Competence Center
Information Technology and Management
Institute at the University of Applied Sciences
and Arts Hannover | cc_itm@hs-hannover.de

**HOCHSCHULE**
**HANNOVER**
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
*Fakultät IV*
*Wirtschaft und*
*Informatik*

Service Computation 2022
April 24 – 28, 2022 – Barcelona, Spain

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

*The Coexistence of Orchestration and Choreography*

A. Koschel, A. Hausotter, R. Buchta, C. Schulze, P. Niemann, C. Rust

Faculty of Business and Computer Science
Hannover University of Applied Sciences and Arts
Ricklinger Stadtweg 120 30459 Hannover

{arne.koschel | andreas.hausotter}@hs-hannover.de

# Presenter

Dr. ANDREAS HAUSOTTER is a professor emeritus for distributed information systems and database systems at the Hannover University for Applied Sciences and Arts, Hanover, Germany, Faculty of Business and Computer Science. His area of specialization comprises service computing – including service-oriented Architectures (SOA) and microservices – Java EE, webservices, distributed information systems, business process management, business rules management, and information modeling.

In 1979 he received his PhD in mathematics at Kiel University, Faculty of Mathematics and Natural Sciences. After graduation he started his career with KRUPP ATLAS ELEKTRONIK, Bremen, as a systems analyst and systems programmer in the area of real time processing. In 1984 he was hired as systems engineer and group manager SNA Communications for NIXDORF COMPUTER, Paderborn. After that, he worked for HAAS CONSULT, Hanover, as a systems engineer and product manager for traffic guidance systems.

In 1996 he was appointed professor of operating systems, networking and database systems at the Hannover University of Applied Sciences and Arts. He has been retired since March 2018.

From the beginning he was involved in several research projects in cooperation with industry partners. During his research semester he developed a Java EE / EJB application framework. Based on this framework a web-based simulation software for securities trading was implemented by his research group to train the apprentices of the industry partner.

In 2005, the Competence Center IT & Management (CC_ITM) was founded in cooperation with industry partners. Different ambitious research projects have since then been carried out in the context of service-computing, microservices, cloud computing, business process management, and business rules management.

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# CC_ITM@HsH

- **Competence Center Information Technology & Management (CC_ITM)**
  - Institute at the Hannover University of Applied Sciences and Arts;
  - Founded in 2005 by colleagues from the departments of **Business Information Systems and Computer Science;**
  - Members: **Faculty staff and industry partners** (practitioners) of different areas of businesses.
- Main objective
  - **Knowledge transfer** between university and industry.
- Research topics
  - Management of information processing;
  - Service computing, including Microservices, Service-oriented Architectures (SOA), Business Process / Rules Management (BPM/BRM);
  - Cloud Computing.

# Introduction

The ultimate goal of our current research is to develop a ’**Microservices Reference Architecture for Insurance Companies**’ jointly with our partner companies.

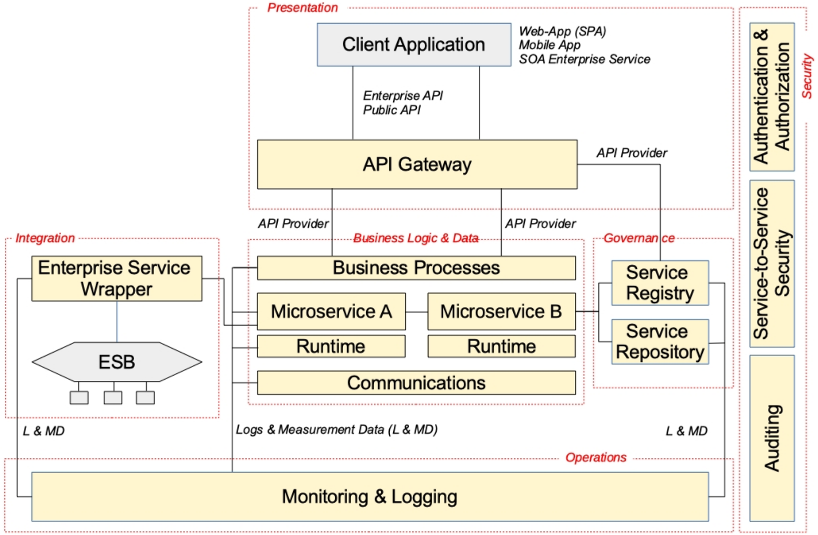
The topic of our current research is **the implementation of workflows in a Microservices Architecture** with the consideration of orchestration and choreography.

Questions to be answered:

- How to implement insurance company processes through workflows within a Microservices Architecture style?
- What are the implementation strategies – difference between orchestration and choreography?
- Selecting the right implementation strategy – how to realize processes with orchestration or choreography?
- Evaluation of these strategies – what issues can arise when using these approaches?

# Reference Architecture for Microservices

- **Coexistence**: Legacy applications, SOA and MSA based applications will be operated in parallel for a longer transition period.
- **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.



Building Blocks of the Logical Reference Architecture RaMicsV [own representation].

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# Orchestration and Choreography

- Evaluation of **core definitions** without other patterns.

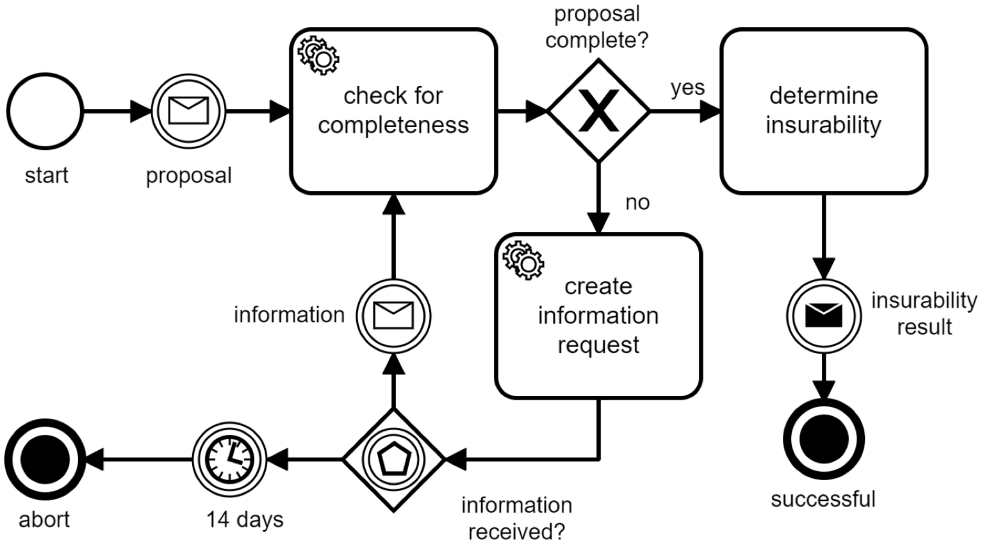
| Orchestration | Choreography |
|---|---|
| - **Coordination** of process steps;<br>- Handled by a (logical) coordination unit or orchestrator;<br>- Distributed implementation possible. | - No orchestrator;<br>- **No explicit modeling** or monitoring of a workflow;<br>- Mapped by **the sequence of actions;**<br>- Responsibility within the services. |

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# Process Creation and Development

- ’**Car insurance coverage process**’ – typical process of the core business;
- Project partners **use BPMN** to describe their processes.
- Large enough to **show and evaluate** orchestration and choreography;
- Only a **subset** of the existing **business process logic.**



Typical car insurance coverage process in BPMN [own representation].

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

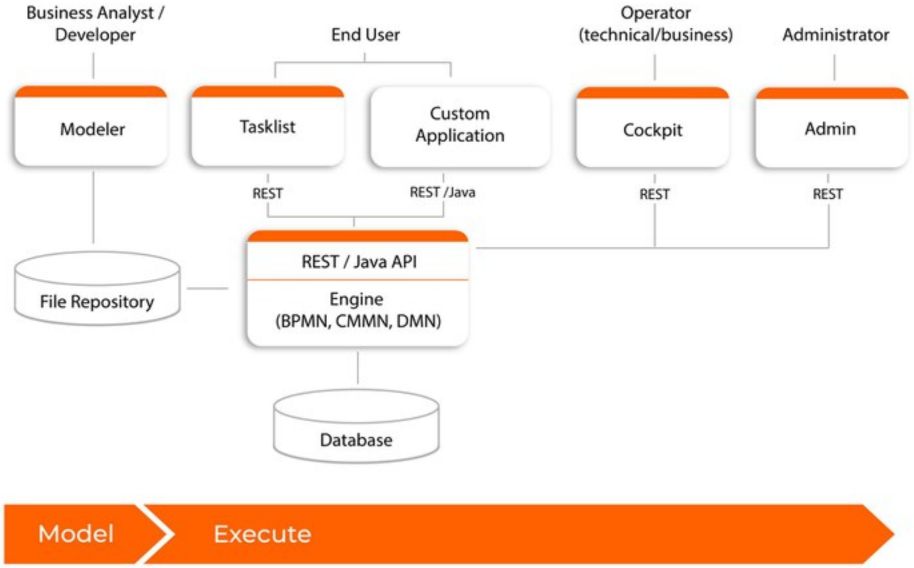
# Implementation with Orchestration

- Modeled in **Camunda Modeler** as a BPMN process;
- Services **as external tasks;**

- Asynchronous communication by **listening to specific topics;**
- Services are created in JavaScript, **only enough logic to trigger the next step;**

- **Camunda handles process flow and control.**

- **Implementation** consists of the **process creation and the instantiation** of individual services with an orchestrator.



Camunda Architecture [18].

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# Problems with Implementing a Choreography

- **No orchestrator that realizes elements** such as BPMN decisions;
- Responsibility lies within the services;
- **Decisions must be taken implicitly** by the choreography;
- **BPMN 2.0 choreography** demonstrates the **message traffic** and is complementing the BPMN;
- **No clear realization and implementation rules**, to map **a BPMN process to a choreography** yet.

# Problems with Implementing a Choreography

- Clarify, **which aspects and elements** of the BPMN are to be **converted;**
- Try to **automate the transformation;**
- **Developing patterns to map the choreography.**

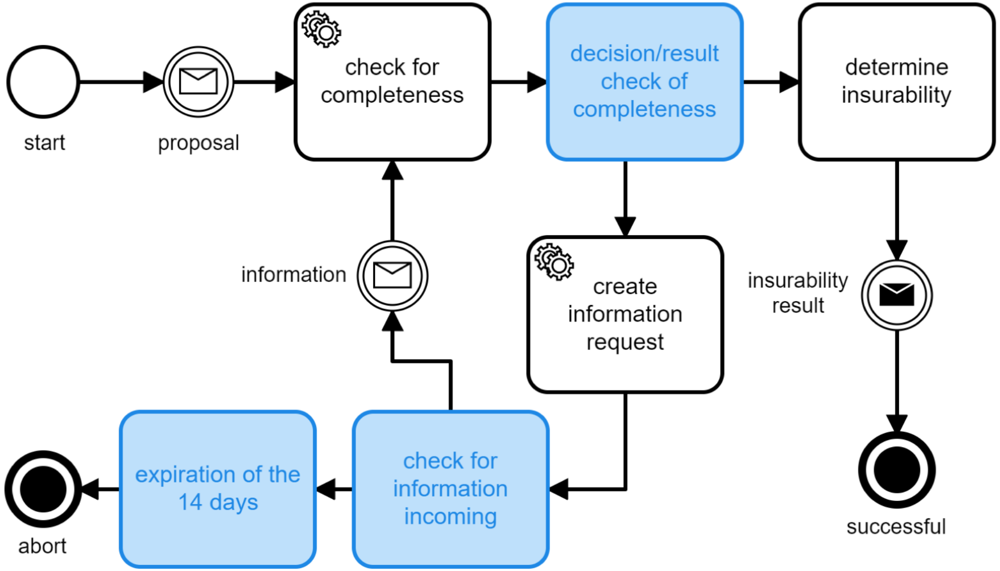
- **Requirement** of our project partners: **map BPMN to choreography**;
- Other **modeling types**, e.g., a UML sequence diagram, could also r**ealize and represent a choreography.**

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# Choreography Pattern

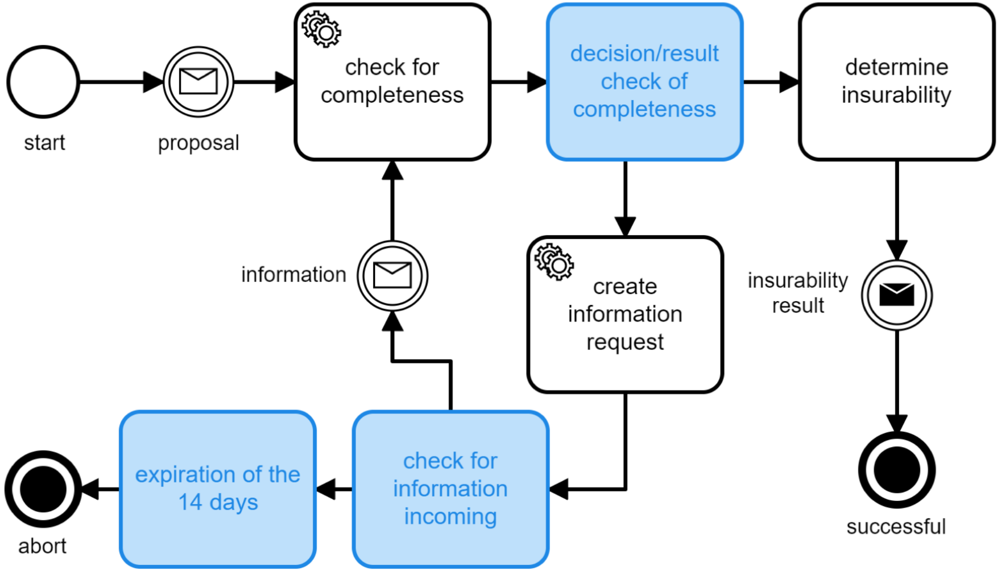
- First pattern: **‘Any Problem Becomes a Service’** – ad hoc solution;
- Each **BPMN element** that an orchestrator would adopt, becomes its **own new service;**
- New services are **technical services** (colored in the figure) that **simply support**.


start
proposal
check for completeness
decision/result check of completeness
determine insurability
information
create information request
insurability result
abort
expiration of the 14 days
check for information incoming
successful

Visualization – Choreography Pattern – ‘Any Problem Becomes a Service’ [own representation].

# Choreography Pattern

- **Quick and simple mapping** to the choreography;
- **Many services are added** to the workflow;
- Technical services are **not based on business capabilities**.


start
proposal
check for completeness
decision/result check of completeness
determine insurability
information
create information request
insurability result
successful
abort
expiration of the 14 days
check for information incoming

Visualization – Choreography Pattern – Any problem becomes a service [own representation].

# Agenda

1. Introduction
2. Orchestration and Choreography
3. Process Creation and Development
4. Implementation with Orchestration
5. Problems with Implementing a Choreography
6. Choreography Pattern
7. Conclusion and Future Work

# Conclusion and Future Work

- In this contribution, we have presented two strategies - orchestration and choreography.
- Implementing an orchestration using a BPMN is straight-forward.
- Implementing a choreography using a BPMN throws up some questions and problems - the presented pattern shows a first possible approach for mapping.

- The next steps in our research:
  - Evaluate the advantages and disadvantages of both strategies and determine when to use one of them, or both together;
  - Development and evaluation of further choreography patterns;
  - Investigation of BPMN elements not considered so far - like events or gateways;
  - And Implementation of the process with choreography.

# Thank you for your attention!

# 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].

[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.

# References

[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 finegrained 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.

# References

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

[13] C. Chen, “Choreography vs orchestration,” Online. Available: https://medium.com/ingeniouslysimple/choreography-vs-orchestrationa6f21cfaccae [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].

[18] Camunda Architecture. Online. Available: https://quintagroup.com/blog/blog-images/camunda-bpmn-example-of-python-based-application/camunda-architecture-min.jpg/view [retrieved: 03, 2022].