
# SERVICE COMPUTATION 2025

The Seventeenth International Conferences on Advanced Service Computing

ISBN: 978-1-68558-257-9

April 6 - 10, 2025

Valencia, Spain

**SERVICE COMPUTATION 2025 Editors**

Petre Dini, IARIA, USA/EU

# SERVICE COMPUTATION 2025

# Forward

The Seventeenth International Conferences on Advanced Service Computing (SERVICE COMPUTATION 2025), held on April 6 – 10, 2025, 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 2025 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 2025. 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 2025 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope SERVICE COMPUTATION 2025 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 Valencia provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city.

**SERVICE COMPUTATION 2025 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, National University of Science and Technology POLITEHNICA Bucharest, Romania
Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Hannover, Germany
Ozgu Can, Ege University, Turkey

**SERVICE COMPUTATION 2025 Publicity Chair**

Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

# SERVICE COMPUTATION 2025

## Committee

### SERVICE COMPUTATION 2025 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, National University of Science and Technology POLITEHNICA Bucharest, Romania
Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Hannover, Germany
Ozgu Can, Ege University, Turkey

### SERVICE COMPUTATION 2025 Publicity Chairs

Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

### SERVICE COMPUTATION 2025 Technical Program Committee

Jocelyn Aubert, Luxembourg Institute of Science and Technology (LIST), Luxembourg
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Eugen Borcoci, National University of Science and Technology POLITEHNICA Bucharest, Romania
Uwe Breitenbücher, Reutlingen University, 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
Patrizio Dazzi, University of Pisa, Italy
Marco Di Girolamo, Hewlett Packard Enterprise, USA
Leandro Dias da Silva, Universidade Federal de Alagoas, Brazil
Erdogan Dogdu, Angelo State University, USA
Monica Dragoicea, National University of Science and Technology POLITEHNICA Bucharest, Romania
Stefano Forti, University of Pisa, Italy
Sören Frey, Mercedes-Benz Tech Innovation, Germany
Steffen Fries, Siemens Corporate Technology - Munich, Germany
Somchart Fugkeaw, Sirindhorn International Institute of Technology | Thammasat University, Thailand
Ankit Garg, Netaji Subhas University of Technology, India
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

**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**

# Insight into Integrating Legacy Systems and Microservices Architectures within the Insurance Industry via an Enterprise Service Bus

Arne Koschel
Andreas Hausotter
Christin Schulze
Tim van Dorp
Lennart Vermehr
Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hanover, Germany
Email: {andreas.hausotter@, arne.koschel@,
christin.schulze@stud., tim.van-dorp@stud.,
lennart.vermehr@stud.}hs-hannover.de

Henrik Meyer
Capgemini
Hanover, Germany
Email: henrik.meyer@capgemini.com

***Abstract*—Microservices Architectures (MSA) have become increasingly popular due to their ability to implement new requirements, enhancing organizational competitiveness quickly. However, their integration with legacy systems poses significant challenges, especially in industries like insurance, where greenfield projects are rare. This research addresses the complexities of integrating microservices with Service-Oriented Architectures (SOA) and third-party systems, such as SAP, with specific requirements (e.g., release cycles, versioning). While complete migration to MSA is unrealistic in the Insurance Industry, targeted improvements and partial replacements are achievable. Integration must also account for existing Enterprise Service Bus (ESB)-based infrastructures. Building on previous work, we developed RaMicsV, a logical reference architecture, and its technical implementation, T-RaMicsV. This paper expands on earlier findings by providing a detailed exploration of integrating microservices with ESB-based SOA environments. The results offer practical insights into adapting microservices for insurance companies, focusing on logging, monitoring, security, workflows, and choreographies while maintaining legacy system compatibility.**

***Keywords*—Ingetration; ESB; SOA; Microservices.**

## I. INTRODUCTION

Software architectures are constantly changing and being explored even further. In particular, Microservices Architectures (MSA) have become increasingly popular in recent years. MSA offers the advantage of being able to implement new requirements quickly. Companies remain highly competitive. However, one of the biggest challenges is connecting and integrating these new systems with existing legacy systems.

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 ( [1]–[4]), 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. Our current research aims to jointly develop a 'Microservice Reference Architecture for Insurance Companies' (RaMicsV) with our partner companies. This shall allow the building of microservice-compliant insurance application systems or at least such system parts.

The German industry sector often faces specific key challenges and foundational requirements in this context. Specifically, insurance companies rarely introduce development projects "on a greenfield" but must integrate with and adapt to their existing application systems. For instance, our partners operate a Service-Oriented Architecture (SOA) in conjunction with legacy systems and additional third-party software, such as SAP systems, which have distinct characteristics, including differences in testing, release cycles, versioning, and administration.

Currently, our partners are eager to realize the promised advantages of microservices, such as enhanced technical and organizational scalability through parallel execution and development and significantly accelerated release cycles (reducing the timeline from quarters or months to weeks or even days).

However, any microservices-based approach must coexist seamlessly with their established systems and SOA services. While targeted improvements or partial replacements within the software landscape using microservices are viable, a complete migration to a microservices architecture is not practical.

Our partners' existing SOA environments are traditionally interconnected via an Enterprise Service Bus (ESB), meaning any new systems must also be compatible with this infrastructure. We will discuss one possible way of realizing this via an Enterprise Service Bus Wrapper in this paper.

In our previous work [5], we already developed RaMicsV, a logical reference architecture considering those insurance industry specifics. Moreover, we explored parts of it, such as



 ISBN: 978-1-68558-257-9 

logging, monitoring, security, workflows, and choreographies, in more depth [5]–[7].

We have technically conceptualized this logical reference architecture and derived a corresponding technical reference implementation, termed T-RaMicsV [8]. RaMicsV serves as a guideline, incorporating best practices and modular building blocks for microservice architectures in the insurance industry. To validate RaMicsV and to demonstrate a feasible technical implementation, we developed T-RaMicsV. However, T-RaMicsV does not aim to establish a standard; it merely represents one possible technical realization of the logical reference architecture. A previous paper described this approach as cloud-native [9]. Additionally, integration via an ESB was a specific requirement of our insurance partners.

This article's main contribution is that, for the first time, we describe integration via the ESB in detail. This integration describes only a part of T-RaMicsV.

We organize the remainder of this article as follows: After some related work in Section II, we present the logical reference architecture in Section III and the technical reference architecture in Section IV. This is followed by a description of the integration via the ESB in Section V. The paper concludes with a summary and an outlook in Section VI.

## 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 [10], as well as Fowler and Lewis [11]. When designing our reference architecture, we benefit from various microservices patterns as discussed by Richardson [12].

We were in close contact with our partners from the insurance industry to design the logical reference architecture. [5] To implement the Technical Reference Architecture [13], 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.

The ESB has been used by enterprises to manage and communicate in a SOA since the early 2000s. Chappell [16] and Hohpe and Woolf [17] already describe the ESB and its use in their books. These books also describe how SOA/ESB architectures can be connected to other systems. The authors use the term adapter. Adapters enable the seamless integration of different applications and technologies into an ESB architecture by performing protocol and data format conversions. They act as intermediaries between the ESB and the systems to be integrated by transforming incoming and outgoing messages accordingly.

For the ESB Wrapper we adopt approaches from earlier work, such as Chappell [16] and Hohpe and Woolf [17]. Our approach is to create an ESB Wrapper to integrate existing SOA/ESB and MSA. The approach is application-specific and was developed for the use case of our insurance partners. This wrapper represents an implementation of our Technical Reference Architecture.

In accordance to Angelov et al., our Technical Reference Architecture can be categorized as a 'semi-concrete type 4' reference architecture, i.e., indicating technology choices to be implemented in one single organization [18]. 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, is a common practice, as discussed in [19]. Furthermore, there are contributions to the reference architecture for microservices for broader enterprise context, e.g., by Yu et al. in [20]. 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 embedded in the cloud-native approach must be integrated cooperatively, which plays an essential role in our overall architectural considerations.

We are using an exemplary insurance industry business process to implement the approach in practice. For this purpose, we have chosen car insurance, one of the core products of German insurers. The authors Stadler and Gail [21] provide the basis for the process. A car insurance is compulsory for every car in Germany. For this reason, it is considered particularly important for acquiring new customers. The elaboration refers to the VAA [22] and describes in detail what car insurance is all about and much more. We use the Business Process Model and Notation (BPMN) to realize the processes [23], as it is widely used in the insurance industry. We present this process in a practical implementation of the technical reference architecture [9].

## III. Service-based Reference Architecture for Insurance Companies

This section presents our RaMicsV as initially started in [5]. In this paper, we briefly describe RaMicsV and then go into the technical implementation, especially the ESB.

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.

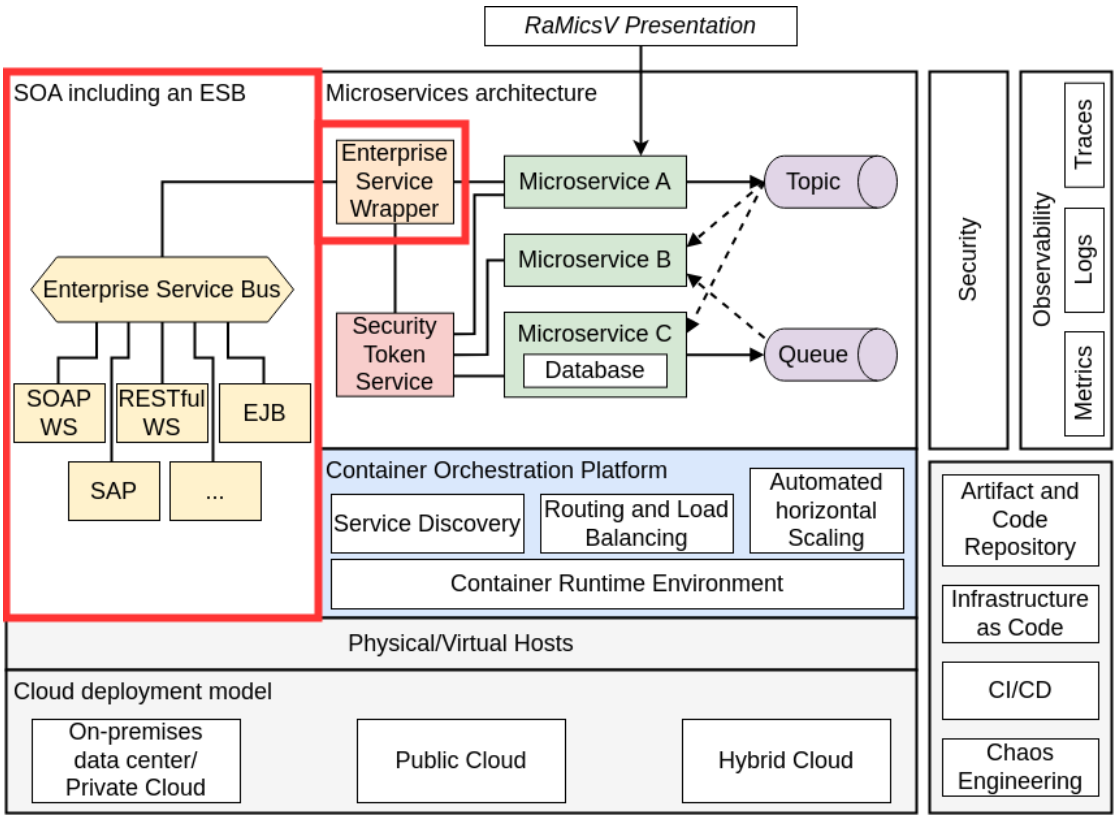


Figure 1. Technical Reference Architecture for RaMicsV.

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. Thus, from their perspective, the Microservices Architecture (MSA) style is only appropriate as an additional enhancement and only a partial replacement of parts from their SOA or other self-developed applications. The connection to the ESB and the related legacy systems is described in more detail in Section V.
- 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 its competitive edge, the enterprise must change its processes in a flexible and agile manner. RaMicsV must, therefore, provide suitable solutions to implement workflows while ensuring the required flexibility and agility.

RaMicsV is divided into building blocks, such as presentation, business logic & data, governance, integration, operation, and security.

The building blocks perform the following tasks:

- **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 produce 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

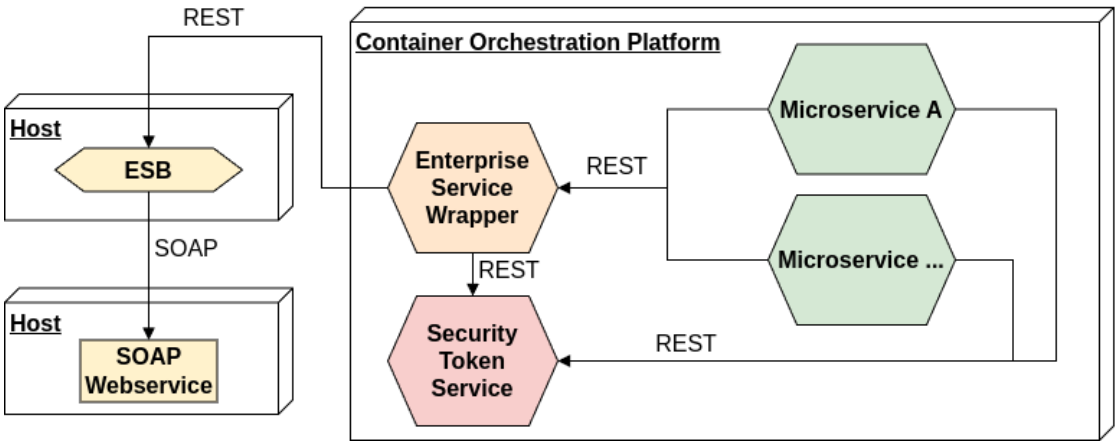


Figure 2. Exemplary integration of SOA and MSA in T-RaMicsV.

Middleware (MOM) or the ESB. The ESB is part of the integration building block and contains a message broker.

In the next section, we present our technical reference architecture, which concretizes the logical one by specifying technical concepts. For further explanations of RaMicsV, we refer to the paper [5].

## IV. Cloud-native Technical Reference Architecture for Insurance Companies

This section provides an overview of the Technical Reference Architecture (see Figure 1). It has been designed according to a cloud-native approach, which is expected to provide good scalability, short release cycles and high resilience.

The Technical Reference Architecture 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 the Technical Reference Architecture. 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 microservice instances.

The MSA includes business-oriented microservices, named A to C, as examples in Figure 1. 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 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. The Technical Reference Architecture stipulates that microservices communicate with each other in an mTLS-encrypted form in addition to using JWT.

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 the Technical Reference Architecture, in which existing SOA services are operated. In the insurance industry, for example, SOAP—or REST-based web services, Enterprise Java Beans (EJB), or SAP systems offered as SOA services are conceivable. The SOA with ESB is run on host machines in the company's 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 (see red parts in Figure 1). Further details on the ESW are covered in Section V.

The Technical Reference Architecture provides a proposal for the ESW, which is a component directly derived from

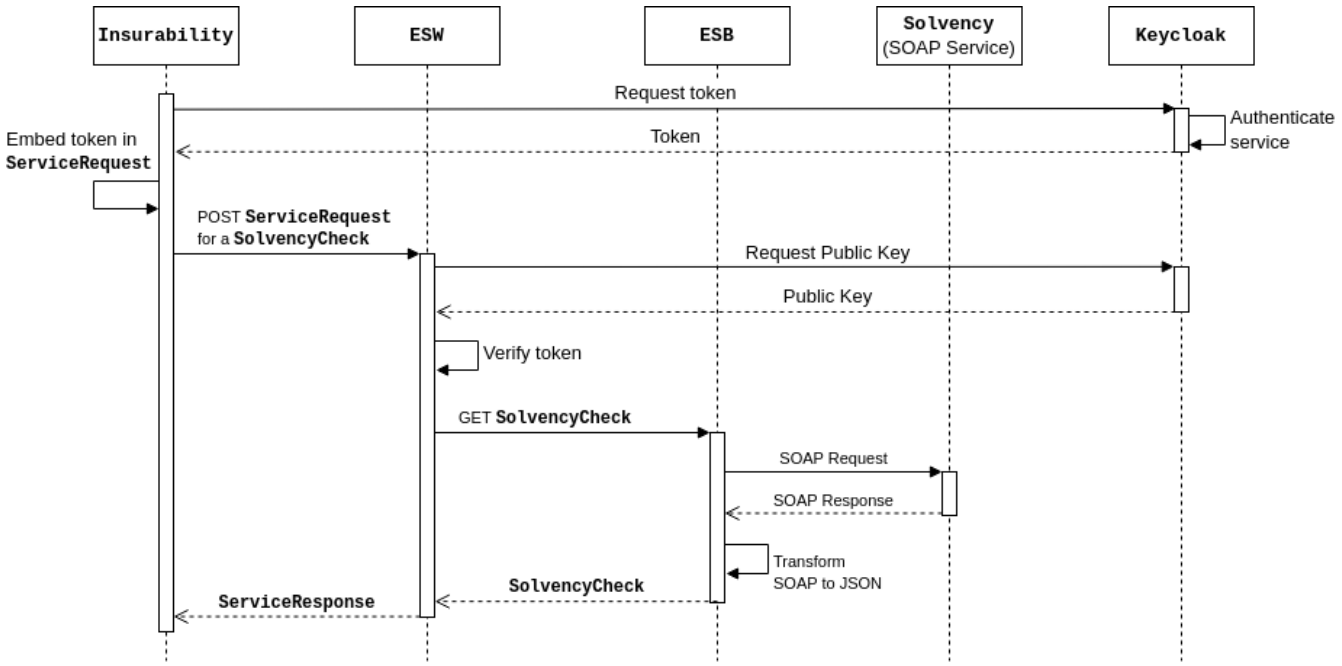


Figure 3. Exemplary call chain for a microservice consuming a SOAP service in T-RaMicsV.

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. Moreover, to increase the resilience of the microservices architecture in production, it may be tested using Chaos Engineering methods.

In the following, we describe the integration via the ESB in the areas of connectivity, logging, and security.

## V. ESB AND WRAPPER

The Enterprise Service Wrapper serves as a crucial central component in the microservices architecture, acting as a gateway for all SOA service consumption and ESB communication such as, for example, consuming a SOAP-based web service (see Figure 2). The wrapper represents the sole point of interaction between microservices and the enterprise's SOA and acts as a kind of "gateway" to the SOA for the microservices, as well as in terms of security.

As the only transition component to the ESB, there is a risk that the wrapper represents a single point of failure. To compensate for this, the ESW is provided as a technical microservice in the technical reference architecture. This is also operated and replicated in containers on the container orchestration platform. The horizontal scaling and the use of the platform's advantages, such as readiness checks and self-recovery mechanisms, can increase the availability of the wrapper. This means that the reliability of the wrapper can be increased to a practically appropriate level with a sufficiently high number of replicas.

Figure 3 shows a concrete insurance-related example of the call chain that a microservice triggers when it wants to consume a SOAP service of the SOA. The Insurability service is a microservice that delegates a solvency check to the Solvency SOAP service to verify the insurability of a customer. The connection is realized via the ESW and the ESB, whereas Keycloak is used as the STS that provides security tokens.

### A. Connectivity

The Enterprise Service Wrapper connects the MSA to the ESB by translating requests from the microservices to the ESB. The request from the wrapper is in a REST style, with a uniform service request format in JSON and a single endpoint for all microservices. The format should include at least the name of the SOA service being used, the SOA operation of the service being used, and the required service data. The wrapper translates the service request of a microservice into a REST call at the endpoint of the ESB. To do this, the wrapper needs to know where and how to call the ESB to consume the appropriate SOA service. For this, we assume that the wrapper can query the Service Registry within the governance area of RaMiscV to access the required information about the used SOA Service. The actual call and the technical details of the ESB remain hidden from the microservice that made the original request to the wrapper. This means that the wrapper acts as an SOA interface for the microservices. The response

from the ESB is translated into a uniform service response format in JSON and sent back to the original requester.

### B. Observability

It is reasonable to implement observability of the requests that are sent from the MSA to the ESB in order to support the transparency of the communication. This can be used as a basis for analysis, which in turn enables effective tuning for optimization and troubleshooting. Because the Enterprise Service Wrapper is deployed as a container, the same type of monitoring used for the microservices can be used here. Metrics, logs, and traces can be collected and aggregated to describe when which microservice sends a request to a SOA service. This can be used, for example, to understand the frequency and content of service requests and responses from a microservice to a selected SOA service.

### C. Security

The communication between the specialized microservices and the enterprise service wrapper is encrypted using mTLS. The respective microservices and the Enterprise Service Wrapper authenticate each other by presenting TLS certificates. This procedure ensures confidentiality and authenticity between the microservices and the Enterprise Service Wrapper. The TLS certificates are issued by a certificate authority, which serves as a trust anchor between the communication partners. After successful authentication, the STS creates a security token for the microservice. The microservice must present this security token to the wrapper when making requests, and the wrapper must verify the token. It is also ensured that the token originates from a legitimate sender so that the wrapper only accepts requests from authentic microservices in the architecture. As the ESW forms the only interface between the MSA and the ESB, it ensures that no unauthorized calls are made from the MSA to the SOA. In addition to mTLS and JWT, other mechanisms are not used in the remaining microservices architecture due to the trade-off between security and additional effort.

## VI. Conclusion and Future Work

In addition to the three mentioned responsibilities, such as connectivity, logging, and security, further responsibilities of the Enterprise Service Wrapper are conceivable. Still, they are not considered mandatory by the technical reference architecture. The Enterprise Service Wrapper could cache the responses returned by SOA services. Depending on the specific SOA service data, it could be reused for subsequent microservice requests. In this way, expensive requests via the network, which the wrapper would otherwise have to trigger via the ESB, could be avoided.

When storing the responses, the wrapper would have to implement an expiry protocol, e.g., Least Recently Used (LRU), which it can use to determine the stored responses that should be evicted first. The Enterprise Service Wrapper could also offer connection pooling to enable reusing connections to the ESB. In such a way, the wrapper could reduce the effort required to set up connections. It could also implement patterns such as timeout, retry, or circuit breaker. These could be used to make the connection to the ESB more resilient.

In this paper, only a part of the implementation of T-RaMicsV has been presented. Other components, such as security, scalability, and monitoring, will be covered in future publications. In addition to approaches such as chaos engineering to test T-RaMicsV, methods for homomorphic encryption were also evaluated in context.

## References


[1] 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.

[2] 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: 02, 2025].

[3] 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: 02, 2025].

[4] A. Koschel, A. Hausotter, M. Lange, and P. Howeihe, "Consistency for Microservices - A Legacy Insurance Core Application Migration Example," in *SERVICE COMPUTATION 2019, The Eleventh International Conference on Advanced Service Computing*, Venice, Italy, 2019, [Online]. Available: https://thinkmind.org/index.php?view=article&articleid=service_computation_2019_1_10_18001. [retrieved: 02, 2025].

[5] 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: 02, 2025].

[6] 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: 02, 2025.

[7] 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: 02, 2025].

[8] C. Schulze, H. Meyer, A. Koschel, A. Hausotter, A. Link, and T. van Dorp, "A Technical Reference Architecture for Microservices-based Applications in the Insurance Industry," in *SERVICE COMPUTATION 2024, 15th Intl. Conf. on Advanced Service Computing*. IARIA, ThinkMind, 2024, pp. 1–6, Online. Available: https://www.thinkmind.org/library/SERVICE_COMPUTATION/SERVICE_COMPUTATION_2024/service_computation_2024_1_10_10009.html [retrieved: 02, 2025].

[9] A. Hausotter, Koschel, H. Meyer, and C. Schulze, "Applying a Technical Reference Architecture to Implement a Microservices-based Insurance Application," *Intl. Journal on Advances in Intelligent Systems*, vol. 17, no. 3 & 4 (in progess), 2024. [Online]. Available: https://www.iariajournals.org/software/

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

[11] M. Fowler and J. Lewis, “Microservices a definition of this new architectural term,” 2014, Online. Available: https://martinfowler.com/articles/microservices.html [retrieved: 02, 2025].

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

[13] 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: The Thirteenth International Conference on Advanced Service Computing*, 2021, pp. 5–9, Online. Available: https://www.thinkmind.org/articles/service_computation_2021_1_20_10002.pdf [retrieved: 02, 2025].

[14] The Linux Foundation - The Cloud Native Computing Foundation, “Cloud Native Computing Foundation (“CNCF”) Charter,” 2024, Online. Available:https://github.com/cncf/foundation/blob/main/charter.md [retrieved: 02, 2025].

[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: 02, 2025].

[16] D. A. Chappell, *Enterprise Service Bus*. O’Reilly, 2004.

[17] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional., 2003.

[18] 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.

[19] M. Gharbi, A. Koschel, A. Rausch, and G. Starke, *Basiswissen für Softwarearchitekten (Basic knowledge for software architects)*. dpunkt.verlag, 2023.

[20] 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.

[21] M. Stadler and U. Gail, *Die Kfz-Versicherung - Grundlagen und Praxis (The car insurance - basics and practice)*. Karlsruhe: VVW GmbH, 2015.

[22] 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.

[23] 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: 02, 2025].