Towards the Implementation of Workflows in a Microservices Architecture for Insurance Companies

The Coexistence of Orchestration and Choreography

Arne Koschel
Andreas Hausotter
Robin Buchta
Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hannover, Germany
Email: arne.koschel@hs-hannover.de

Christin Schulze
Pascal Niemann
Christopher Rust
Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hannover, Germany
Email: andreas.hausotter@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 true even for the more conventional insurance industry with its highly heterogeneous application landscape and sophisticated cross-domain business processes. Therefore, the question arises of how workflows can be implemented to grant the required flexibility and agility and, on the other hand, to exploit the potential of the MSA style. In this article, we present two different approaches – orchestration and choreography. Using an application scenario from the insurance domain, both concepts are discussed. We introduce a pattern that outlines the mapping of a workflow to a choreography.**

Keywords*—*Workflow; Orchestration; Choreography; Insurance Industry; Microservices Architecture; SOA.

I. INTRODUCTION

Multi-step business processes and business workflows 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].

Over time, 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.

Recently, the MSA style and cloud computing joined the field. Taking all those typical cornerstones from (over time grown) insurances into account, the ultimate goal of our currently ongoing research [4] is to develop a "Microservice Reference Architecture for Insurance Companies (RaMicsV)" jointly with partner companies from the insurance domain. Placed within our work on RaMicsV is the question: "how to implement (insurance) business workflows using potentially several logical parts from RaMicsV, especially including microservices"?

While traditionally, for example, in SOAs, such workflows are mainly implemented using orchestration [5], the MSA style favors the more decoupled choreography for this purpose [6] [7]. Since RaMicsV aims to address the *combined usage* of more traditional approaches and microservices, 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.

However, since only a few authors (see Section II) look at the *combination* of choreography and orchestration and especially do not take insurance domain specifics into account, this article contributes initial steps on this way. In particular, we contribute in the present article our ongoing work and intermediate results about:

- How to implement insurance company processes through workflows within a MSA style utilizing an application scenario.
- Mapping processes for distribution of (micro)services.
- Types and a discussion of pros and cons for workflow implementations, including:
  - Orchestration, which controls the workflow and explicitly maps the workflow;
  - Choreography, which maps the workflow implicitly and places responsibility and control into the services;
  - Technical means: For example, implementing an orchestration based on a BPMN [8] model is relatively

 ISBN: 978-1-61208-947-8

straightforward – but how is this realized with a choreography?

The remainder of this article is structured as follows: After discussing related work in Section II, we place our current work into our initial logical reference architecture from [4] in Section III. Next, Section IV shows core definitions of orchestration and choreography. Section V provides an application scenario (car insurance coverage) and compares orchestration and choreography. Resulting from this, we discovered some typical mapping patterns. As an intermediate result, Section VI presents one mapping pattern. Finally, Section VII summarizes our results and concludes with some outlook to future work.

II. Related Work

The basis of our research builds on renowned authors in the scope of microservices, such as the foundational work from Newman [7] as well as Fowler and Lewis [9]. Within the design of our reference architecture, we benefit from various microservices patterns, for example, as they are discussed by Krause [10] and Richardson [6].

Directly related work to ours comes from authors, which deal with workflows in combination with microservices. In this context, it was important that the authors approach the combination of orchestration and choreography and not only examine their opposites.

One of the authors, who use orchestration and choreography frequently, is Ruecker [11] [12]. He recommends both approaches when implementing workflows and evaluates the right balance. Another author who evaluates the combination of both approaches is Chen [13]. He deals with the use and distinction of the two approaches and distinguishes between different use cases of their usage.

However, both authors do not address the core definitions, preferring to combine the approaches with other patterns. For this reason, we have tried to approach a core definition and present it in this article. We aim to develop patterns for the implementation of *choreography using BPMN* in order to achieve a clear realization with precise implementation rules. A first pattern is presented in this article as well.

As a further contribution, implementing the approaches is put into practice using an example business process from the insurance industry. For this purpose, we have chosen car insurance, one of the core products for German insurers. The authors Stadler and Gail [14] provide the basics for the process. Car insurance is mandatory for every car in Germany. For this reason, it is considered particularly important for attracting new customers. The elaboration refers to the VAA [1] and describes in detail what car insurance is all about and more.

III. Service-based Reference Architecture for Insurance Companies

This section will present our logical Microservice Reference Architecture for Insurance Companies (RaMicsV) as initially started in [4].

RaMicsV defines the setting for the architecture and the design of a microservices-based application of our industry partners. The application's architecture is out of scope, 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 must be taken into account. Concerning this contribution, the most relevant are:

- Coexistence: Legacy applications, SOA, and microservices-based applications will be operated parallel for an extended transition period. This means that RaMicsV must provide approaches for integrating applications from different architecture paradigms.
- Business processes are critical elements in an insurance company's applications 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** contains the set of microservices to provide the 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, non-repudiation, accountability and, audibility.

Components communicate 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 contains a message broker (see Figure 1).

In the next section, we will have a detailed look at the *Business Processes* component.

IV. Orchestration and Choreography

This section will present the core definitions of orchestration and choreography. The focus will be on the functional

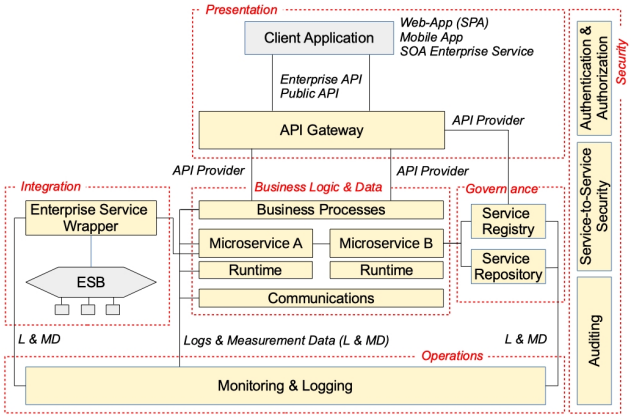

Figure 1. Building Blocks of the Logical Reference Architecture RaMicsV

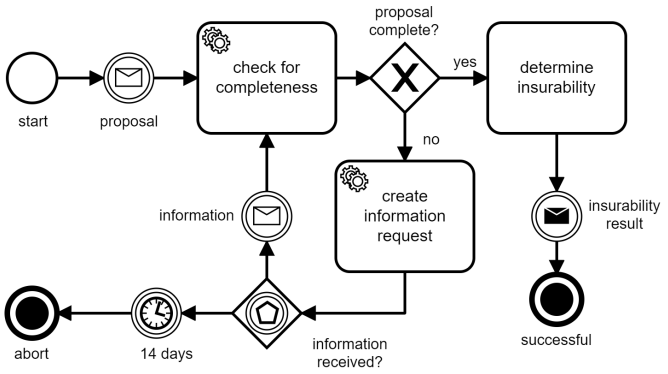

Figure 2. Insurance Application Process in BPMN

definitions, without reference to the insurance companies or a use case. Orchestration and choreography are often used in combination with other patterns. The coexistence of both is also possible [11]. The combination with other patterns and the coexistence of both will be evaluated in future work.

A. Orchestration

Orchestration is an often overloaded term, so Ruecker equates it with coordination [11], which captures the core definition well. Orchestration to implement a workflow simply describes the coordination of process steps. Such steps can include business services, technical services, or even user tasks [8]. Coordination is handled by a coordination unit or orchestrator. It is important to note that this is only a logical unit, i.e., it can be implemented in a distributed manner [15].

B. Choreography

Choreography follows a different approach. In contrast to orchestration, there exists no orchestrator [13]. Therefore, there is no explicit modeling and monitoring of a workflow. The workflow is implicitly mapped by the sequence of actions that the services perform. Consequently, the responsibility for adequately executing and processing the workflow lies with the services involved in a workflow [12].

Choreography is often combined with other patterns, for example, event-driven architecture [11]. Within our work here, we will focus only on the core definition. Consequently, we only look at the functional realization of a workflow with choreography, not (yet) focusing on technical details.

V. Application Scenario

This section discusses the creation of a sample process for our research. Its implementation is discussed based on orchestration as well as on choreography.

A. Process Creation and Development

We need a process at several points for our research within the RaMicsV context. As such, we have chosen a typical insurance product as an example, namely 'car insurance coverage.' Thus, we have developed an end-to-end process that extends from customer consultation to policy issuance. This process is based on [14] and has been evaluated through interviews with our project partners so that the theory developed is close to reality. As can be seen, this process contains only a subset of the available business process logic. This will be used as the basis for our research to evaluate the feasibility and identify challenges to build upon in future work. The process also refers to the (more generic) VAA use cases: review application, make application decision, obtain state information, provide contract relevant data, calculate base premium [1].

In various steps, we occasionally look at part aspects of the process so that the completeness is large enough to illustrate the concepts and small enough for it to be done efficiently.

The process starts with the receipt of a proposal from the customer and ends with sending the insurability result. This is visualized in Figure 2. Technically, the BPMN representation was created with Camunda Modeler [16].

After a proposal is received, it is reviewed for completeness. A case distinction follows. If the proposal is not complete, a complex event is triggered, which starts a 14-day countdown and waits for the missing documents to be received from the customer. The process instance terminates if the documents are not submitted within the time limit. The other path, used if the documents are complete, determines insurability.

B. Evaluating Implementation Strategies for an Insurance Application

The aforementioned process was used to evaluate the two implementation strategies (within the MSA style), and the strategies were applied separately. Relatively straight forward, we implemented the process using orchestration. However, only initial mapping and implementation approaches could be formulated for choreography, which somewhat opened a new field of (applied) research.

 ISBN: 978-1-61208-947-8

1) Implementation with Orchestration: As a start, our example process was modeled in Camunda Modeler [16] as a BPMN process, with services being implemented as external tasks. The communication happens asynchronously by listening to specific topics, which the services or the orchestrator sends. The services are created in JavaScript as mocks and contain only enough logic to trigger the next step.

The implementation consists of the process creation and the instantiation of individual services with an orchestrator. Camunda handles process flow and control.

2) Problems with Implementing a Choreography: Compared to orchestration, the choreography has no orchestrator that realizes elements such as BPMN decisions. In the introduced ”Insurance Application” process (see Figure 2), the decision ”proposal complete?” would not be taken by the orchestrator. This responsibility lies within the services that perform actions in the workflow. The decision must be taken *implicitly* by the choreography into the steps of the workflow.

There is a BPMN choreography notation called BPMN 2.0 Choreography, but it is insufficient for implementing tasks and interactions. It demonstrates the message traffic between two partitions and illustrates them complementing the BPMN [17]. So far, the notation only visualizes the choreography. Accordingly, there are no clear realization and implementation rules yet. These rules or patterns could better clarify which aspects and elements of the BPMN are to be converted and would help automate the transformation. The BPMN elements must be mapped to the choreography to realize workflows. For this reason, we decided to develop patterns to map the choreography better.

We decided to map a BPMN to choreography because it is an operational requirement of our project partners in the insurance domain. Other modeling types, e.g., a UML sequence diagram, could also realize and represent a choreography.

VI. Choreography Pattern

The first pattern we developed to map the BPMN elements to the choreography is called ”Any problem becomes a service” (see Figure 3). At the core of this pattern, each BPMN element that an orchestrator would adopt, becomes its own new service. In this case, the new services are technical services (colored in Figure 3) that do not process any tasks of the actual workflow but simply support it. Figure 3 does not show a traditional BPMN. It is simply intended to visualize where which technical services support the workflow.

In the ”Insurance Application” process, a new technical service named ”decision/result check of completeness” would take the result of the completeness check and trigger the next service of the workflow based on the result. Other elements, such as checking whether information has arrived or the time limit expired, are also mapped to new technical services.

This first pattern is an ad hoc solution, which can be used for a quick and simple mapping to the choreography. Its core drawback is that it might result in many services

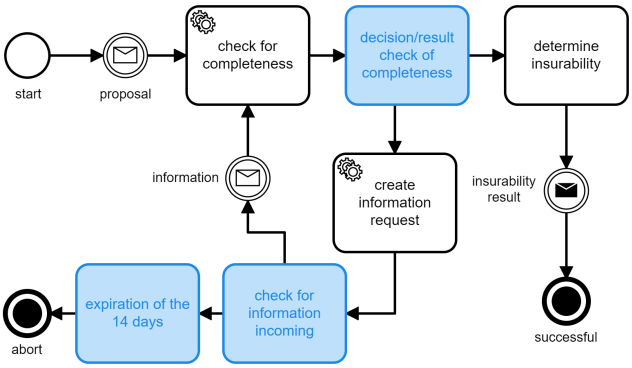

Figure 3. Visualization of the ”Any Problem Becomes a Service”–Pattern

being added to the workflow, increasing complexity easily. However, the technical services within this pattern are not business capability services and are not included as part of the service development. They simply provide a way to map an existing BPMN to a choreography.

Future work of us will present and evaluate additional patterns. The use of ”smart infrastructure” or the combination of orchestration and choreography by ”small orchestrators” may represent possible patterns.

VII. Conclusion and Future Work

Orchestration and choreography can map and implement workflows, such as workflow-based business processes, within software development that follows the MSA style. Several approaches are used to implement workflows. In principle, orchestration forms monitoring of the workflow, and choreography relies on the ownership of the services. Too much choreography might easily result in chaos. In contrast, too much orchestration might lead to a monolithic system. Applying both approaches can be a suitable solution for implementing workflows based on this assessment.

In future work, we will evaluate which approach should be used in an MSA style and which advantages or disadvantages they have. Our objective is to create a criteria catalog of when which approach or combination is to be preferred. Moreover, further Choreography Patterns will be presented and evaluated. We aim to define these patterns so that realizing the ”Insurance Application” process using the most suitable form of choreography becomes feasible. As already mentioned in subsection V-A, the process only covers a part of the business process logic, so the scope of the elements will be enlarged in future work. Among other things, compensations, more gateways, and different event types are considered.

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: 03, 2022].

 ISBN: 978-1-61208-947-8

[3] Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) - Federal Financial Supervisory (BaFin), "Versicherungsaufsichtliche Anforderungen an die IT (VAIT) (Insurance Supervisory Requirements for IT (VAIT)) vom 03.03.2022," 2022, Online. Available: https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Meldung/2022/meldung_2022_03_03_Aktualisierung_VAIT.html [retrieved: 03, 2022].

[4] 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: 03, 2022].

[5] 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.

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

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

[8] 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: 03, 2022].

[9] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term," 2014, Online. Available: https://martinfowler.com/articles/microservices.html[retrieved: 03, 2022].

[10] L. Krause, *Microservices: Patterns and Applications: Designing fine-grained services by applying patterns*. Lucas Krause, 2015.

[11] B. Ruecker, *Practical Process Automation - Orchestration and Integration in Microservices and Cloud Native Architectures*. O'Reilly, 2021.

[12] B. Ruecker, "The Microservices Workflow Automation Cheat Sheet," Online. Available: https://blog.bernd-ruecker.com/the-microservice-workflow-automation-cheat-sheet-fc0a80dc25aa [retrieved: 03, 2022].

[13] C. Chen, "Choreography vs orchestration," Online. Available: https://medium.com/ingeniouslysimple/choreography-vs-orchestration-a6f21cfaccae [retrieved: 03, 2022].

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

[15] G. B. Chafle, S. Chandra, V. Mann, and M. G. Nanda, "Decentralized orchestration of composite web services," in *Proc. 13th Intl. World Wide Web Conf. on Alternate Track Papers & Posters*. NY, USA: Association for Computing Machinery, 2004, p. 134–143.

[16] "Workflow and decision automation platform," Nov 2021, Online. Available: https://camunda.com/ [retrieved: 03, 2022].

[17] J. Ladleif and A. von Weltzien, "chor-js – an editor for bpmn choreography diagrams," Online. Available: https://camunda.com/blog/2021/01/chor-js-an-editor-for-bpmn-choreography-diagrams/ [retrieved: 03, 2022].

 ISBN: 978-1-61208-947-8