A Technical Reference Architecture for Microservices-based Applications in the Insurance Industry

Arne Koschel*
Andreas Hausotter*
Christin Schulze*
*Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hanover, Germany
Email: arne.koschel@hs-hannover.de
andreas.hausotter@hs-hannover.de

Henrik Meyer†
Alexander Link*
Tim van Dorp*
†Capgemini
Hanover, Germany
Email: henrik.meyer@capgemini.com

***Abstract*—To overcome the shortcomings of traditional monolithic applications, the Microservices Architecture (MSA) style is playing an increasingly vital role in the provision of business services. This also applies to the insurance industry, which is facing challenges like cut-throat competition and decreasing customer loyalty. Providing scalable and resilient services of high availability in a flexible and agile manner, which comes with the MSA style, is undoubtedly a competitive advantage. However, the insurance industry's application landscape is characterized by the coexistence of historically grown systems based on different architectural paradigms. Therefore, the integration of microservices with Service-Oriented Architecture (SOA) services or even legacy systems induces additional complexity. A reference architecture may lower the complexity of this integration task by defining an architectural framework of MSA-based applications in a heterogeneous environment. In this contribution, we present a technical reference architecture for our partner insurance companies. The reference architecture is shaped along a cloud-native approach to provide good scalability, short release cycles, and high resilience. As a key feature, a technical microservice supports the integration of SOA services.**

***Keywords*—Microservices Architecture; Reference Architecture; Cloud Native; SOA; Insurance Industry**

I. Introduction

A long-lasting trend in software engineering is to divide software into lightweight, independently deployable components. Each component can be implemented using different technologies because they communicate over standardized interfaces. This approach to structuring the system is known as the MSA style [1]. A study from 2019 (see [2]) shows, the MSA style is already established in many industries such as e-commerce. However, this is rarely the case for the insurance services industry.

Our current research is the most recent work of a long-standing, ongoing applied research–industry cooperation on service-based systems. This includes cooperative work on traditional SOA, Business Rules and Business Process Management (BRM/BPM), SOA-Quality of Service (SOA-QoS), and microservices ([3]–[6]), between the *Competence Center Information Technology and Management (CC_ITM)* from the University of Applied Sciences and Arts Hanover and two regional, medium-sized German insurance companies. The ultimate goal of our current research is to develop a 'Microservice Reference Architecture for Insurance Companies' (RaMicsV) jointly with our partner companies. This shall allow building microservice conformant insurance application systems or at least such system parts.

For this purpose, however, several cornerstones and resulting challenges exist frequently in at least the German industry domain. Especially, insurance companies rarely start development 'in the green field', but must integrate and comply with existing application systems, like Knoche and Hasselbring showed in [7]. This requirement stems from building complex systems over multiple years, or decades even, where a big bang replacement is too risky and cost intensive. For example, our partners both operate a SOA and additional 3rd party software, such as SAP systems, which both have significantly different characteristics, for example, for testing, release cycles, versioning, administration etc.

Nowadays, our partners would like to get the promised benefits of microservices, such as improved scalability, both technical and organizational through parallel execution and also parallel development, significantly faster release cycles (a few weeks or even days instead of quarters or several months) etc. However, a microservices-based approach to help them must still work well in 'cooperative existence' with their existing systems and SOA services. Thus, improvements or partial replacements of their existing software landscape for particular goals using microservices is fine, but a complete migration to the Microservices Architecture style is not a feasible option. The main goal being extension of a system, and not replacing it.

In our previous work [8], we already developed RaMicsV, a logical reference architecture, which takes those insurance industry specifics into account. Moreover, we started to explore parts of it, such as logging, monitoring, security, workflows,

 ISBN: 978-1-68558-162-6

and choreographies, in more depth [8]–[10].

As the major contribution of this article, we present for the first time, a technical reference architecture for RaMicsV. Our technical reference architecture (T-RaMicsV) is based on a cloud-native approach for microservices including, for example, containerization, message-driven communication and an ESB-wrapper microservice. This also marks the first practical implementation of this architecture and works as a proof of concept.

We organize the remainder of this article as follows: After discussing related work in Section II, we briefly repeat RaMicsV in Section III. Next, Section IV and Section V contribute our cloud-native approach to the technical reference architecture based on RaMicsV. Section VI concludes and looks at future work.

II. Related Work

The foundations of this work relate to the concepts of Microservices Architecture, cloud computing, cloud-native architecture and practical application within the insurance industry.

Our research builds on authors in the field of microservices, such as the work of Newman [11], Fowler and Lewis [1], as well as Fachat [12]. When designing our reference architecture, we benefit from various microservices patterns as discussed by Richardson [13].

To design T-RaMicsV, we use a cloud-native approach that is definitionally based on the descriptions of the Cloud Native Computing Foundation [14]. Furthermore, we supplement our understanding with the aspects of cloud computing, introduced by the National Institute of Standards and Technology (*NIST*) [15], as well as containerization, automation, and observability.

In this contribution, we lay out T-RaMicsV, which is derived from RaMicsV [16] as our foundation. An implementation of RaMicsV that demonstrates its technical feasibility, as envisaged in this paper, has not yet taken place.

Regarding its realization, a technical reference architecture must be developed that makes fundamental statements about the technologies used, such as programming languages or infra-structure. In accordance to Angelov, Grefen, and Greefhorst [17] T-RaMicsV can be categorized as a 'semi-concrete type 4' reference architecture, i.e., indicating technology choices to be implemented in one single organization [17].

In this context, the technology-agnostic approach of microservices is broken with, to provide practical specifications that harmonize with the existing system landscape of an insurance company. The cloud-native approach used in this contribution is a conceivable option that seems promising to us, to realize RaMicsV technically.

Deriving a technical reference architecture from a logical one, like RaMicsV, seems to us to be a common practice. Furthermore, there are contributions to reference architecture for microservices for broader enterprise context, e.g., by Yu, Silveira, and Sundaram in [18]. To our knowledge, however, this has not yet been done for insurance companies such as our industry partners and their specific requirements.

These operate a historically grown heterogeneous system landscape characterized by an existing SOA. The use of microservices, that is embedded in the cloud-native approach, must be integrated in a cooperative manner, which plays an important role in our overall architectural considerations.

We are currently evaluating T-RaMicsV on a typical car insurance process [19]. In previous work, we have implemented other typical insurance use cases, as can be seen in [20]–[22].

III. Service-based Reference Architecture for Insurance Companies

This section presents our RaMicsV as initially started in [8]. 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 operated SOA landscape, which seems suitable for our industry partners for several years to come. Thus, from their perspective, the Microservices Architecture style is only suitable as an additional enhancement and only a piecewise replacement 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.

 ISBN: 978-1-68558-162-6

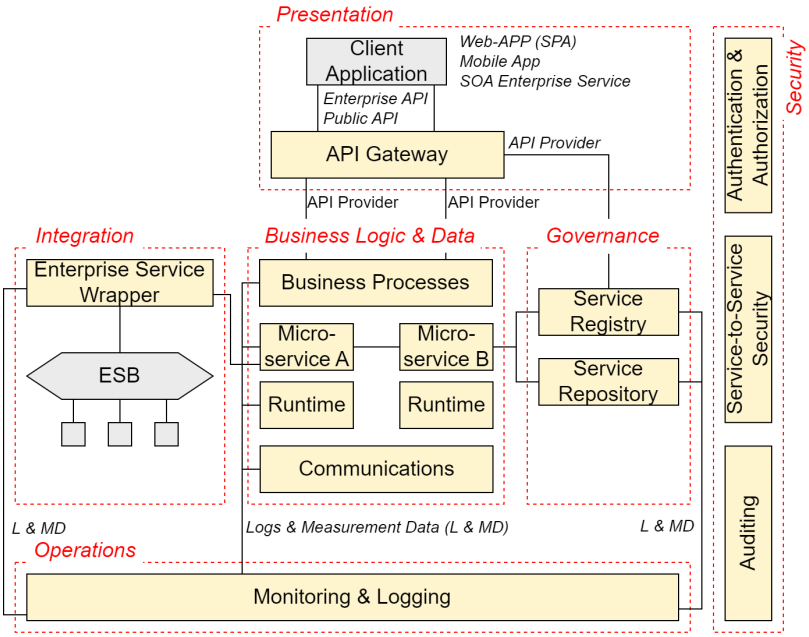

Figure 1. Building Blocks of the Logical Reference Architecture RaMicsV.

- **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 produce unified monitoring, logging, and tracing, 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 present our understanding of cloud-native architecture, which represents our approach to developing T-RaMicsV as this paper's contribution.

IV. Cloud-native Architecture

The architectural approach of our contribution is based on the following five aspects of a cloud-native architecture.

- **Cloud Computing** means that an IT service is offered by a cloud service provider and used by a cloud service consumer. The NIST [15] defines Cloud Computing via five **characteristics**: *On-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity*, and *measured service*. Different **deployment models** (*Public*, *Private*, *Community*, and *Hybrid*) are used to describe ownership and control of the cloud environment, while **service models** describe the level of abstraction from a consumer perspective (*Software-*, *Function-*, *Platform-*, *Container-*, and *Infrastructure as a Service*. Abbreviated to SaaS, FaaS, PaaS, CaaS, and IaaS respectively). CaaS and FaaS are often used in the industry.
- **Microservices** are considered – clearly not the only – but the 'most native' architectural style in cloud-native architectures. They grant loose coupling, as well as independent scaling and deployment [13]. These services are modeled around business functionality and gain high cohesion in the process.
- **Containerization** is the process of deploying software for a cloud-native architecture as isolated, virtualized units. Containers offer efficient hardware usage and dynamic resource allocation.
- **Automation** in a cloud-native architecture aims at automated deployments of microservices and new functionality. To achieve this goal, concepts like Continuous Integration and Continuous Deployment (CI/CD) [11] and Infrastructure as Code (IaC) [23] play a vital role.
- **Observability** is the aggregation and analysis of data in a system to gain transparency and understanding about the internal components [24]. This supports troubleshooting in a microservices-based system, where data and business logic is distributed [11].

The following section will provide insight into our cloud-native technical reference architecture, which was designed

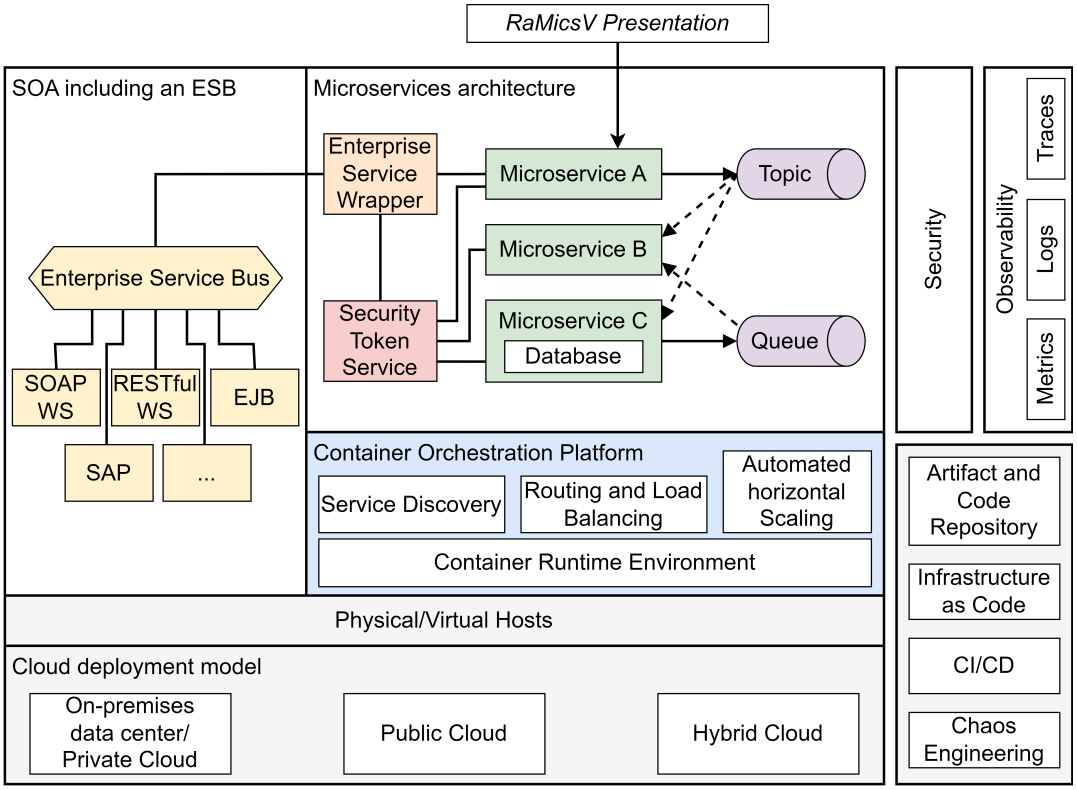

Figure 2. Building Blocks of the Technical Reference Architecture T-RaMicsV.

with our business partners in mind.

V. Cloud-native Technical Reference Architecture for Insurance Companies

This section provides an overview of T-RaMicsV (see Figure 2). It has been designed according to a cloud-native approach, which is expected to provide good scalability, short release cycles and high resilience.

T-RaMicsV provides for private cloud and public cloud as the underlying operating model. The host machines of all elements of the architecture are physical or virtual machines, whereby the former can also include mainframes typical for insurance companies. In line with the cloud operating model, the host machines are operated in the company's own data center or booked as IaaS solutions from public cloud service providers. A hybrid cloud scenario is also made possible by T-RaMicsV. In the insurance context, for example, it would be conceivable to use only certain IT services in the public cloud while continuing to process highly regulated, sensitive customer data in the company's own data center.

The chosen cloud-native approach implies the operation of microservices as containerized workloads. Thus, a central element of the reference architecture is the container orchestration platform (COP). It is operated by the company itself or purchased as a managed service from a public cloud service provider. All microservices and their databases, the Enterprise Service Wrapper (ESW), a MoM for message-driven communication, and the Security Token Service (STS) are operated in containers. These run in a container runtime environment managed by the container orchestration platform, which also distributes them across multiple host machines. A microservice is horizontally scaled in an automated fashion by the platform as a set of containers, where one container corresponds to one service instance.

Furthermore, the platform performs the task of service discovery, i.e., a mechanism, by which the microservices of the architecture can find and address each other. In addition, the COP provides routing capabilities and allows load balancing between multiple instances of a microservice.

The MSA includes business-oriented microservices, named A to C as examples in Figure 2. These mainly communicate in an asynchronous, message-driven fashion via topics or queues provided by the MoM. Depending on persistence needs, a microservice may exclusively use its own database.

For authentication and authorization purposes, microservices exchange JSON Web Tokens (JWT) with each other as security tokens. These are issued by the STS of the architecture, to which the microservices must authenticate themselves. In addition to the use of JWT, T-RaMicsV stipulates that microservices communicate with each other in an mTLS-encrypted form. To gain transparency in the architecture, metrics, logs, and traces of the services are collected. These can be aggregated, processed and analyzed in corresponding solutions.

As in correspondence with RaMicsV the SOA with ESB

is understood as an existing system landscape part in T-RaMicsV, in which existing SOA services are operated. The connection to the ESB is a classic use case in the insurance context. The legacy systems characterized by SOA cannot be replaced by microservices, but should rather be extended. The insurance partners are aiming for the coexistence of SOA and microservices.

In the insurance industry, for example, SOAP- or REST-based web services, Enterprise Java Beans (EJB), or SAP systems that are offered as SOA services are conceivable. The SOA with ESB is run on host machines in the company's own data center, but not on the container orchestration platform. The Microservices Architecture on the COP is considered a part of the landscape, in which new services can be realized as microservices. Existing SOA services could be migrated successively to the Microservices Architecture as required.

T-RaMicsV provides a proposal for the ESW, which is a component directly derived from RaMicsV. The ESW acts as a central transition from the Microservices Architecture to the ESB of the SOA. It is accessed by one or more microservices via REST when they need to consume a SOA service.

The software delivery process is planned to be automated as much as possible. For this purpose, CI/CD pipelines are used in which the build, test, and delivery of artifacts is carried out. Provisioning of virtual infrastructure, especially the one provided by a public cloud service, should be based on the IaC principle. The infrastructure and application code as well as the software artifacts are captured and versioned in appropriate repositories. With the aim of increasing the resilience of the Microservices Architecture in production, it may additionally be tested using Chaos Engineering methods.

VI. Conclusion and Future Work

In this contribution, we present a cloud-native based technical reference architecture that aims to build compliant microservices-based applications for insurance companies. The architecture adopts our partner's special requirements, such as integrating SOA services.

To evaluate T-RaMicsV, it needs to be applied to a business process, typical for the insurance industry. We are currently evaluating the technical realization by means of the car insurance process in order to evaluate the feasibility in the insurance industry [19]. In this context, we aim to shape an application-specific architecture that uses a concrete selection of technologies in accordance to T-RaMicsV. For example, the open-source system Kubernetes [25], which is well-known in the cloud-native sector, could be considered for implementing the COP (see block 'Container Orchestration Platform', Figure 2).

Furthermore, our future investigations may also, for example, include the integration of a prominent public cloud solution like Microsoft Azure (see block 'Cloud deployment model', Figure 2). The findings may cause the need to enhance or even redesign our technical reference architecture.

Another topic is the refinement of the building block Business Processes within the responsibility area Business Logic & Data (see Figure 1). Choreography is the preferred approach to implementing workflows. Based on BPMN Choreography diagrams [26], we identified and classified frequently occurring patterns in the context of the insurance industry. Our goal is to implement the patterns to make choreography diagrams executable.

References

[1] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term," 2014, Online. Available: https://martinfowler.com/articles/microservices.html [retrieved: 04, 2024].

[2] H. Knoche and W. Hasselbring, "Drivers and Barriers for Microservice Adoption–A Survey among Professionals in Germany," *Enterprise Modelling and Information Systems Architectures*, vol. 14, p. 10, 2019.

[3] A. Hausotter, C. Kleiner, A. Koschel, D. Zhang, and H. Gehrken, "Always Stay Flexible! WfMS-independent Business Process Controlling in SOA," in *15th IEEE Intl. Enterprise Distributed Object Computing Conference Workshops*. IEEE, 2011, pp. 184–193.

[4] 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 On Advances in Intelligent Systems*, vol. 9, no. 3,4, pp. 287–297, Dec. 2016, [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=intsys_v9_n34_2016_6. [retrieved: 04, 2024].

[5] A. Hausotter, A. Koschel, J. Busch, and M. Zuch, "A Flexible QoS Measurement Platform for Service-based Systems," *Intl. Journal On Advances in Systems and Measurements*, vol. 11, no. 3,4, pp. 269–281, Dec. 2018, [Online]. Available: https://www.thinkmind.org/index.php?view=article\&articleid=sysmea_v11_n34_2018_4. [retrieved: 04, 2024].

[6] A. Koschel, A. Hausotter, M. Lange, and P. Howeihe, "Consistency for Microservices - A Legacy Insurance Core Application Migration Example," in *SERVICE COMPUTATION 2019, 11th Intl. Conf. on Advanced Service Computing*, Venice, Italy, 2019, [Online]. Available: https://thinkmind.org/index.php?view=article&articleid=service_computation_2019_1_10_18001. [retrieved: 04, 2024].

[7] H. Knoche and W. Hasselbring, "Using Microservices for Legacy Software Modernization," *IEEE Software*, vol. 35, no. 3, pp. 44–49. [Online]. Available: https://ieeexplore.ieee.org/document/8354422/

[8] 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, 2024].

[9] A. Hausotter, A. Koschel, M. Zuch, J. Busch, and J. Seewald, "Microservices Authentication and Authorization from a German Insurances Perspective," *Intl. Journal od Advances in Security*, vol. 15, no. 3 & 4, pp. 65–74, 2022, Online. Available: https://www.iariajournals.org/security/tocv15n34.html [retrieved: 04, 2024].

[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, 2024].

[11] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, Kalifornien, USA: O'Reilly Media, Inc., 2021.

[12] André Fachat. Challenges and benefits of the microservice architectural style, part 1. [Online]. Available: https://developer.ibm.com/articles/challenges-and-benefits-of-the-microservice-architectural-style-part-1/

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

[14] Linux Foundation, "Cloud Native Computing Foundation Charter," 2023, Online. Available:https://github.com/cncf/ [retrieved: 04, 2024].

[15] P. Mell and T. Grance, *NIST Special Publication 800-145: The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, Maryland, USA, 2011, Online. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf [retrieved: 12, 2024].

[16] 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*, 2021, pp. 5–9, Online. Available: https://www.thinkmind.org/articles/service_computation_2021_1_20_10002.pdf [retrieved: 04, 2024].

[17] S. Angelov, P. Grefen, and D. Greefhorst, "A classification of software reference architectures: Analyzing their success and effectiveness," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, 2009, pp. 141–150.

[18] Y. Yu, H. Silveira, and M. Sundaram, "A microservice based reference architecture model in the context of enterprise architecture," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016, pp. 1856–1860.

[19] A. Koschel, A. Hausotter, C. Schulze, A. Link, T. van Dorp, and H. Meyer, "Towards a Technical Reference Architecture for Microservice-based Applications in the Insurance Industry," 2024, [Manuscript submitted for publication].

[20] M. Lange, S. Gottwald, A. Koschel, and A. Hausotter, "Keep it in Sync! Consistency Approaches for Microservices An Insurance Case Study," *SERVICE COMPUTATION 2020, The 12h Intl. Conference on Advanced Service Computing*, pp. 7–10, 2019, Online. Available: https://www.thinkmind.org/index.php?view=article&articleid=service_computation_2020_1_20_10016 [retrieved: 04, 2024].

[21] M. Lange, , A. Koschel, and A. Hausotter, "Applied SOA with ESB, BPM, and BRM - Architecture Enhancement by Using a Decision Framework," *SERVICE COMPUTATION 2016: The 8th Intl. Conferences on Advanced Service Computing*, pp. 20–27, 2016, Online. Available: https://personales.upv.es/thinkmind/dl/conferences/servicecomputation/service_computation_2016/service_computation_2016_2_10_10013.pdf [retrieved: 04, 2024].

[22] ——, "Microservices in Higher Education - Migrating a Legacy Insurance Core Application," *2nd International Conference on Microservices*, pp. 1–8, 2019, Online. Available: https://www.conf-micro.services/2019/papers/Microservices_2019_paper_8.pdf [retrieved: 04, 2024].

[23] K. Morris, *Infrastructure as Code: Dynamic Systems for the Cloud Age*, 2nd ed. Sebastopol, Kalifornien, USA: O'Reilly Media, Inc., 2021.

[24] C. Majors, L. Fong-Jones, and G. Miranda, *Observability Engineering: Achieving Production Excellence*. Sebastopol, Kalifornien, USA: O'Reilly Media, Inc., 2022.

[25] The Kubernetes Authors, "Kubernetes," 2023, Online. Available: https://kubernetes.io/ [retrieved: 04, 2024].

[26] 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, 2024].

 ISBN: 978-1-68558-162-6