.

As a result, the semantic information of each class is composed of a collection of terms that are used in different parts such as comments, method names, and variable names. To preprocess these words, we separate them using CamelCase, filter out stop words and normalise them using stemming.

As seen in task 2.2 in Figure 1, the processed semantic information will be represented in two options:

- Terms frequencies: It involves incorporating the frequencies of terms found within the vocabulary of the application. By doing so, we can ensure that the terms with higher frequencies are more closely related to the domain of the class.
- Term Frequency-Inverse Document Frequency: TF-IDF can improve class clustering in a variety of ways, it considers not only the frequency of a word in a class, but also the inverse document frequency to assess how unique a term is to a class in comparison to the vocabulary. Thus, unique terms that are exclusive to a class will have a larger weight and will be more informative.

B. Clustering algorithms

The objective is to extract microservices by encoding classes structurally and semantically using different combinations of options. To achieve this, we experimented with the Boosted Mean Shift Clustering (BMSC) [13] algorithm, along with other well-known clustering algorithms such as Density-Based

 ISBN: 978-1-68558-043-8

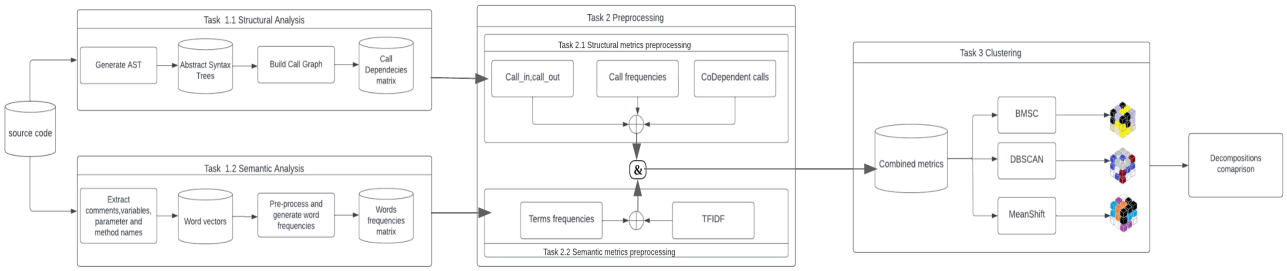

Figure 1: Overview of the Microservices Extraction Process.

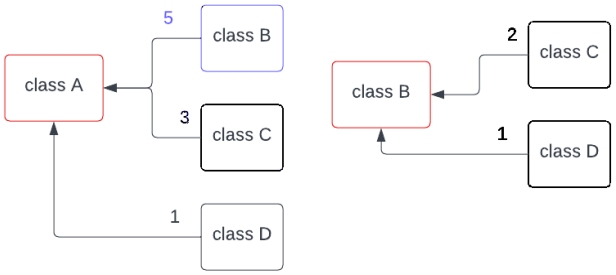

Figure 2: Illustrative example of Codependent calls metric.

Spatial Clustering of Applications with Noise (DBSCAN) [14] and Mean Shift [15].

1) DBSCAN algorithm: DBSCAN is a clustering technique to detect clusters and noise. The user must specify two hyperparameters, Eps and MinPts. The method uses these parameters to arrange densely related points into a single cluster. One major benefit of DBSCAN is that based on the data and the provided hyperparameters, the number of clusters can be arbitrary detected, leading in more accurate clusters [16].

Hyperparameters :

- **Eps** (ϵ) **:** Refers to the radius of the neighbourhood surrounding the cluster's central point.
- **MinPts :** This is the bare minimum of points required to build a cluster.

To build clusters, DBSCAN begins by picking an arbitrary Core point, then it collects data points within a distance equal to Eps. A cluster is produced if the total number of points acquired is more than or equal to MinPts. To enlarge the original cluster, this procedure is repeated for each cluster point. During this step, the algorithm creates the first cluster. The procedure is then repeated after removing all of the points that composed it from the database. When no further clusters can be produced with the provided hyparameters, the algorithm stops. The rest of the points are labelled as Noise.

However, this algorithm is highly sensitive to its hyper-parameters, leading to significant variation in microservices' quality. Moreover, DBSCAN-based approaches may not work well with datasets with varying densities or non-globular shapes.

2) Mean Shift: The Mean Shift method does not need any assumptions about the underlying distribution of the data. It can automatically detect non-linearly formed clusters and compute the number of clusters [15].

It begins by arbitrary identifying a region of interest and calculate its center of density. The mean shift vector is then generated and the center of the area is shifted along the vector until it corresponds with the centre of mass.

Although the Mean Shift method has shown excellent results, the research in [13] demonstrates that BMSC outperformed Mean Shift in a similar clustering problem with more stable clustering.

Given the difficulties involved with clustering algorithms specially when there is no obvious separation between clusters, we decided to investigate alternatives to standard techniques. We picked the BMSC technique since it has showed higher performance in similar clustering tasks.

3) BMSC algorithm: BMSC algorithm is a hybrid clustering technique that combines Mean Shift and DBSCAN. It is a density-based clustering method that overcomes some of the limitations of both approaches and can find clusters of any form and size with varied densities without the need for a predetermined number of clusters [13].

The BMSC first applies the Mean Shift algorithm to generate a set of initial centers that will be the input to the DBSCAN algorithm. BMSC selects a sample of the data that captures the skeleton of the clusters in order to properly identify the data's underlying structure.

Algorithm 1 outlines the steps involved in applying the BMSC algorithm. The first step is to divide the data uniformly into cells of a grid. Then, the Mean Shift algorithm is applied independently to the data in each cell. This produces a list of intermediate mode points (iModes). Next, it disperse the data of the cells using a specific mechanism that involves each grid cell interacting with a limited number of cells in its neighborhood. The BMSC paper [13] presents various

 ISBN: 978-1-68558-043-8

Algorithm 1 Boosted Mean Shift Clustering

Require: X, width, height, Eps.
Ensure: the final clustering results cl_final.
Initialize Grid(X,width,height)
▷ Distribute X over G = width × height cells.
iModes ← ∅
counter ← 1
while $counter! = 3$ **do**
  for j ← 1 **to G do**
    newiModes ← MeanShift($cellData_i$)
    iModes.Append(newiModes)
▷ collect the iModes of each cell of the Grid
  end for
  ConfidenceAssignement(Semantic_similarity)
▷ Assign confidence values to classes in each cell

  for j ← 1 **to G do**
    CollectedData ← CollectNeighborhoodData(j, neighborhood_structure) ∪ $cellData_j$
    $cellData_j$ ← WeightedSampling(CollectedData)
▷ update $cellData_j$
  end for
  cl_iModes, numberOfClusters ← DBSCAN (iModes_similarity, Eps)
▷ cl_iModes is the clustering results of the iModes
  if numberOfClusters == lastnumberOfClusters **then**
    counter++
  else
    counter ← 1
  end if
end while
cl_final ← DataAssignement(X,cl_iModes)

neighborhood structures, which are depicted in Figure 3. In our work, we adopt the linear (5) neighborhood structure.

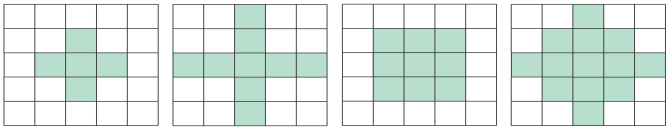

Figure 3: Potential neighbourhood structures.

The subsequent phase involves calculating the distances between all data points in the parent cell and those in its neighboring cells, relative to the iModes using a semantic similarity metric that assesses the confidence level of each relationship. The second stage of the BMSC algorithm utilizes the list of iModes to run DBSCAN. The latter is applied to identify clusters of densely packed iModes, which in turn generates clusters of the original data points.

In our particular scenario, we utilize an aggregation function to transform the iModes into a format similar to that of the legacy application's classes. We represent each group center by summing the structural encodings of its classes, thus capturing the structural aspect of the mode point. Additionally, we compute the semantic part of the vector by summing the term frequencies of words used in those specific classes.

For the purpose of extracting reliable microservices, we adopt a novel approach inspired from the work of Sellami and al [17] where instead of directly inputting the encoders of iModes into the DBSCAN algorithm, we provide the connections or links between each pair of iModes. To achieve this, we employ the iModes similarity measure that capture the structural and semantic relationships. This approach aims to produce microservices that are consistent from implementation and use cases perspectives. The similarity is calculated as follows :

- **iModes Similarity (MS) :** The weighted sum of two similarity metrics, as provided by equation 1.

$$MS(m_i, m_j) = \alpha Sim_{str}(m_i, m_j) + \beta Sim_{sem}(m_i, m_j) \quad (1)$$

With : $\alpha, \beta \in$ *[0,1],* α + β = *1.*
Each one of the similarities is computed as follow:

- **Structural similarity** (Sim_{str}) : This allows us to evaluate their similarity from a functional perspective. It's computed using equation 2

$$sim_{str}(m_i, m_j) = \begin{cases} \frac{1}{2}(\frac{call(m_i,m_j)}{call_{in}(m_j)} + \frac{call(m_i,m_j)}{call_{in}(m_i)}) & If call_{in}(m_i) \neq 0 and call_{in}(m_j) \neq 0 \\ \frac{call(m_i,m_j)}{call_{in}(m_j)} & If call_{in}(m_i) = 0 and call_{in}(m_j) \neq 0 \\ \frac{call(m_i,m_j)}{call_{in}(m_i)} & If call_{in}(m_i) \neq 0 and call_{in}(m_j) = 0 \end{cases} \quad (2)$$

With:

- call(m_i , m_j): The number of calls of m_j by m_i,
- call$_{in}$(m_i): The number of incoming calls in m_i.
- **Semantic Similarity** (Sim_{sem}) **:** Is represented by the cosine similarity between their respective vectors. This is useful for measuring the similarity between different iModes and identifying possible relationships at the domain level [18].

Finally, we employ DBSCAN algorithm on the iModes similarity metric. The process is iterated until production of the same number of clusters for three consecutive iterations.

III. Evaluation

To help better understand the evaluation, Table I summarizes the characteristics of the monolithic application used to evaluate different aspects of our approach.

TABLE I: CHARACTERISTICS OF MONOLITHIC APPLICATION

Project	Version	SLOC	# of classes
DayTrader	1.4	18,224	118

A. Research Questions

The goal of our experimental investigation is to address these research questions (RQs):
RQ1: What is the most effective and promising option among the various choices in our approach?
RQ2: How does the stability and robustness of BMSC algorithms compare to that of Mean Shift and DBSCAN?

B. Evaluation metrics

We used a set of metrics specified in [19] to analyse various aspects of the extracted microservices without relying on the ground truth microservices:

- **Structural Modularity (SM):** Determined by measuring the structural cohesiveness of classes inside a partition and the coupling between partitions.
  The higher SM value, the better the decomposition.

- **ICP:** Depicts the percentage of calls that occur between two divisions.
  The lower the ICP value, the better the recommendation.
- **Interface Number (IFN):** It counts the number of interfaces present in a microservice. An interface is defined as a class within a microservice that is invoked by a class within another microservice.
  The lower the IFN value, the better the recommendation.
- **Non-Extreme Distribution (NED):** It assesses the distribution of classes within microservices and aims to ensure that a microservice is non-extreme. According to [19], a microservice is considered non-extreme if it contains a number of classes within the range of [5, 20]. The lower the NED value, the better the recommendation.

C. Evaluation and Results for RQ1

1) Evaluation protocol: The objective is to compare the quality of the findings from different possible combinations of structural and semantic information in order to identify the most effective strategy for each algorithm using DayTrader application. We assigne abbreviations as follows:

- Option 1 : Call$_{in}$,$Call_{out}$ + Terms frequencies.
- Option 2 : Call$_{in}$,$Call_{out}$ + TFIDF
- Option 3 : Call Frequencies + Terms frequencies.
- Option 4 : Call Frequencies + TFIDF.
- Option 5 : Codependant calls + Terms frequencies.
- Option 6 : Codependant calls + TFIDF.

Hyperparameters were fixed according to the literature:

- Bandwidth :Is set using the estimate bandwidth function from scikit-learn, which estimates the value of the bandwidth based on the provided data.
- MinPts : Is set to 5 because a cluster is considered not extreme if its size ranges from 5 to 20 [19] for DBSCAN alone and set to its default value (MinPts = 1) for BMSC algorithm [13].
- Eps : Is set using a k-distance graph technique.

2) Results: According to the results presented in Table II, option 6 is deemed the most suitable for the Mean Shift algorithm when considering the SM metric and shows a performance that is comparable to the best results obtained when evaluating other metrics. For DBSCAN, presented in Table III, option 6 has shown better results in terms of IFN, ICP, and NED, with a SM value that is close to the maximum. Both DBSCAN and Mean Shift algorithms presented varied results, while BMSC had very similar results for all options and all metrics. Furthermore, BMSC was able to detect a more stable number of microservices compared to the other algorithms, which often formed one large cluster or unique classes that did not meet the research goals. In contrast, the resultant microservices from BMSC were balanced and stable across different approaches, with the largest microservice containing a maximum of 17 classes as presented in Table IV.

TABLE II: EVALUATION RESULTS OF DAYTRADER APPLICATION USING MEAN SHIFT ALGORITHM

Metrics	Option 1	Option 2	Option 3	Option 4	Option 5	Option 6
SM	0.8526	0.7853	0.7944	0.8614	0.8575	0.8742
IFN	1.235	1.8	1.277	1.0454	1.0	1.214
ICP	1.0	0.9	1.0	1.0	1.0	1.0
NED	1.0	0.9	1.0	1.0	1.0	1.0
# microservices	17	10	18	22	21	14
size of the largest micro	98	102	97	92	97	104

TABLE III: EVALUATION RESULTS OF DAYTRADER APPLICATION USING DBSCAN ALGORITHM

Metrics	Option 1	Option 2	Option 3	Option 4	Option 5	Option 6
SM	0.120	0.1085	0.2702	0.2718	0.2487	0.1116
IFN	0.120	0.1085	0.2702	0.2718	0.2487	0.1116
ICP	0.3244	0.1426	0.1591	0.2859	0.3482	0.0079
NED	0.5	0.666	1.0	0.5	0.5	0.333
# microservices	2	3	2	2	2	3
size of the largest micro	108	86	116	113	113	104

Option 6 is the optimal approach for all three clustering algorithms. It uses co-dependent calls metric as structural information and TF-IDF vector as semantic information. Consequently, our work will continue to focus on this strategy.

D. Evaluation and Results for RQ2

1) Evaluation protocol: The purpose is to examine the sensitivity to the hyperparameters of BMSC algorithm compared to that of DBSCAN and Mean Shift individually. For each hyperparameter, we firstly specified the range of potential values. The other hyperparameters were then fixed, and the algorithm was performed for each possible value, recording the extracted microservices. The outcomes were then reviewed using multiple metrics, and the metric values were plotted at each step. We focused on the DayTrader monolithic project for our investigation since it is a well-established benchmark for this topic.

- **Bandwidth :** using the estimate bandwidth function from the scikit-learn we estimate the maximum value of the kernel bandwidth, and then we varied the values of the hyperparameter from 0 to this estimated value.
- **Eps:** We varied the Epsilon values from 0 to 1 with a step equal to 0.05.

2) Results: Figure 4 showcases an evaluation of five different techniques, represented as subfigures. The Y-axes in each subfigure indicate the metric scale, while the X-axis displays the boxplot results for each technique arranged in the following order:

1) BMSC_eps: BMSC results varying its epsilon hyperparameter.
2) DBSCAN_eps: DBSCAN results varying its epsilon hyperparameter.
3) BMSC_band: BMSC results varying its bandwidth hyperparameter.
4) MeanShift_band: Mean shift results varying its bandwidth hyperparameter.

Each subfigure in Figure 4 is dedicated to a specific evaluation metric, allowing for a thorough comparative analysis of various aspects of the techniques to derive insights regarding

TABLE IV: EVALUATION RESULTS OF DAYTRADER APPLICATION USING BMSC ALGORITHM

Metrics	Option 1	Option 2	Option 3	Option 4	Option 5	Option 6
SM	0.3696	0.3435	0.3887	0.4697	0.40545	0.4052
IFN	1.0344	1.250	1.0370	0.9677	1.0769	1.318
ICP	0.6500	0.591	0.618	0.6432	0.6257	0.639
NED	0.7241	0.6666	0.7037	0.7419	0.6538	0.636
# microservices	29	24	27	31	26	22
size of the largest micro	13	13	13	13	16	17

their stability. The subfigures are arranged in the following order: SM, IFN, ICP, and NED, with each focusing on the evaluation of the corresponding metric for each technique. The last subfigure provides the results for the number of generated microservices per technique, represented as "# microservices". By carefully examining these evaluation metrics, we can gain a comprehensive understanding of the performance and stability of the techniques.

Upon analyzing Figure 4, it becomes clear that BMSC exhibits greater sensitivity compared to DBSCAN in relation to the epsilon hyperparameter (BMSC_eps vs. DBSCAN_eps), as well as greater sensitivity compared to Mean Shift in relation to the bandwidth hyperparameter (BMSC_band vs. Mean shift_band) across all evaluated metrics.

In the analysis of Figure 4, several noteworthy observations can be made. Firstly, despite DBSCAN outperforming BMSC in terms of the structural modularity (SM) and interface number (IFN) metrics, it results in a significantly high number of microservices. With an average of 115 microservices for an application containing only 118 classes, this outcome does not align with our migration goals. This discrepancy suggests that DBSCAN may be suffering from the "boulders and grains" problem, generating microservices that are either too small or too large. Such an outcome fails to address the limitations of the monolithic application and does not contribute to the desired loosely coupled microservices architecture.

On the other hand, Mean Shift exhibits better performance in terms of the number of generated microservices. Its mean number of microservices is comparable to that of BMSC, indicating a more balanced decomposition approach with fewer than 20 microservices on average. This suggests that Mean Shift provides a more suitable solution for achieving the desired granularity in the migration process of monolithic applications.

Furthermore, the analysis reveals that BMSC demonstrates greater sensitivity when varying the epsilon hyperparameter compared to the bandwidth hyperparameter (BMSC_band vs. BMSC_eps), as evident from the boxplots in the final subfigure. This sensitivity is also apparent in the boxplot variation of the non-extreme distribution (NED) subfigure, where the variation in epsilon results in up to a 60% change. This discrepancy can be attributed to the fact that varying the bandwidth can generate different modes that are connected using DBSCAN, whereas varying the epsilon hyperparameter directly affects the final number of microservices, as indicated by the comparison of variations in the number of microservices.

In contrast to the findings in [13], our analysis suggests that for our case, BMSC is more susceptible to the selection of its hyperparameters, specifically the epsilon parameter, compared to DBSCAN and Mean Shift when used independently. However, BMSC demonstrates greater consistency in the resulting decompositions across hyperparameter variations.

IV. Related Work

The first component of a decomposition approach is concerned with the type of input and how it is handled. The methods suggested by MSExtractor [20], Bunch [21], and [22] , for example, take as input the source code of a monolithic system and apply various static analysis techniques to it. The approach called HierDecomp [17], employes in addition the semantic similarity generated from the code text analysis. Other approaches, such as Mono2Micro [19], FoSCI [23], and COGCN [24], are based on the study of monolithic system use cases and execution traces. Sellami and al [25] combine both static and dynamic analysis in order to cover the individual disadvantages of each of the analysis approaches. There are, on the otehr hand, systems that employ different inputs, such as MEM [26], which analyses the git commit history of monolithic programs.

Most methods utilize clustering algorithms, such as [22] which feeds vectors derived from code embedding into an Affinity propagation clustering process [27]. The similarity metrics computed by an agglomerative single-linkage clustering method [28] are used by Mono2Micro [19]. Based on the graph it developed, MEM [26] provides its own clustering mechanism. Based on the similarity metrics, HierDecomp [17] and HyDecomp [25] employ a DBSCAN [16] density based clustering algorithm which ends by having a hierarchical microservices decomposition recommendation. Some methods suggest search algorithms to accomplish their goal. MSExtractor [29] uses the non-dominated sorting genetic algorithm (NSGA-II) [30] whereas FoSCI [23] employs both NSGA-II and hierarchical clustering. A community discovery method is used by Service Cutter to provide a decomposition.

However, many existing approaches encounter the challenge known as the "boulders and grains" problem, which arises when microservice decompositions lean towards being excessively large or overly small. Both situations introduce drawbacks in terms of system architecture and management.
When a microservice decomposition becomes too large, it can lead to heightened complexity and diminished modularity. Large microservices that encompass numerous classes or functionalities become cumbersome to maintain, understand, and update. Furthermore, even minor changes to a component within a large microservice may necessitate redeploying the entire service, impeding agility and scalability.
Conversely, when a microservice decomposition is excessively small, it can result in an abundance of services and unnecessary network communication overhead. Microservices consisting of only a few classes can lead to an excessively fragmented architecture, resulting in increased latencies and

 ISBN: 978-1-68558-043-8

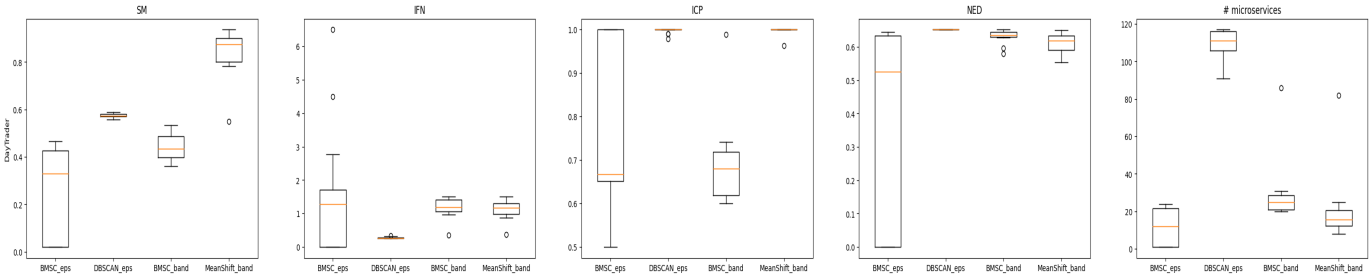

Figure 4: Evaluation metrics for different hyperparameters values when extracting microservices from the project DayTrader.

added complexity in managing the interactions among numerous small services.

Assessing whether a microservice decomposition is too large or too small requires careful evaluation. Qualitative factors such as complexity, cohesion, and adherence to the Single Responsibility Principle offer valuable insights into the size of a microservice. Additionally, quantitative metrics can be employed to measure microservice size, such as counting the number of classes or lines of code it encompasses. For example, "Mono2micro" paper [19] suggests a guideline for microservice size, recommending that an optimal microservice consists of 5 to 20 classes. This quantitative threshold aims to strike a balance, ensuring that microservices remain manageable and cohesive without succumbing to excessive granularity or complexity.

V. Conclusion and Future Work

In conclusion, this paper has presented a comparative study of different strategies for decomposing monolithic applications into microservices. Our proposed approach, utilizing the BMSC algorithm, effectively groups semantically and structurally similar classes to extract potential microservices. Notably, our approach demonstrates promising results by solely utilizing select characteristics of the monolithic application's source code as input, distinguishing it from approaches that require additional data sources.

Through extensive evaluation using various performance metrics, we have compared our approach with two well-established algorithms in the field. The experimental results highlight the superior cohesion within microservices, reduced interactions between microservices, and overall improved stability achieved by our method. However, it should be noted that the sensitivity to the epsilon hyperparameter remains a limitation, posing challenges in its selection.

Looking ahead, our future work will focus on developing more refined metrics to evaluate the extracted microservices and conducting comparative analyses against existing decomposition techniques. We also aim to explore different similarity metrics and investigate alternative types of interactions between classes beyond direct method calls. To further enhance the granularity of our approach, we intend to extend it to consider methods or functions of the monolithic application as a basis for decomposition, going beyond class-level granularity. Additionally, we recognize that static analysis alone may not provide a comprehensive understanding of application functionalities and interactions during runtime. Therefore, we propose exploring hybrid solutions that incorporate dynamic analysis of the source code to enrich the decomposition process.

By addressing these avenues for future research, we aim to advance the field of microservice decomposition and contribute to the development of effective and scalable approaches for migrating monolithic applications to microservices architectures.

References

[1] "sample.daytrader7," 2023-04-24. [retrieved: May, 2023].

[2] F. Tapia, M. . Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, "From monolithic systems to microservices: A comparative study of performance," *Applied Sciences*, vol. 10, 2020.

[3] O. Benomar, H. Abdeen, H. Sahraoui, P. Poulin, and M. A. Saied, "Detection of software evolution phases based on development activities," in *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 15–24, IEEE, 2015.

[4] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "A kubernetes controller for managing the availability of elastic microservice based stateful applications," *Journal of Systems and Software*, vol. 175, p. 110924, 2021.

[5] V. Velepucha and P. Flores, "Monoliths to microservices - migration problems and challenges: A sms," in *2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)*, pp. 135–142, 2021.

[6] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," in *2018 IEEE 11th international conference on cloud computing (CLOUD)*, pp. 970–973, IEEE, 2018.

[7] M. A. Saied, H. Sahraoui, E. Batot, M. Famelis, and P.-O. Talbot, "Towards the automated recovery of complex temporal api-usage patterns," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1435–1442, 2018.

[8] S. Huppe, M. A. Saied, and H. Sahraoui, "Mining complex temporal api usage patterns: an evolutionary approach," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pp. 274–276, IEEE, 2017.

[9] M. A. Saied, O. Benomar, and H. Sahraoui, "Visualization based api usage patterns refining," in *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, pp. 155–159, IEEE, 2015.

[10] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Microservice based architecture: Towards high-availability for stateful applications with kubernetes," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 176–185, IEEE, 2019.

[11] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022.

 ISBN: 978-1-68558-043-8

[12] P. Rosati, F. Fowley, C. Pahl, D. Taibi, and T. Lynn, "Right scaling for right pricing: A case study on total cost of ownership measurement for cloud migration," vol. 1073, pp. 190–214, Springer Verlag, 2019.

[13] Y. Ren, U. Kamath, C. Domeniconi, and G. Zhang, "Boosted mean shift clustering," in *Machine Learning and Knowledge Discovery in Databases* (T. Calders, F. Esposito, E. Hllermeier, and R. Meo, eds.), vol. 8725, pp. 646–661, Springer Berlin Heidelberg, 2014.

[14] H. V. Singh, A. Girdhar, and S. Dahiya, "A literature survey based on DBSCAN algorithms," in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 751–758, 2019.

[15] K. G. Derpanis, "Mean shift clustering."

[16] D. Deng, "Dbscan clustering algorithm based on density," *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pp. 949–953, 2020.

[17] K. Sellami, M. A. Saied, and A. Ouni, "A hierarchical dbscan method for extracting microservices from monolithic applications," in *The International Conference on Evaluation and Assessment in Software Engineering 2022*, pp. 201–210, 2022.

[18] A. Mishra and S. K. Vishwakarma, "Analysis of tf-idf model and its variant for document retrieval," *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 772–776, 2015.

[19] A. K. Kalia, X. Jin, K. Rahul, S. Saurabh, V. Maja, and B. Debasish, "Mono2micro: A practical and effective tool for decomposing monolithic java applications to microservices," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021.

[20] K. Sellami, A. Ouni, M. A. Saied, S. Bouktif, and M. W. Mkaouer, "Improving microservices extraction using evolutionary search," *Information and Software Technology*, vol. 151, p. 106996, 2022.

[21] B. S. Mitchell and S. Mancoridis, "On the evaluation of the bunch search-based software modularization algorithm," *Soft Computing*, vol. 12, no. 1, pp. 77–93, 2008-01-01.

[22] O. Al-Debagy and P. Martinek, "A microservice decomposition method through using distributed representation of source code," *Scalable Computing*, vol. 22, pp. 39–52, 2021.

[23] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, and Q. Zheng, "Service candidate identification from monolithic systems based on execution traces," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 987–1007, 2021.

[24] U. Desai, S. Bandyopadhyay, and S. Tamilselvam, "Graph neural network to dilute outliers for refactoring monolith application."

[25] K. Sellami, M. A. Saied, A. Ouni, and R. Abdalkareem, "Combining static and dynamic analysis to decompose monolithic application into microservices," in *Service-Oriented Computing* (J. Troya, B. Medjahed, M. Piattini, L. Yao, P. Fernández, and A. Ruiz-Cortés, eds.), (Cham), pp. 203–218, Springer Nature Switzerland, 2022.

[26] G. Mazlami, J. Cito, and P. Leitner, "Extraction of microservices from monolithic software architectures," in *2017 IEEE International Conference on Web Services (ICWS)*, pp. 524–531, 2017.

[27] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007-02-16.

[28] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method," *Comput. J.*, vol. 16, pp. 30–34, 1973.

[29] I. Saidani, A. Ouni, M. W. Mkaouer, and A. Saied, "Towards automated microservices extraction using muti-objective evolutionary search," in *International Conference on Service-Oriented Computing*, pp. 58–63, Springer, 2019.

[30] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, 2002.

 ISBN: 978-1-68558-043-8

A Review on Digital Wallets and Federated Service for Future of Cloud Services Identity Management

Fatemeh Stodt and Christoph Reich
Institute for Data Science, Cloud Computing, and IT Security; Furtwangen University,
Robert-Gerwig-Platz 1, 78120 Furtwangen, Germany
{Fatemeh.Stodt, Christoph.Reich}@hs-furtwangen.de

***Abstract*—In today's technology-driven era, managing digital identities has become a critical concern due to the widespread use of online services and digital devices. This has led to a fragmented landscape of digital identities, burdening individuals with multiple usernames, passwords, and authentication methods. To address this challenge, digital wallets have emerged as a promising solution. These wallets empower users to store, manage, and utilize their digital assets, including personal data, payment information, and credentials. Additionally, federated services have gained prominence, enabling users to access multiple services using a single digital identity. Gaia-X is an example of such a service, aiming to establish a secure and trustworthy data infrastructure. This paper examines digital identity management, focusing on the application of digital wallets and federated services. It explores the categorization of identities needed for different cloud services, considering their unique requirements and characteristics. Furthermore, it discusses the future requirements for digital wallets and federated identity management in the cloud, along with the associated challenges and benefits. The paper also introduces a categorization scheme for cloud services based on security and privacy requirements, demonstrating how different identity types can be mapped to each category.**

***Index Terms*—Digital wallet, Identity management, Federated service, Cloud**

I. Introduction

The management of digital identities has become a critical concern in today's digital age [1]. With the increase of online services and the widespread use of digital devices, individuals are constantly required to provide personal information to access different platforms, services, and applications [2]. This has led to a fragmented landscape of digital identities, where users have to manage multiple usernames, passwords, and authentication methods, which can be both cumbersome and insecure [3].

Digital wallets have emerged as a promising solution to tackle the challenge of managing digital identities [4]. These software applications enable users to conveniently store, manage, and utilize various digital assets, such as personal data, payment information, and credentials.

In addition, Gaia-X, a groundbreaking project [5], exemplifies the importance of federated services and the significant benefits they offer. Gaia-X is designed to provide users with a robust and secure data infrastructure, empowering them with unprecedented control over their personal information [6]. By adopting Gaia-X's unified digital wallet, users can conveniently access multiple platforms and services with a single digital identity, streamlining the management of their digital presence while bolstering security and privacy. This usercentric approach not only enhances individual control but also promotes innovation and competition within the digital landscape, reinforcing the advantages of Gaia-X's federated service model.

In the contemporary digital landscape, the significance of effective digital identity management cannot be emphasised enough. Fortunately, emerging solutions such as digital wallets and federated services provide promising avenues to address this intricate challenge. This paper aims to delve into the concept of digital identity management and shed light on its applications, specifically within the realm of digital wallets and federated services. Additionally, we will explore the utilisation of digital wallets for accessing cloud services, offering insights into their benefits and potential challenges.

The structure of this paper is as follows: Section II provides background information on digital wallets and federated services. Section III discusses the requirements for identity management in wallets to access the cloud. Section IV categorises cloud access based on identity group levels. Finally, in Section V, we draw a conclusion.

II. Background (State of the Art)

This section provides an overview of two key components: Digital Wallets and Federated Services, which play pivotal roles in ensuring secure and efficient digital experiences.

A. Digital Wallet

The digitisation of transactions has accelerated, particularly in response to the pandemic, resulting in an increased reliance on electronic services. Users now engage in various activities, such as tax declarations, accessing vaccination and test certificates, and interacting with public administrations, through digital platforms [7]. To access these services, users must authenticate themselves and provide electronic identification (eID) to secure personalised services and data. This authentication process is facilitated by identity management (IdM) systems, which ensure reliable and secure user authentication.

Digital wallets have emerged as a crucial component in managing identities in the digital identity domain. A digital wallet is a secure and encrypted storage solution that allows users to store and manage their digital identities, credentials,

 ISBN: 978-1-68558-043-8

and other relevant information [8]. It acts as a central repository where users can securely store their authentication data, such as usernames, passwords, and digital certificates [9].

Digital wallets offer several advantages in identity management [10]. They provide convenience by allowing users to have a single repository for all their identities across different services and platforms. Users can store and manage multiple sets of credentials within their digital wallet, eliminating the need to remember separate usernames and passwords for each service provider. This simplifies the user experience and reduces the cognitive burden of managing multiple identities [11].

Various models of identity management systems have emerged over the years. The isolated model, where each service provider has its own identity provider (IdP), was the earliest and most prevalent [12]. However, this model requires users to register separately with each service provider, resulting in the burden of managing multiple credentials. To address this, the central identity model was introduced, outsourcing the IdP functionality to a central entity that multiple service providers can utilise [13]. Users only need to register once with the central IdP and can then access various services with the same set of credentials.

While the central identity model improves usability, it raises concerns about the central IdP becoming a single point of failure and potential privacy breaches. To overcome these challenges, the federated IdM model was introduced, establishing trust relationships among multiple IdPs [14]. This model allows users registered with one IdP to authenticate themselves to service providers served by other IdPs within a circle of trust. An example is the European eIDAS interoperability framework [15], which enables cross-border authentication processes by federating national IdM systems of EU Member States.

Another approach is the user-centric IdM model, where identity data is stored in the user's domain, such as on a smartcard or a smartphone with a hardware-based security element [16]. Users retain control over their identity data, enhancing privacy. National IdM solutions utilising smartcards, such as the Austrian Citizen Card and the German eID, exemplify this model. During authentication, the necessary identity information is retrieved from the user's domain and forwarded to the requesting service provider.

Recent advancements include the concept of Self-Sovereign Identity (SSI), where users have sole control over their credentials [17] [10]. SSI reduces reliance on central authorities by utilising distributed ledgers among multiple IdPs within a circle of trust for registering new credentials. Initiatives like the European Self-Sovereign Identity Framework (ESSIF) [18] and Veramo [19] embody this model. These developments reflect a trend towards user-controlled identity data and have attracted attention from policymakers, as evident in the European Commission's proposal for a new European Digital Identity.

The OpenWallet Foundation (OWF) has emerged as a new opportunity in the realm of digital wallets [20]. Established under the umbrella of the Linux Foundation Europe, OWF aims to develop open-source software that facilitates interoperability across a broad spectrum of wallet applications [21]. These applications encompass various use cases, including payments, identity verification, and the secure storage of validated credentials.

B. Federated Services

The concept of a federated catalogue plays a vital role in identity management by facilitating the discovery and access to various services through a centralised repository [6]. In a federated catalogue model, multiple catalogues collaborate and share information about available services, creating a unified and comprehensive resource for users [22]. This collaborative approach allows users to search, browse, and access services from different providers using a single interface, streamlining the process of service discovery.

Inter-catalogue synchronisation is a critical aspect of federated catalogues. It ensures that information about services, including their availability, descriptions, and attributes, remains up-to-date and consistent across different catalogues. Through inter-catalogue synchronisation mechanisms, updates and changes made in one catalogue can be propagated and reflected in others, maintaining data integrity and ensuring accurate and real-time information for users. This synchronisation process enables a seamless user experience, where users can rely on the federated catalogue to provide reliable and consistent information about services.

The integration of wallets with federated catalogues introduces an additional layer of functionality and convenience to identity management [23]. Wallets, which store and manage users' digital identities and associated credentials, can interact with federated catalogues to enhance the service discovery and access process. When a user accesses the federated catalogue through their wallet, the wallet can authenticate the user and provide relevant identity information to the catalogue. This interaction enables personalised service recommendations, tailored search results, and seamless authentication and authorisation processes, ultimately enhancing the user experience and security.

Federated services and federated catalogues are closely intertwined concepts in identity management. Federated services rely on federated catalogues to provide a centralised and comprehensive view of available services, allowing users to discover and access services using a single digital identity. The collaboration between service providers and catalogues within a federated model streamlines identity management processes, as the catalogue acts as a trusted intermediary, enabling authentication, authorisation, and seamless information exchange between users and service providers [24].

III. Requirements for Identity Wallets for Future Clouds

As cloud computing continues to shape the digital landscape, effective identity management becomes paramount to ensure secure and seamless access to cloud services. In this

 ISBN: 978-1-68558-043-8

section, we delve into the categorisation of identities required for different cloud services, discuss their unique requirements and characteristics, explore the future requirements for digital wallets and federated identity management in the cloud, identify potential challenges in implementing identity wallets for future clouds, and highlight the potential benefits of using digital wallets for identity management in the cloud.

Categorising identities according to their usage in different cloud services provides a comprehensive understanding of the diverse identity landscape. These identities can be broadly classified into user identities, service identities, and device identities. User identities represent individuals accessing cloud services, service identities are associated with specific cloud services or applications, and device identities pertain to the authentication and authorisation of devices interacting with cloud resources. Each identity category has distinct requirements and characteristics that must be considered to ensure effective identity management.

R1: Secure storage of identity-related data: In the context of cloud services, it is crucial to securely store identity and identity-related information. This requirement ensures that sensitive data associated with user identities, service identities, and device identities is stored in a protected manner, safeguarding the integrity and confidentiality of the data.

R2: Effective management of identity-related data: Managing identity-related data in the cloud encompasses various functionalities. These include the ability to select, remove, and review identity data stored within the cloud environment, as well as the capability to choose which identity data should be shared outside the cloud. Such management ensures that users have control over their stored information, promoting privacy and data control.

R3: Secure sharing of identity-related data: Enabling the secure sharing of stored identity-related data outside the cloud is a critical requirement for cloud-based identity management. This involves establishing secure communication channels and protocols for sharing identity data with trusted entities, ensuring that data integrity and confidentiality are maintained during the sharing process.

R4: Secure storage of cryptographic material: As the cloud environment handles digital identities, it becomes essential to securely store cryptographic material related to digital identity. This requirement focuses on the need to protect cryptographic elements, such as keys and certificates, ensuring their confidentiality and preventing unauthorised access.

R5: Combining identity data before sharing: In the cloud context, the requirement to combine identity data before sharing aligns with the concept of selective disclosure. Users should have the ability to selectively share identity data, combining relevant information based on specific sharing requirements. This ensures privacy and controlled sharing of identity-related data.

The use of digital wallets for identity management in the cloud offers a range of benefits. Digital wallets enhance user convenience by providing a centralized platform for managing identities across multiple cloud services. They bolster security through secure authentication mechanisms, robust encryption of identity data, and efficient access control. Moreover, digital wallets empower users by giving them control over their personal information and the ability to selectively share it with trusted entities. The integration of digital wallets with federated identity management further streamlines identity management processes, enabling seamless access to cloud resources and promoting interoperability.

TABLE I
COMPARISON BETWEEN DIFFERENT DIGITAL WALLET BASED ON IDM AND WALLET REQUIREMENTS

Reference	IdM	Environment	R1	R2	R3	R4	R5
[17]	SSI	Local	✓	✓	✓	✓	✓
[23]	Federated	Local	✓	✓	✓	✓	✓
[9]	Centralized	Local	✓	✓	✓	✓	✗
[15]	as a Service	Remote	✓	✓	✓	✗	✓
[16]	User-centric	Local	✓	✓	✓	✓	✗
[13]	Centralized	Remote or Local	✓	✓	✓	✓	✗
[10]	SSI	Remote or Local	✓	✓	✓	✓	✓
[20], [21]	SSI	Remote or Local	✓	✓	✓	✓	✓

In this context, Table I provides a comprehensive comparison between different digital wallets based on their identity management (IdM) capabilities and wallet requirements. This table serves as a valuable reference for understanding the strengths and features of various digital wallet solutions in relation to identity management. It highlights key factors such as authentication mechanisms, encryption techniques, access control capabilities, and user control features. By referring to Table 1, readers can gain insights into the specific characteristics and functionalities of each digital wallet, aiding in the selection of an appropriate solution for their identity management needs.

IV. ACCESS MANAGEMENT AND CATEGORISING IDENTITIES FOR CLOUD SERVICES

In the realm of cloud computing, it is crucial to consider the security and privacy requirements of different types of cloud services. In this section, we introduce a categorisation scheme that classifies cloud services based on their security and privacy requirements. We then explore how different types of identities can be mapped to each category, considering their respective security and privacy features. Furthermore, we provide practical examples to illustrate how this categorisation scheme can guide the selection of the appropriate identity type for a given cloud service.

The categorisation scheme for cloud services is designed to capture the varying degrees of security and privacy requirements across different service types. We propose a three-tier categorisation: low-security services, moderate-security services, and high-security services as shown in Figure 1. Low-security services typically involve non-sensitive data and require minimal protection measures. Examples include publicly accessible websites or public information repositories. Moderate-security services handle moderately sensitive data, such as personal information or internal organisational documents. High-security services, on the other hand, handle highly

 ISBN: 978-1-68558-043-8

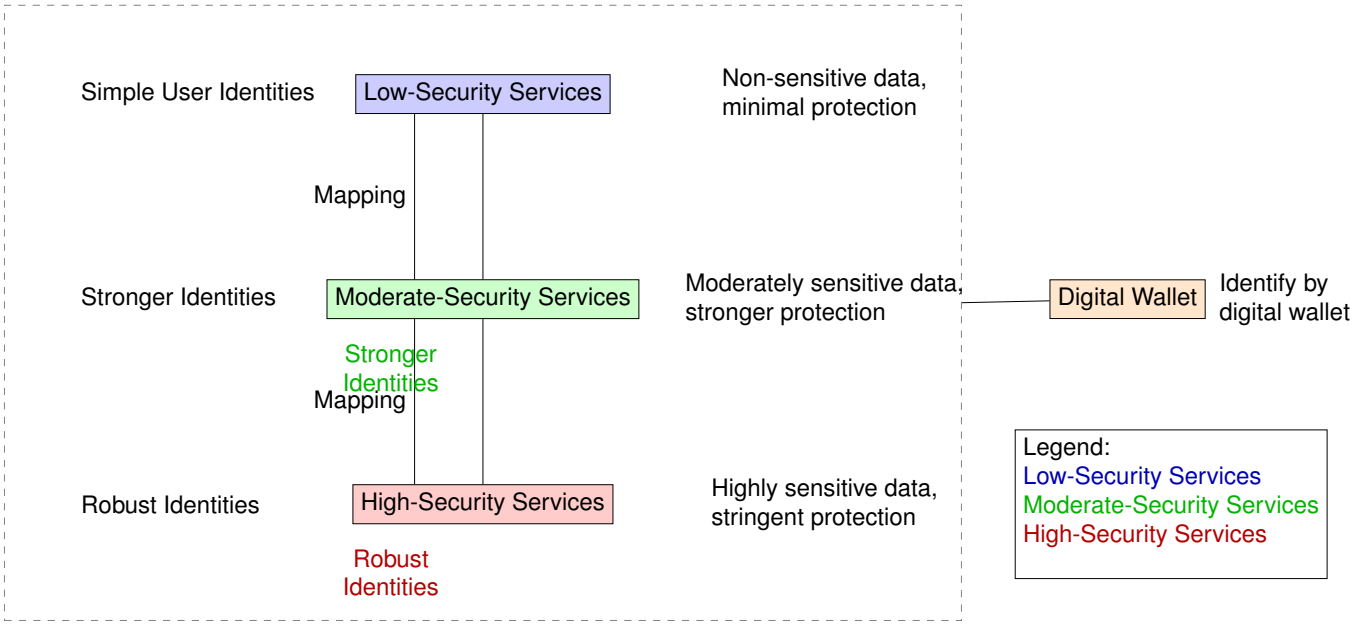

Fig. 1. Categorization of Cloud Services and Identity Types

sensitive data, such as financial records or health information, and demand stringent security measures.

Mapping different types of identities to each category is vital to align the level of security and privacy features with the corresponding cloud service. For low-security services, simple user identities, such as usernames and passwords, may be sufficient for authentication and access control. However, moderate-security services may require stronger authentication mechanisms, such as two-factor authentication or biometrics, to enhance security. High-security services necessitate even more robust identity types, such as digital certificates or hardware tokens, to ensure the highest level of protection and privacy for sensitive data.

To illustrate the practical application of this categorisation scheme, let us consider an example. Suppose a cloud service involves a public-facing web application that provides access to general information about a company. Based on the categorisation scheme, this service would fall under the low-security category. Consequently, a simple user identity, such as a username and password, would be sufficient to authenticate users and manage access to the service. However, if the same company offers a cloud-based Customer Relationship Management (CRM) system that handles customer data, the service would be classified as a moderate-security service. In this case, a stronger identity type, such as two-factor authentication or biometrics, would be necessary to ensure the security and privacy of customer information.

V. Conclusion

In conclusion, digital wallets and federated services offer significant advantages in digital identity management. Digital wallets provide a secure and convenient way for users to store and manage their digital assets, simplifying the management of digital identities while enhancing security and privacy. The emergence of different identity management models, including federated and user-centric approaches, along with advancements like Self-Sovereign Identity (SSI), empower users with greater control over their credentials. Projects like Gaia-X exemplify the aim to give users increased control over their personal information and foster innovation in the digital realm.

Moving forward, future research should focus on integrating emerging technologies such as blockchain and decentralised identity systems to further enhance the security and privacy of digital wallets and federated services. Additionally, exploring the usability and user experience aspects of these solutions can drive their adoption and acceptance among users. Continued efforts in research and development will contribute to addressing the complex challenges of digital identity management and ensure its importance in today's digital era.

Acknowledgement

This research was funded by the Federal Ministry of Education and Research (BMBF) under reference number COSMIC-X 02J21D144, and supervised by Projektträger Karlsruhe (PTKA).

References

[1] P. J. Windley, *Digital Identity: Unmasking identity management architecture (IMA).* " O'Reilly Media, Inc.", 2005.

[2] A. Rashid and A. Chaturvedi, "Cloud computing characteristics and services: a brief review," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 2, pp. 421–426, 2019.

[3] S. Rajamanickam, S. Vollala, R. Amin, and N. Ramasubramanian, "Insider attack protection: Lightweight password-based authentication techniques using ecc," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1972–1983, 2019.

 ISBN: 978-1-68558-043-8

[4] D. R. Malik, D. A. Kataria, and D. N. Nandal, "Analysis of digital wallets for sustainability: A comparative analysis between retailers and customers," *International Journal of Management*, vol. 11, no. 7, 2020.
[5] A. Braud, G. Fromentoux, B. Radier, and O. Le Grand, "The road to european digital sovereignty with gaia-x and idsa," *IEEE network*, vol. 35, no. 2, pp. 4–5, 2021.
[6] B. Otto, "A federated infrastructure for european data spaces," *Communications of the ACM*, vol. 65, no. 4, pp. 44–45, 2022.
[7] M. M. Alam, A. E. Awawdeh, and A. I. B. Muhamad, "Using e-wallet for business process development: Challenges and prospects in malaysia," *Business Process Management Journal*, vol. 27, no. 4, pp. 1142–1162, 2021.
[8] M. A. Hassan and Z. Shukur, "Device identity-based user authentication on electronic payment system for secure e-wallet apps," *Electronics*, vol. 11, no. 1, p. 4, 2022.
[9] S. Gajek, H. Löhr, A.-R. Sadeghi, and M. Winandy, "Truwallet: trustworthy and migratable wallet-based web authentication," in *Proceedings of the 2009 ACM workshop on Scalable trusted computing*, 2009, pp. 19–28.
[10] J. Sedlmeir, R. Smethurst, A. Rieger, and G. Fridgen, "Digital identities and verifiable credentials," *Business & Information Systems Engineering*, vol. 63, no. 5, pp. 603–613, 2021.
[11] R. Dhamija and L. Dusseault, "The seven flaws of identity management: Usability and security challenges," *IEEE Security & Privacy*, vol. 6, no. 2, pp. 24–29, 2008.
[12] B. Zwattendorfer, T. Zefferer, and K. Stranacher, "An overview of cloud identity management-models." *WEBIST (1)*, pp. 82–92, 2014.
[13] B. Pfitzmann and M. Waidner, "Privacy in browser-based attribute exchange," in *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, 2002, pp. 52–62.
[14] N. Selvanathan, D. Jayakody, and V. Damjanovic-Behrendt, "Federated identity management and interoperability for heterogeneous cloud platform ecosystems," in *Proceedings of the 14th international conference on availability, reliability and security*, 2019, pp. 1–7.
[15] C. Cuijpers and J. Schroers, "eidas as guideline for the development of a pan european eid framework in futureid," 2014.
[16] S.-H. Kim, S.-R. Cho, and S.-H. Jin, "Context-aware service system architecture based on identity interchange layer," in *2008 10th International Conference on Advanced Communication Technology*, vol. 2. IEEE, 2008, pp. 1482–1486.
[17] A. Abraham, C. Schinnerl, and S. More, "Ssi strong authentication using a mobile-phone based identity wallet reaching a high level of assurance." in *SECRYPT*, 2021, pp. 137–148.
[18] D. Du Seuil, "European self sovereign identity framework," 2019.
[19] "Veramo," https://veramo.io/, 2023, accessed: Jun 27, 2023.
[20] T. South and R. Mahari, "Justice in a vaccum?" 2023.
[21] A. Kudra, "Self-sovereign identity (ssi) in deutschland: Projekte mit strahlkraft für die globale community," *Datenschutz und Datensicherheit-DuD*, vol. 46, no. 1, pp. 22–26, 2022.
[22] M. Gaedke, J. Meinecke, and M. Nussbaumer, "A modeling approach to federated identity and access management," in *Special interest tracks and posters of the 14th international conference on World Wide Web*, 2005, pp. 1156–1157.
[23] V. Siska, V. Karagiannis, and M. Drobics, "Building a dataspace: Technical overview," 2023.
[24] K. Bernsmed, M. G. Jaatun, P. H. Meland, and A. Undheim, "Security slas for federated cloud services," in *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE, 2011, pp. 202–209.

 ISBN: 978-1-68558-043-8