SERVICE COMPUTATION 2021

The Thirteenth International Conferences on Advanced Service Computing

ISBN: 978-1-61208-844-0

April 18 - 22, 2021

SERVICE COMPUTATION 2021 Editors

Andreas Hausotter, Hochschule Hannover - University of Applied Sciences and Arts, Germany

SERVICE COMPUTATION 2021

Forward

The Thirteenth International Conferences on Advanced Service Computing (SERVICE COMPUTATION 2021), held on April 18 - 22, 2021, 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.

The conference had the following tracks:

- Service innovation, evaluation and delivery
- Service quality
- Challenges
- Advanced Analysis of Service Compositions

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 2021 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 2021. 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 2021 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope SERVICE COMPUTATION 2021 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.

SERVICE COMPUTATION 2021 Steering Committee

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

SERVICE COMPUTATION 2021 Publicity Chair

Jose Luis García, Universitat Politecnica de Valencia, Spain
Lorena Parra, Universitat Politecnica de Valencia, Spain

SERVICE COMPUTATION 2021 Industry/Research Advisory Committee

Steffen Fries, Siemens Corporate Technology - Munich, Germany
Matthias Olzmann, noventum consulting GmbH - Münster, Germany
Rong N. Chang, IBM T.J. Watson Research Center, USA
Jan Porekar, SETCCE, Slovenia
Bernhard Hollunder, Hochschule Furtwangen University - Furtwangen, Germany

SERVICE COMPUTATION 2021

Committee

SERVICE COMPUTATION 2021 Steering Committee

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

SERVICE COMPUTATION 2021 Industry/Research Advisory Committee

Steffen Fries, Siemens Corporate Technology - Munich, Germany
Matthias Olzmann, noventum consulting GmbH - Münster, Germany
Rong N. Chang, IBM T.J. Watson Research Center, USA
Jan Porekar, SETCCE, Slovenia
Bernhard Hollunder, Hochschule Furtwangen University - Furtwangen, Germany

SERVICE COMPUTATION 2021 Publicity Chairs

Jose Luis García, Universitat Politecnica de Valencia, Spain
Lorena Parra, Universitat Politecnica de Valencia, Spain

SERVICE COMPUTATION 2021 Technical Program Committee

Jocelyn Aubert, Luxembourg Institute of Science and Technology (LIST), Luxembourg
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Eugen Borcoci, University "Politehnica" of Bucharest, Romania
Uwe Breitenbücher, University of Stuttgart, Germany
Antonio Brogi, University of Pisa, Italy
Isaac Caicedo-Castro, Universidad de Córdoba, Colombia
Rong N. Chang, IBM T.J. Watson Research Center, USA
Dickson Chiu, The University of Hong Kong, Hong Kong
Leandro Dias da Silva, Universidade Federal de Alagoas, Brazil
Erdogan Dogdu, Angelo State University, USA
Monica Dragoicea, University Politehnica of Bucharest, Romania
Sebastian Floerecke, University of Passau, Germany
Sören Frey, Daimler TSS GmbH, Germany
Steffen Fries, Siemens Corporate Technology - Munich, Germany
Somchart Fugkeaw, Sirindhorn International Institute of Technology | Thammasat University, Thailand
Katja Gilly, Miguel Hernandez University, Spain
Victor Govindaswamy, Concordia University - Chicago, USA
Maki Habib, The American University in Cairo, Egypt

Copyright Information

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

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

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

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

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

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

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

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

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

Table of Contents

Towards a Resiliency Decision Framework for Microservices

Johannes Busch
Faculty IV, Dept. of Computer Science
Univ. of Appl. Sciences&Arts Hannover
Hannover, Germany

Andreas Hausotter
Faculty IV, Dept. of Computer Science
Univ. of Appl. Sciences&Arts Hannover
Hannover, Germany
Email: Andreas.Hausotter@hs-hannover.de

Arne Koschel
Faculty IV, Dept. of Computer Science
Univ. of Appl. Sciences&Arts Hannover
Hannover, Germany
Email: Arne.Koschel@hs-hannover.de

***Abstract*—Microservices build a deeply distributed system. Although this offers significant flexibility for development teams and helps to find solutions for scalability or security questions, it also intensifies the drawbacks of a distributed system. This article offers a decision framework, which helps to increase the resiliency of microservices. A metamodel is used to represent services, resiliency patterns, and quality attributes. Furthermore, the general idea for a suggestion procedure is outlined.**

Keywords*—*Microservice; Resiliency; Software Architecture.

I. Introduction

Microservices are a current trend in software development. They divide complex systems into several independent and lightweight services [1]. This approach has several benefits over traditional architectures like monoliths or Service-Oriented Architectures (SOA). One major point is the independence of these services at runtime and in development. Furthermore scalability, security or equivalent questions can be answered on a per service basis, which offers a higher degree of flexibility [2][3].

Besides these benefits, microservices also come with various challenges, one important of them being their distributed nature. To compute non-trivial business functions, several microservices have to work together. Thus, they have to communicate over networks, which cannot guarantee complete availability. Modern cloud based systems further increase this problem by their volatile nature.

Key contribution of this article is a decision framework, which offers suggestions to increase the resiliency of specific microservices. To achieve this goal services and resiliency patterns need to be put in perspective. This is done through the resiliency decision framework metamodel, a novel way to describe services, patterns, quality attributes and their interconnections. Furthermore, a suggestion procedure will be presented. Its purpose is to analyze service requirements and pattern effects to compute a list of suggestions to strengthen these requirements.

In this article resiliency is defined as the ability of a software to handle failures in a meaningful way or recover completely from it without human intervention. The context for resiliency in this article is based upon safety and not security. The definition is based upon [4][5].

This work was developed under the *Competence Center Information Technology and Management* (CC_ITM), which is part of the *University of Applied Sciences and Arts Hannover*. Its main objective is the transfer of knowledge between university and the insurance industry.

The remainder of this article is organized as follows: First related work is presented in Section II. An application scenario based on the german insurance sector is presented in Section III. The core of the Resiliency Decision Framework is described in Section IV. Section V summarizes this article and gives an outlook on future work.

II. Related Work

While resiliency is important in microservice based architectures, it is certainly not a new topic in general. In Service-Oriented Architectures resiliency is often realized by means of fault tolerant services. These can be designed either through specific middleware [6][7] or alternative implementations [8]. Furthermore, fault tolerance can be evaluated over several services [9], which can increase the resiliency of complex business functions or processes.

Another field besides fault tolerance are self healing systems. Self healing can be achieved either through internal techniques [10] or external components [11]. Furthermore, self healing can be build into the architecture itself [12].

In [13], a catalogue of resiliency patterns is described. Furthermore, a framework for resiliency in high performance computing is defined. In contrast we provide a full resiliency decision framework for microservices, which is a novelty to the best of our knowledge.

III. Application Scenario

To stress the importance of resiliency in microservice-based systems and to evaluate the suggestions given by the Resiliency Decision Framework following application scenario will be used. It is based on prior CC_ITM [14] work. The described `Partner Management System` is expanded further by a business process from the reference architecture for the German insurance companies (VAA) [15].

An overview of the application scenario is given in Figure 1. It consists of several microservices grouped into different systems. Each system represents a bounded context as described

 ISBN: 978-1-61208-844-0

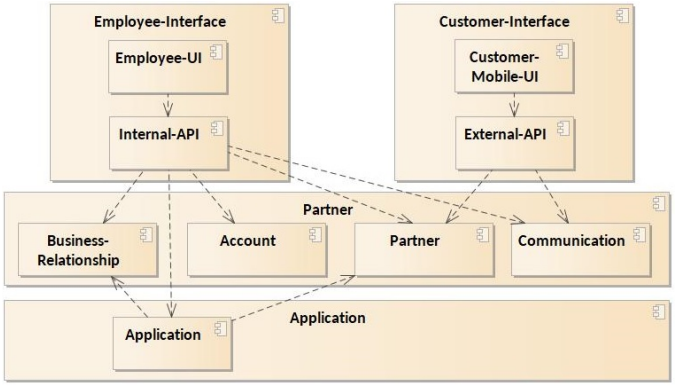

Figure 1. Overview of the application scenario

by Evans in [16]. The goal was to develop a realistic and complex enough system, which could be found at partner companies of the CC_ITM.

The `Partner Management System` offers several CRUD operations to manage insurance partners. It is used throughout the scenario and represents the core of the scenario. The `Application System` implements the business process `Manage Application` ('Antrag verwalten') from the VAA. It processes applications for a new insurance contract. Thus it retrieves information about partners and adds a new insurance contract, if an application finishes successful. The `Employee-Interface System` contains a UI and API for internal use. It offers a comprehensive access to all insurance systems and is the sole way to work with the implemented business process. Customers and external companies can access insurance information through the `Customer-Interface System`. Both UIs are designed as modern Single Page Applications.

IV. Resiliency Decision Framework

The goal of the `Resiliency Decision Framework` (RDF) is to suggest resiliency design patterns based upon the quality requirements of the analyzed microservices. To achieve this goal the different patterns, services and quality requirements need to be presented in a well defined way. This is done through a metamodel. To evaluate these information and provide a list of suggestions a suggestion procedure is described.

A. Resiliency Decision Framework Metamodel

An overview of the `Resiliency Decision Framework Metamodel` is given in Figure 2. Base is the `Framework` element, which is the root for a given RDF instance. The metamodel consists of three major parts:

`Quality Tree`: Services and patterns are described through a quality attributes defined in a quality tree. The `QualityTree` element offers, for example, to define a resilient specific one. The metamodel structure is based on well know quality trees like ISO 25010 [17].

`Services`: The `Service` element describes a specific microservice. Each service has several characteristics, which

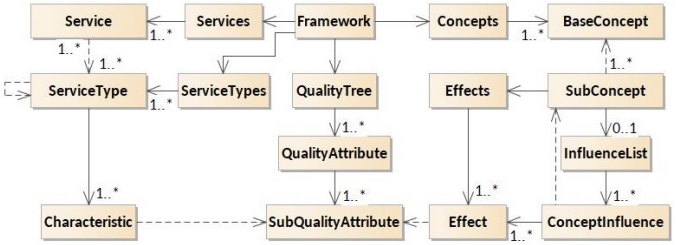

Figure 2. Overview of the developed metamodel

correspond to a specific requirement described by an quality attribute. Each characteristic also contains a numeric value, representing its importance. Currently these have to be defined by the architect. `ServiceType` elements are used to organize these characteristics and enable reusability in complex systems.

`Concepts`: This element defines resilient patterns in a consistent way. Each pattern consists of a base concept and one or more sub concepts. The base concept describes a general problem and solution (for example, Monitoring). A Sub concept narrows this to a specific pattern (for example, Logging) with its effects on service quality and requirements. The effect is described through a numeric value. Currently these are given by the authors based upon research into the corresponding patterns. Furthermore, influences amongst patterns can be defined in order to describe how well certain patterns work together.

B. Resiliency Decision Framework Suggestion Procedure

The `Suggestion Procedure` analyzes a defined model and computes a list of resiliency pattern suggestions for each defined service. It consists of several steps to determine and enhance the list of suggestions for a given service. Especially the filter steps are highly configurable. All steps are based on the numeric values given to pattern effects and service characteristics. Currently these represent qualitative values and not quantitative ones based on concrete measurements. The different steps are:

1) `Collect requirements`: Collect the requirements for a service by collecting all service types.
2) `Determine positive patterns`: Determine all resiliency patterns, which have a positive effect (for example, a corresponding positive numeric value) on at least one of the service requirements.
3) `Domain filter`: Filter these patterns by domain rules, for example, that would cause a negative impact onto the service requirements.
4) `Threshold filter`: Additionally filter these patterns by thresholds, for example, the combined negative effects of a pattern. A simple approach to combine these effects is to add corresponding numeric values but more complex approaches (for example, weighing) are also possible.
5) `Sorting`: Evaluate each pattern against the service requirements and sort the pattern list based on this eval-

 ISBN: 978-1-61208-844-0

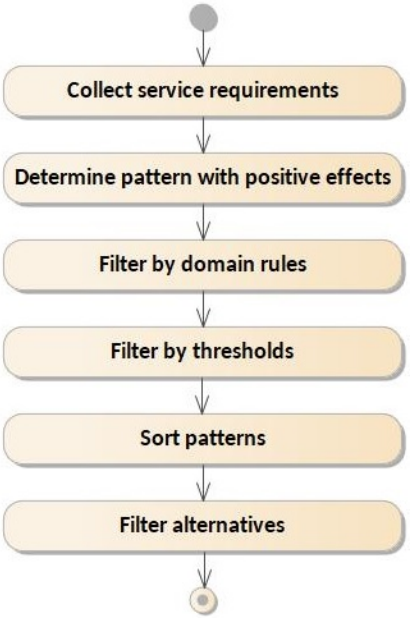

Figure 3. Overview of the developed procedure

uation. A simple approach for evaluation is to combine the positive and negative effect values in correspondence to the service requirements.

6) `Filter alternatives`: Filter less fitting pattern alternatives to diversify the list of suggestions, for example, filter less fitting approaches to load balancing.

These steps will be repeated for each service defined in the Resiliency Decision Framework model. An overview of the procedure is given in Figure 3.

C. Applying the Resiliency Decision Framework

To apply the Resiliency Decision Framework to a microservice based system, several activities have to be performed. Besides configuring the procedure filters, a comprehensive pattern catalogue has to be developed. This was done by a comprehensive literature analysis into software patterns. Furthermore, the different microservices have to be defined in the framework model.

After applying the procedure (cf. Figure 3) to this model, the last activities are implementing the suggestions and evaluating the improved microservices. These activities especially have to be repeated each time the catalogue of patterns or the service requirements change. All these activities form the Resiliency Decision Framework process.

How the microservices are evaluated, is out of scope of this contribution. One approach could be the QoS Measurement Model described in [18].

D. Resiliency Decision Framework Example

As described above, the basis for all suggestions is the definition of a service. For example, the partner service needs to be highly available because of its central role. Also it has to offer minimal latencies for user interfaces and external partner companies. Resiliency patterns are analyzed by the framework procedure based on these service `Characteristics`. A possible list of patterns could include automatic scaling, escalation and monitoring. These are ordered by their positive impact onto the service requirements. Besides this, all effects of required patterns and pattern influences are also analyzed.

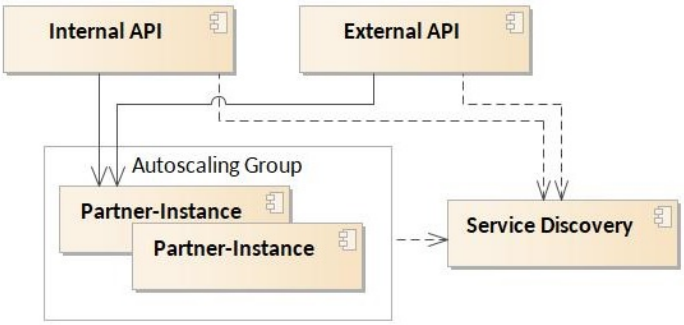

Figure 4. RDF applied onto the Partner service

`Automatic scaling` increases the availability of the service, but has a list of requirements. To distribute the requests and support scaling, a load balancing pattern is needed. Several load balancing patterns are known, thus the suggestions procedure evaluates the different alternatives. An external load balancer would, for example, add another hop, thus load balancing based upon service discovery is suggested by the framework.

The enhanced `Partner-Service` is given in Figure 4. The partner service is now deployed through an autoscaling group. Requests to a partner service a distributed through a service discovery, which could implement different algorithms (for example, round robin). To minimize the impact on client services, the service discovery should implement DNS as its API. Thus, no changes to the API services are needed.

V. Conclusion and Future Work

The basis for the `Resiliency Decision Framework` was described in this article. By using a well-defined metamodel, the RDF can be applied to different sectors and areas. The `Suggestion Algorithm` uses the metamodel to create a list of diverse suggestions with maximum positive effect on the resiliency of a microservice.

Future work will revolve around the development of a quality tree for software resiliency. Furthermore, an extensive library of resiliency patterns will be developed. An evaluation will be done by applying the framework to the complete application scenario. The quality of the suggestions is directly dependent on the quality of the service requirements and pattern catalogue definitions. Thus, a way to evaluate the quality of service requirements and pattern definitions need to developed.

References

[1] M. Fowler and J. Lewis, “Microservices a definition of this new architectural term,” [retrieved 11, 2020]. [Online]. Available: https://martinfowler.com/articles/microservices.html

 ISBN: 978-1-61208-844-0

[2] E. Wolf, *Microservices: Grundlagen flexibler Softwarearchitekturen*. dpunkt.verlag, 2016.
[3] M. Richards, *Microservices vs. Service-Oriented Architecture*. O'Reilly Media Inc., 2016.
[4] U. Friedrichsen, "Unkaputtbar: Eine kurze Einführung in Resilient Software Design," *Business Technology Magazin*, vol. 19, 4 2014.
[5] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, V. Sekar, "Gremlin: Systematic resilience testing of microservices," in *Proc. 36th IEEE International Conference on Distributed Computing Systems*, 2016, pp. 57–66.
[6] Z. Zheng and M. R. Lyu, "A qos-aware middleware for fault tolerant web services," in *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, 2008, pp. 97–106.
[7] I. Chen, G. Ni, and C. Lin, "A service-oriented fault-tolerant environment for telecom operation support systems," in *2008 IEEE International Symposium on Service-Oriented System Engineering*, 2008, pp. 208–214.
[8] A. S. Nascimento, C. MF. Rubira, R. Burrows, F. Castor, and P. HS Brito, "Designing fault-tolerant soa based on design diversity," *Journal of Software Engineering Research and Development*, vol. 2, no. 1, p. 13, 2014.
[9] K. Peng and C. Huang, "Reliability evaluation of service-oriented architecture systems considering fault-tolerance designs," *Journal of Applied Mathematics*, vol. 2014, pp. 1–11, 01 2014.
[10] A. Carzaniga, A. Gorla, A. Mattavelli, and N. Perino, "A self-healing technique for java applications," in *Proc. ICSE '12: International Conference on Software Engineering*, 2012, pp. 1445–1446.
[11] . K. Ravi and V. Sathyanarayana, "Container based framework for self-healing software system," in *Proc. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004.*, 2004, pp. 306–310.
[12] Y. Qun, Y. Xian-chun, and X. Man-wu, "A framework for dynamic software architecture-based self-healing," in *Proc. 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2005, pp. 2968–2972 Vol. 3.
[13] S. Hukerikar and C. Engelmann, "Resilience design patterns: A structured approach to resilience at extreme scale," *Supercomputing Frontiers and Innovations*, vol. 4, 08 2017.
[14] A. Koschel, A. Hausotter, M. Lange, and S. Gottwald, "Keep it in sync! consistency approaches for microservices: An insurance case study," *Service Computation*, pp. 7–14, 10 2020.
[15] GDV, "Vaa-fachliches referenzmodell," [retrieved 07, 2017]. [Online]. Available: http://www.gdv-online.de/vaa
[16] E. Evans, *Domain Driven Design — Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.
[17] iso25000.com, "Iso/iec 25010," [retrieved 3, 2021]. [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=0
[18] A. Hausotter, A. Koschel, , J. Busch, and M. Zuch, "A generic measurement model for service-based systems," *Service Computation*, pp. 12–18, 2 2018.

 ISBN: 978-1-61208-844-0

Towards a Microservice Reference Architecture for Insurance Companies

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

Alexander Grunewald
Moritz Lange
Pascal Niemann
Hochschule Hannover
University of Applied Sciences & Arts Hannover
Faculty IV, Department of Computer Science
Hannover, Germany
Email: andreas.hausotter@hs-hannover.de

***Abstract*—Microservices are meanwhile an established software engineering vehicle, which more and more companies are examining and adopting for their development work. Naturally, reference architectures based on microservices come into mind as a valuable thing to utilize. Initial results for such architectures are published in generic and in domain-specific form. Missing to the best of our knowledge however, is a domain-specific reference architecture based on microservices, which takes into account specifics of the insurance industry domain. Jointly with partners from the German insurance industry, we take initial steps to fill this gap in the present article. Thus, we aim towards a microservices-based reference software architecture for (at least German) insurance companies. As the main results of this article we thus provide an initial such reference architecture together with a deeper look into two important parts of it.**

Keywords–Microservices; Insurance Industry; Reference Architecture; SOA co-existence

I. INTRODUCTION

A current 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 structure the system is known as the microservice architectural style [1]. A study from 2019 (see [2]) shows, the microservice architecture 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 Service-Oriented Architecture (SOA), Business Rules and Business Process Management (BRM/BPM), SOA-Quality of Service (SOA-QoS), and microservices (cf, [3]–[6]), between the *Competence Center Information Technology and Management (CC_ITM)* from the University of Applied Sciences and Arts Hanover and two regional, middle-sized German insurance companies. The ultimate goal of our current research is to develop a 'Microservice Reference Architecture for Insurance Companies' jointly with our partner companies. This shall allow to build microservice conformant insurance application systems or at least such system parts.

When developing a reference architecture for our partner companies, 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. 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 by means of microservices is fine, but a complete migration to the microservices architectural style is not a desired option.

On the one hand, there is a desire to raise the potential of the microservices approach and, on the other hand, to take into account requirements that result from the existing application landscape. This leads to several guidelines, respectively, questions to be answered by a reference architecture, such as, for example, 'Which information from business and technical services shall be provided for architects, developers, operators, etc.?', 'How to integrate with business processes – is service orchestration or choreography (or both) more suitable for microservices?', 'How to co-exist with the given SOA services and their Enterprise Service Bus (ESB)?', 'What about transactions and consistency?', 'Compliance aspects', and more. While initial research on reference architectures with microservices exists in general as well as in some domain-specific variations, we are not aware of such research for the insurance domain in particular.

In this article, we present our initial steps towards a microservices reference architecture for the insurance domain as mentioned above, that complies with the above-named cornerstones and guidelines. In particular, we present our initial logical reference architecture and more logical and technical details about two selected important components from it, namely logging and monitoring.

We organize the remainder of this article as follows: After discussing related work in Section II, we present our initial logical reference architecture in Section III. Afterwards, Section IV shows how details about the logging and monitoring parts of our reference architecture. Section V summarizes the results and draws a conclusion.

 ISBN: 978-1-61208-844-0

II. Related Work

The work related to our research falls into several categories. We will discuss these categories in sequence.

Publications of renowned authors in the area of microservices form the solid base of our research work. Worth mentioning are the basic works of Newman [7] and Fowler and Lewis [1]. The design of our reference architecture benefits from diverse microservices patterns as they are discussed by Krause [8] and Richardson [9].

The contribution of Angelov et al. [10] explains that a reference architecture is successful only if *context*, *objectives* and *design* can be brought into line. Our design refers to 'type 4', which amongst other things means that findings of the implementation of microservices-based application flow into the design of the target reference architecture. Here our previous work – a prototypic implementation of the *Partner Management System* – comes into play [11].

The closest relationship to our research has an article published by Yu et al. [12]. They present a microservices-based reference architecture in the context of *enterprise architecture*. However, this reference architecture aims to be applied to many organizations and is therefore rather generic, while our approach tries to meet the requirements of our industry partners from the insurance sector.

III. Reference Architecture for Microservices

In this section, we will present our logical reference architecture for microservices in the insurance industry (RaMicsV).

RaMicsV defines the setting for the architecture and the design of a microservices-based applications of our industry partners. The application's architecture itself 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 have to be taken in account. With respect to this contribution the most relevant are:

- Enterprise Service Bus (ESB): The ESB as part of the SOA must not be questioned.
- Coexistence: Legacy applications, SOA and microservices-based applications will be operated in parallel for a longer transition period. This means that RaMicsV has to provide approaches for the integration of applications from different architecture paradigms.
- Observability: To observe microservices-based as well as SOA and legacy applications in a comprehensive and consistent manner, a unified monitoring and logging approach has to be designed.

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 that are 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.

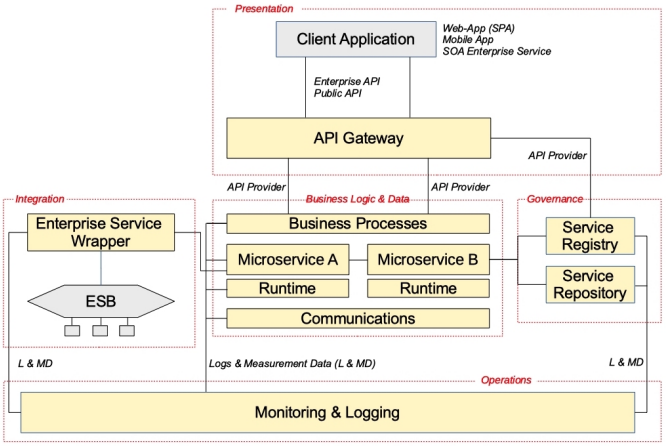

Figure 1. Building Blocks of the Logical Reference Architecture RaMicsV.

- **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 a unified monitoring and logging, which encloses all systems of the application landscape.

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

In the next section, we will have a detailed look at the *operations* responsibility area in detail.

IV. Logging and Monitoring

In this section we privide details about the logging and monitoring parts of our reference architecture, starting with fundamental concepts, followed by a logical and technical reference architecture.

A. Introduction to Logging and Monitoring

In order to provide production-ready software, it is not enough to fulfill only the functional requirements. Figure 2 shows a typical process that is followed when creating production-ready microservices. Observability is assigned to the final quality attributes along with configurability and security. Only if these components have been considered, is it possible to speak of production-ready software [9]. In the following we would like to focus on the aforementioned requirements for the reference architecture, specifically on observability. We are concerned with the objective of how we can create a uniform, fully comprehensive, traceable environment for monitoring and logging.

In his book *Release IT*, Michael T. Nygard does not call this observability but transparency [13]. We do not distinguish between the different terms used for this purpose, but focus on the activities behind the terms, i.e., logging of data and subsequent monitoring. Logging includes the automatic

 ISBN: 978-1-61208-844-0

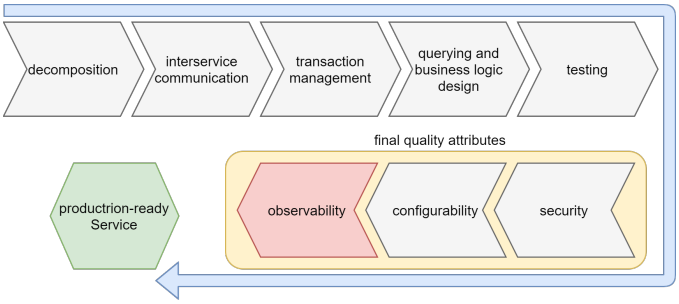

Figure 2. Typical building blocks for the development of a (micro)Service.

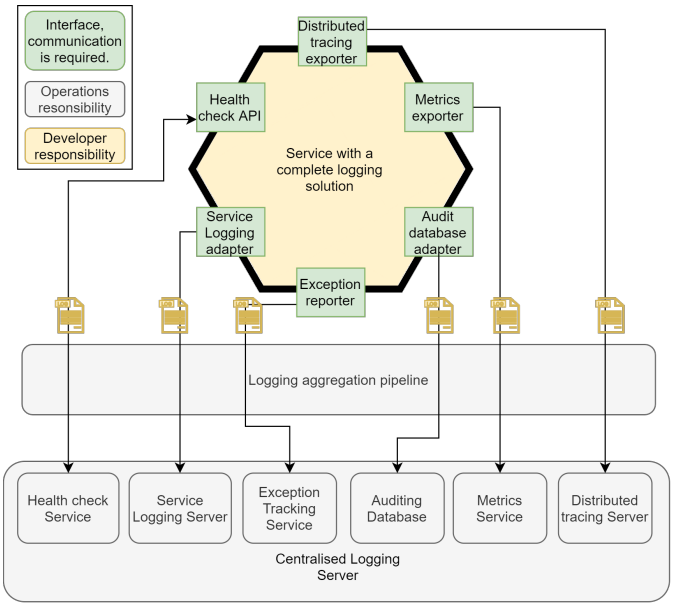

Figure 3. Exemplary implementation of all patterns in combination [9, adopted with modifications].

generation of messages. The generation is based on different triggers. The messages are sent to a location and collected there [14]. Monitoring includes the tools and processes that measure and manage the systems. Furthermore, it includes the process of extracting business value from the underlying data. This data is used to generate added value [15]. We will not go into the basics of monitoring and logging as this would exceed the scope. We will look at what is involved in a fully comprehensive logging of a service.

In the next subsection IV-B, patterns are presented that take account of the provision of data. The aim is to cover all areas of the service. It is important to note that we are focusing on the service and not on the environment in which it is located.

B. Patterns for Logging and Monitoring

In the following, the individual patterns that have been considered in Figure 3 are going to be explained. Figure 3 shows an example implementation of all patterns. In the Figure, the *log aggregation* is placed over all logs that are created so that when the pattern is implemented, all logs are considered.

We first consider the *health check API* pattern. The service receives an endpoint that provides information about its current health status. For example, Spring Boot Actuator can be used for this purpose, which automatically creates a health endpoint that can be adjusted if required [9]. This can be a simple ping for accessibility, but also a smoke test that ensures functionality. Figure 3 shows a bidirectional connection from the health check service to the corresponding API. This is because the endpoint must be queried, and the results obtained. The queries can take place periodically and/or before each invocation of the service. It is important to note that the health check service is a logical component and that requesting the endpoint and collecting the results may very well be two independent components [9].

Next, we are going to look at the *log aggregation* pattern. This is for aggregating all the logs of the multiple instances of a service, to be able to make themselves available together afterwards. This is important because you are interested in all the logs of the service and not just those of one instance. And if a particular instance is of importance, it should be found in the log entry [9].

In Figure 3, the log aggregation goes across all log entries, as the aggregation will refer to all logs regardless of the type. Here it becomes clear that the implementation of this pattern depends strongly on how the service is implemented. If there is only one instance, aggregation is not needed. The same applies to the implementation of the other endpoints. For example, if the audit logs are written directly to a database, no aggregation layer is needed. Again, this is only from a logical point of view.

The *distributed tracing* pattern is particularly important if control flows are of interest, and requests are being passed through multiple services. For this purpose, each request is given a unique ID and it is logged where and how long the request was in the individual services concerned.

The *application metrics* pattern is designed to collect metrics provided by the service. The developer is responsible for ensuring that valuable metrics can be collected, and the operator is responsible for managing them [9].

The *exception tracking* pattern considers the exceptions thrown by a service separately. Exceptions are also special to the service and require special attention. Here, the exceptions are duplicated and handled in detail if necessary. An alert function is optional. An attempt is made to prepare the information so that action can be taken as quickly as possible [9].

The last pattern we look at is the *audit logging* pattern. Here, all user actions are recorded. An audit log contains the identity of the user, the action taken, and the business objects involved [9].

Not all patterns can always be implemented in a meaningful way. In addition, the level of detail in which the individual patterns are implemented varies from application to application. Most of the time, a subset of the patterns presented is the right and sufficient choice to fully observe the service for the purpose it fulfills. If there are multiple instances of a service, log aggregation should be performed across all log types to evaluate the real behavior of the service. In the implementation of the patterns, there are sometimes clear responsibilities of the task areas, as can be seen in Figure 3. However, coordination at the interfaces is also unavoidable. The developers are responsible for creating decent log entries. The operators are responsible for what the users get to see. In

 ISBN: 978-1-61208-844-0

the end, it can be said that monitoring and logging is essentially very similar to that of distributed systems.

C. Logical reference architecture

A big building block of the logical reference architecture for microservices (RA4MicsV) is monitoring and logging to add an observability layer to the architecture. One of our main goals was to identify the logical components to implement logging and monitoring in a microservice environment, while maintaining the requirement of coexistence mentioned in Section III. To accomplish this it was important that the logging and monitoring concept for the microservices could be integrated as well as possible into the existing environment, which consists of a combination of monolithic systems and SOA. Figure 4 shows the components of the system itself and the identified, logical components needed to implement an extensive logging and monitoring infrastructure. The key components for the logging and monitoring in this figure will be explained in detail down below.

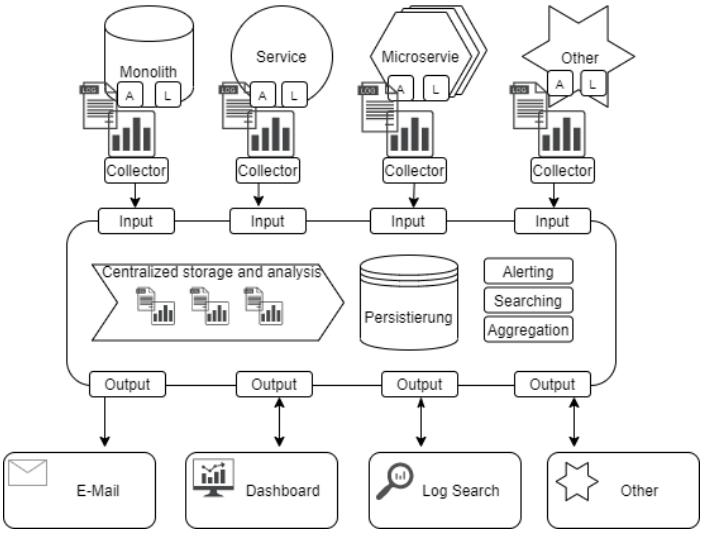

Figure 4. Logical reference architecture of the monitoring and logging environment

- **Agent (A):** Agents are some sort of external process or processes to instrument processes at runtime. There are two major methods agents use to instrument a service directly. The first is some external process or monitoring service that injects code into your service. The second method is through some sort of in-process agent that is imported to the runtime environment of a process and uses a system of user defined rules to trace specific actions [16].
- **Library (L):** Standardized libraries used in services that handle the key components for instrumentation and context propagation through a standardized API. Libraries can support a polyglot heterogeneous application by defining a relatively small API that supports the least-common set of features shared by all of the target languages [16].
- **Collector:** The functions of a collector varies from implementation to implementation, but in common cases the following functionalities are provided by an collector: Translate incoming data into another format for further processing, sampling and compute aggregate statistics about incoming data [16].
- **Centralized storage and analysis:** Responsible for gathering all of the telemetry data, storing it and analyzing it. As mentioned before, the functionality will vary widely based on the implementation [16].

Our model attempts a combination of white box and black box monitoring and logging, since the systems does not only consists of microservices. Apart from that, a whitebox model should generally be considered first when it comes to microservices [16].

D. Technical reference architecture

Since most monitoring and logging components vary in their functionality depending on the specific implementation chosen, this section deals with a sample implementation shown in Figure 5. The model is using a combination of the open source frameworks open telemetry for the instrumentation, elasticsearch as endpoint for the data and Kibana for visualization. The most important components will be explained in detail after a brief introduction of the technologies used.

1) Open Telemetry: Open Telemetry is a nascent project of the Cloud Native Computing Foundation (CNCF) and the result of a merger between the OpenTracing and OpenCensus projects. Its goal is to simplify the telemetry ecosystem by providing a unified set of instrumentation libraries and specifications for observability telemetry [16], [17].

2) Elasticsearch and Kibana: Elasticsearch is a distributed search and analytics engine, which provides near real-time search and analytics for all types of data. Kibana is the in-house dashboard for visualizing and analyzing data as well as managing, monitoring and securing the elastic stack [18].

- **Open Telemetry Libraries:** In order to receive data, the targets need to be instrumentalized. Open Telemetry provides this mechanism throughout libraries, which support manual (code modified) instrumentation as well as automatic (byte-code) instrumentation [17].
- **Open Telemetry Collector:** The Collector is a vendor-agnostic implementation to receive, process, and export telemetry data. It is the default location instrumentation libraries export telemetry data and it can be deployed in two different ways. First is an agent running with the application or on the same host as the application and second is a gateway running as a standalone service typically per cluster, datacenter or region [17].

In addition, we added a Kafka-Queue and another collector as optional components to the architecture. The queue provides a kind of buffer for the data in case the endpoint is temporarily unable to ingest data or the endpoint is unreachable. The optional collector is deployed as a gateway to provide advanced capabilities such as tail-based sampling. In addition, the Gateway can limit the number of egress points required to send data as well as consolidate API token management [17].

 ISBN: 978-1-61208-844-0

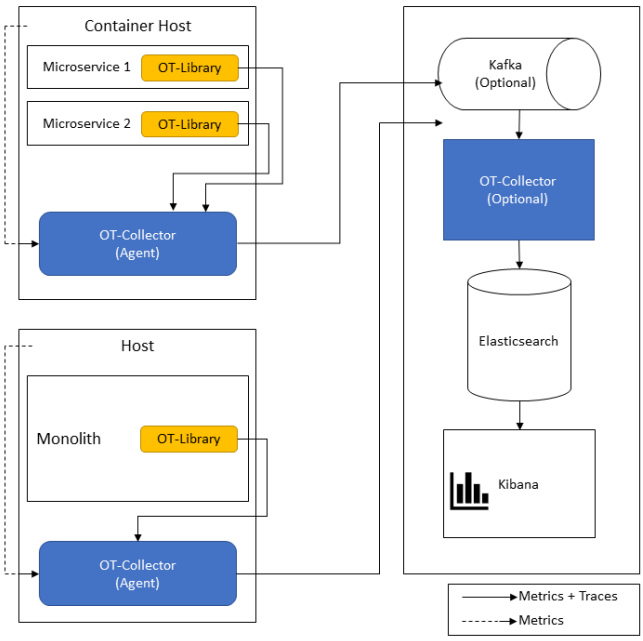

Figure 5. Technical reference architecture of the monitoring and logging environment using open telemetry

V. Conclusion and Future Work

In this article, we presented initial steps towards a reference architecture for microservices, which we are creating jointly with our partners from the insurance industry. The reference architecture aims to build compliant microservices-based applications that meet the specified guidelines and best practices.

We first give an overview of the architecture with its building blocks. We then focus on the operations responsibility area, by presenting conceptual and technical details on logging and monitoring.

The next steps in our research are the design of the business process component and the integration responsibility area. The latter is of particular interest as our partners operate a service-oriented landscape, so it's necessary to identify coexistence pattern to run a SOA and microservices-based applications concurrently.

References

[1] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term," https://martinfowler.com/articles/microservices.html, March 2014, [retrieved: 3, 2021].

[2] H. Knoche and W. Hasselbring, "Drivers and barriers for microservice adoption–a survey among professionals in germany," Enterprise Modelling and Information Systems Architectures (EMISAJ), vol. 14, 2019, p. 10.

[3] A. Hausotter, C. Kleiner, A. Koschel, D. Zhang, and H. Gehrken, "Always stay flexible! wfms-independent business process controlling in soa," in 2011 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, Dec. 2016, pp. 287–297, [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=intsys_v9_n34_2016_6. [retrieved: 3, 2021].

[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, Dec. 2018, pp. 269–281, [Online]. Available: https://www.thinkmind.org/index.php?view=article\&articleid=sysmea_v11_n34_2018_4. [retrieved: 3, 2021].

[6] 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: 3, 2021].

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

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

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

[10] 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 & European Conference on Software Architecture, IEEE, Ed., 2009.

[11] A. Koschel, A. Hausotter, M. Lange, and S. Gottwald, "Keep it in Sync! Consistency Approaches for Microservices - An Insurance Case Study," in SERVICE COMPUTATION 2020, The Twelfth International Conference on Advanced Service Computing, Nice, France, 2020, [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=service_computation_2020_1_20_10016. [retrieved: 3, 2021].

[12] 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). IEEE, 2016, pp. 1856–1860.

[13] M. Nygard, Release It! Design and Deploy Production-Ready Software. Pragmatic Bookshelf, 2007.

[14] A. Chuvakin, K. Schmidt, and C. Phillips, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management. Waltham, Massachusetts: Syngress Publishing, 2012.

[15] J. Turnbull, The Art of Monitoring. Turnbull Press, 2014.

[16] A. Parker, D. Spoonhower, J. Mace, B. Sigelman, and R. Isaacs, Distributed Tracing in Practice - Instrumenting, Analyzing, and Debugging Microservices. Sebastopol, California: "O'Reilly Media, Inc.", 2020.

[17] The OpenTelemetry Authors, "Documentation | OpenTelemetry," https://opentelemetry.io/docs/ [retrieved: 3, 2021].

[18] Elastic, "What is Elasticsearch? | Elasticsearch Reference [7.11] | Elastic," https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html, [retrieved: 3, 2021].

 ISBN: 978-1-61208-844-0

Executable Architectures for Complex Software Systems

Sebastian Apel
Technische Hochschule Ingolstadt
Ingolstadt, Germany
Email: sebastian.apel@thi.de

Thomas M. Prinz
Course Evaluation Service
Friedrich Schiller University Jena
Jena, Germany
Email: thomas.prinz@uni-jena.de

***Abstract*—The design and implementation of complex software systems can be achieved by modern software architecture styles and well-chosen tool stacks. The resulting systems have their benefits in technical cleanliness, reproducibility, and automation (e. g., of processes). However, there is a gap between the design of the system architecture and its implementation. Well-advised architectures get lost in tool configurations and implementations of simple service-to-service communications that both do not belong to the scope of the architecture. How could this gap be closed without losing the advantages of tool stacks? This paper introduces the idea of focusing not on toolstacks, but on data models, data streams, algorithms, and business logic. Instead of designing architectures for documentation and overview, the architecture itself represents the executable system software. Our main idea is to describe the data model used in architectures and provide a language to describe the data's transformations.**

Keywords–Software Architectures; Distributed Systems; Development Tools; Model Transformation

I. Introduction

Architectures describe abstract components of complex software systems and how they communicate. They follow fundamental software engineering principles for independent and isolated development: high cohesion and low coupling [1].

Modern development approaches allow implementing an architecture closer to the components' descriptions to generate rudimentary applications (stubs), e. g., by using the *Google Web Toolkit* (GWT) [2][3], *swagger.io* [4], or *jHipster* [5][6]. Another recent trend for architectures is *microservices* [7]. This trend forces to create service-oriented components in isolation that are independent and resilient. The resulting service components represent functionality whose combination results in the complete and complex software system [8].

The usage of such modern development approaches requires different, individual tool stacks [9]. These tool stacks include middlewares, construction tools, container formats, process automation, and services deployment. This allows services and processes to be deployed in different runtime environments. The overall result should be a service-based modern software architecture.

Microservice systems claim to have advantages in technical cleanliness, reproducibility, reliability, and automation processes [9]. In reality, however, there is a gap between the design of the system architecture and its implementation, which usually leads to discrepancies between them. Architectures get lost in tool configurations and implementations of simple service-to-service communications. As a result, all application developers, from programmers to software architects, have to operate at multiple abstraction levels to implement the architecture, with much time not spent in application logic. A real-world project by Apel et al. [10] has shown that the implementation overhead ratio between functional and additional code is sometimes less than or equal to 1 : 3. In other words, for 100 lines of functional code, 300 lines of organizational code are needed. *It would be a gain in time, cost, robustness, and correctness if the developer can focus only on the application logic.* But how could this gap be closed without losing the benefits of different tool stacks?

This paper describes an idea to close this gap in Section II. Section III discusses this idea shortly compared to existing ones. A short conclusion ends the paper in Section IV.

II. Idea

Our idea includes four main aspects: (1) a meta-language that allows programming in different programming languages, (2) compilation into existing tool stacks, (3) automation of the appropriate tool stack selection, and (4) a suitable development platform.

The language (1) that allows implementation in different programming languages can be interpreted as a meta-programming or domain-specific language. However, it does not have to be a new one. One can assume an extension of Java or another popular programming language, such as is done in *ArchJava* [11].

The goal of the language is to reduce the effort of managing and configuring services and using different programming languages for them. 85% of software engineers within one study use multiple languages to solve problems during software development [9]. Instead of developing each service on its own, the language provides a common execution environment and abstracts from their communication and deployment. One advantage is that common data models can be implemented centrally and used in all components. It also avoids tedious mapping of parameters. The disadvantage is that it seems somewhat centralized, where the advantage of an independent service implementation can increase its generalization and minimize its coupling. When designing such a language, this fact must be carefully considered.

Since there is a trend towards data streams and data science, the language should enable data orientation. In addition to defining data structures and functional programming, it should also allow processes that connect different data

 ISBN: 978-1-61208-844-0

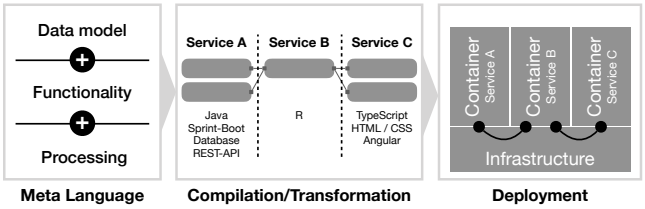

Figure 1. Linkage between meta language, compilation / transformation, and deployment.

streams.

Architectures described in the meta-programming language should be executable in an ad-hoc fashion as shown in Fig. 1. We propose to focus on both interpretation and compilation (2). Language interpretation has the advantage of fast error detection, debugging, and bottlenecks identification. Compilation should increase performance, especially if it distributes the various services across different (virtual) execution environments.

Traditional compilation translates a software system into one set of (virtual) system instructions. However, our idea is to compile the language into instructions in those programming languages that best fit the functionality's realization — in case the developer does not want to choose this and describes the functionality abstractly in the meta-language. In other words, since programming languages belong to different tool stacks, the compiler must translate the language into an individual set of tool stacks. Data structures, functions, and processing chains described at an abstract level would then have to be translated into multiple programming languages.

The compiled components and tool stacks must communicate with each other. A surrounding execution environment should enable this communication. Furthermore, compilation remains within the problem complexity of the architecture description. As in the case of the *Unified Modeling Language* (UML) [12], our goal is not to find a language that covers every use case by default. The language should provide bounds, and every part of it should be executable. It is not the idea to cover all existing development approaches. On the contrary, the focus is on questioning some daily development practices and searching for alternative approaches.

One difficulty with our idea is identifying those parts/ functionality of the language that will be compiled into the same components. Another difficulty is choosing an appropriate tool stack for these components (3). An automatic decision must interpret essential information provided in the language – such as functions and component-specific data. Another difficulty is the inclusion of existing dependencies (other systems, libraries, and services) and how their integration works within and between services.

Our idea follows well-established computer science principles in the problem description, compilation, and execution. As with ordinary programming languages, a development tool (4) should support development with the meta-language and with all phases of software development (planning, analysis, design, implementation, and maintenance). Since one of our primary goals is to reduce technical details, the tool should focus on problem-solving, i. e., describing and programming the software, rather than on the particular technical configurations.

This development environment is our final goal. It allows focusing only on architecture and business logic. Different components should be described in separate projects and supplemented by dependencies. However, since every project knows and uses the same meta-language, the development environment can provide support across programming language and tool boundaries. The application should be immediately interpretable in the environment. When projects are deployed, the application is compiled and made executable in the form of services with service-specific tool stacks. For example, the results of the compilation could be containers (such as Docker [13]) that are published. The compilation realizes the communication described in the architecture, and the service publication assures availability.

III. Short Discussion

Of course, our idea is not completely new and there are several, other approaches in the literature. For this reason, we compare our idea in the following with two approaches that have the same research direction but a different focus. The first of them is *ArchJava* by Aldrich et al. [11]. ArchJava is a *Java* extension that provides three new language constructs: *Components*, *Ports*, and *Connectors*. Components describe architectural components with their ports, i. e., what communication endpoints are needed and provided. Some components can be connected via connectors, which then results in a concrete software instance consisting of multiple components. ArchJava is very promising, but Aldrich et al. state as limitations that it is language-bound to Java and only runs on a single Java Virtual Machine (JVM). However, our goal is to be free of these limitations. In addition, ArchJava operates on a high architectural level and method calls are only allowed within components. Although this is understandable for an architectural view, in some cases developers would benefit from a low-level method call where (technical) architecture details are hidden from the developer. In particular, tool stack decisions are sometimes unnecessary or disruptive when high performance is not a concern. In summary, however, ArchJava offers good and clear concepts and its focus on implementation and language constructs should be strongly considered when implementing our idea.

A second approach to architecture-level design and implementation is Archface by Ubayashi et al. [14]. Archface is very high level in architecture decisions and is oriented towards UML. Like ArchJava, it provides the language constructs *component* and *connector* and a new one *architecture*. Components and connectors are special, abstract interfaces. Component interfaces describe the communication endpoints of the component, while connectors describe their interaction. Architectures finally define concrete implementations of the components and connectors and, therefore, define a concrete software instance. Archface is also promising, but seems to have the same limitations as ArchJava. Although

 ISBN: 978-1-61208-844-0

it can be implemented for different programming languages — like ArchJava —, it is unclear how different components communicate across languages. However, in summary, the ideas of Archface are valuable during implementation outlined by our idea.

IV. Conclusion

Our idea is a new executable meta-language that should support the complete application development lifecycle. This language covers data structures, functionality, and descriptions of processes. It enables that each part of the described application is compiled into a best fitting tool stack. The selection of service boundaries and tool stacks is done automatically. In addition, it automatically implements how the components (services) of the architecture communicate. The result is a correctly configured distributed application based on existing technologies. The developer benefits by always acting on the level of the architecture description to realize the application. To support the developer in using the language, one goal is to provide a development platform. This platform covers the development along the complete software development process. Starting with planning and analysis, the support is possible up to implementation and maintenance.

References

[1] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, Third, Ed. Addison-Wesley, 2013.

[2] A. Tacy, R. Hanson, J. Essington, and A. Tokke, GWT in Action, 2nd ed. Greenwich, CT, USA: Manning Publications Co., Feb. 2013.

[3] Google, "[GWT]," [retrieved: March, 2021]. [Online]. Available: http://gwtproject.org/

[4] Swagger, "The best apis are built with swagger tools," [retrieved: March, 2021]. [Online]. Available: https://www.swagger.io/

[5] M. Raible, The JHipster mini-book, 5th ed. USA: C4Media, 2018.

[6] JHipster, "JHipster - Generate your Spring Boot + Angular/React applications!" [retrieved: March, 2021]. [Online]. Available: https://jhipster.tech/

[7] M. Viggiato, R. Terra, H. Rocha, M. T. Valente, and E. Figueiredo, "Microservices in practice: A survey study," arXiv preprint arXiv:1808.04836, 2018.

[8] F. De Paoli, "Challanges in services research: A software architecture perspective," in Advances in Service-Oriented and Cloud Computing, A. Lazovik and S. Schulte, Eds. Cham: Springer International Publishing, 2018, pp. 219–227.

[9] H. Zhang, S. Li, Z. Jia, C. Zhong, and C. Zhang, "Microservice architecture in reality: An industrial inquiry," in IEEE International Conference on Software Architecture, ICSA 2019, Hamburg, Germany, March 25-29, 2019. IEEE, 2019, pp. 51–60. [Online]. Available: https://doi.org/10.1109/ICSA.2019.00014

[10] S. Apel, F. Hertrampf, and S. Späthe, "Towards a Metrics-Based Software Quality Rating for a Microservice Architecture - Case Study for a Measurement and Processing Infrastructure," in Innovations for Community Services - 19th International Conference, I4CS 2019, Wolfsburg, Germany, June 24-26, 2019, Proceedings, ser. Communications in Computer and Information Science, K. Lüke, G. Eichler, C. Erfurth, and G. Fahrnberger, Eds., vol. 1041. Springer, 2019, pp. 205–220. [Online]. Available: https://doi.org/10.1007/978-3-030-22482-0_15

[11] J. Aldrich, C. Chambers, and D. Notkin, "Archjava: connecting software architecture to implementation," in Proceedings of the 24th International Conference on Software Engineering, ICSE 2002, 19-25 May 2002, Orlando, Florida, USA, W. Tracz, M. Young, and J. Magee, Eds. ACM, 2002, pp. 187–197. [Online]. Available: https://doi.org/10.1145/581339.581365

[12] Object Management Group, OMG Unified Modeling Language — Version 2.5.1, Object Management Group Std., Dec. 2017, [retrieved: March, 2021]. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[13] Docker Inc., "Empowering App Development for Developers — Docker," [retrieved: March, 2021]. [Online]. Available: https://www.docker.com/

[14] N. Ubayashi, J. Nomura, and T. Tamai, "Archface: a contract place where architectural design and code meet together," in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010, J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, Eds. ACM, 2010, pp. 75–84. [Online]. Available: https://doi.org/10.1145/1806799.1806815

 ISBN: 978-1-61208-844-0

Towards Extending USEfUL-ness for Urban Logistics with Service-orientation

Richard Pump*, Sophie Gohde†, Maik Trott‡, Marvin auf der Landwehr‡, Arne Koschel*,
Volker Ahlers*, Lars Gusig†, Christoph von Viebahn‡
University of Applied Sciences and Arts
Hannover, Germany
*Department of Computer Science
†Department of Mechanical Engineering and Bio-Process Engineering
‡Department of Business Informatics
e-mail: {richard.pump | sophie.gohde | maik.trott | marvin.auf-der-landwehr |
arne.koschel | volker.ahlers | lars.gusig | christoph-von.viebahn}@hs-hannover.de

***Abstract*—In this paper the workflow of the project 'Untersuchungs-, Simulations- und Evaluationstool für Urbane Logistik' (USEfUL) is presented. Aiming to create a web-based decision support tool for urban logistics, the project needed to integrate multiple steps into a single workflow, which in turn needed to be executed multiple times. While a service-oriented system could not be created, the principles of service orientation was utilized to increase workflow efficiency and flexibility, allowing the workflow to be easily adapted to new concepts or research areas.**

***Index Terms*—Service Orientation; Urban Logistics; Decision Support Tool.**

I. Introduction

Urban logistic processes are currently transformed in many ways to reduce emissions, increase efficiency and follow new political guidelines [1]. This creates a complex environment for urban planners when making decisions, since many novel concepts can be utilized to achieve different objectives within the planning area, requiring new tools to support the decision making process [2]. To support urban planners in their decision making process, the project USEfUL created a web-based decision support tool that provides important information in an easy to comprehend way. One of the main goals of the project was to create an application that can easily be used while in discussion with other planners and decision makers.

A Workflow was devised to generate the data utilized in the web-based decision support tool, as many different domains had to be combined. First, data about the city Hannover was collected to select representative areas that could be used for the evaluation of novel logistic concepts. In the next step, simulation models were built that utilized the data to simulate the selected novel concepts within the representative areas, producing new data about the populations behavior and traffic. The newly created data was then analyzed using purpose built evaluation models, which derived simple tendencies that could be presented to the end user. As the last step, data had to be ingested into a web-based decision support tool.

The general workflow fits well with the application of a service-oriented software system, in which each domain team develops their own services, connected by a common service bus. Creating a complete service oriented system was however not possible due to constraints to time, budget and software development expertise. With service orientation providing many benefits within software development [3], the application of service orientation to other kinds of processes was considered. This leads to our research question: How can the principles of service oriented software development be applied to partially automated workflows?

In the following sections we will further implore the workflow as well as the application of service-orientation in a manual process. To this end Section II will discuss related work, before Section III will present the utilized service principles. The acquisition of data will be shown in Section IV, while simulations are discussed in Section V. Analysis of data and presentation within the web-based decision support tool are shown in Sections VI and VII respectively. Results are discussed in Section VIII and Section IX provides the conclusion to this article.

II. Related work

Few other tools have been created to publicly present the impacts of different logistic concepts on urban areas.

As one of the earlier projects, BestUFS [4] analyzed different urban logistic solutions in general, providing rough advantages and disadvantages of concepts. The effects of concepts were analyzed through living labs, implementing pilot projects and evaluating impacts. While rough guidelines are also provided by the web based decision tool developed in USEfUL, no pilot projects were utilized in the project, relying on simulations instead. Furthermore, BestUFS does not present the results in the form of a web-based tool, but as simple documents, reducing the user experience.

Another project of a decision support tool combined route planning and the implementation of urban transport via PPGIS data running on a tangible interface [5]. Using the statistical data of three European capitals a support tool was created. In addition, workshops have been executed to teach the participants. In the case of USEfUL, the generalization of the urban area was further fostered as well as the easy utilization.

The most important tool is the urban-transport-road maps shown by De Stasio et al. [6]. While the road maps show

 ISBN: 978-1-61208-844-0

similar key performance indicators to the user, the results are simulated in real time. To achieve real time simulations, a very rough grained simulation was utilized instead of concrete agent based transport simulations as used in USEfUL. Furthermore the urban-transport-road maps-tool is a single system integrating different modules instead of a workflow consisting of multiple different tools.

Bozzo et al. [7] present a literature survey and a theoretical ex-ante-framework for the evaluation of logistic concepts created in the project SIPLUS. However, no simulations were utilized and no concrete evaluation of logistical concepts are presented in the paper. Furthermore, the authors remain with a theoretical model, not implementing a concrete decision support tool.

Overall, very little works concentrate on creating and managing a workflow to evaluate novel logistic concepts and presenting evaluation results to a user. Other works often focus on single issues within a possible workflow, while holistic views are uncommon.

III. Principles of service-orientation

Providing the web-based decision support tool with data required a multi-step process that contained data collection, simulation, analysis and data ingest. All these steps had to be followed for many different logistic concepts that had to be evaluated in different research areas in respect to multiple key performance indicators. A fully automated approach was therefor likely to reduce overall project run time and increase productivity. Figure 1 shows the workflow as well as the different models utilized in the project.

Unfortunately, due to budgetary constraints, the project team consisted mostly of domain experts, firm enough in computer science to develop and maintain domain specific models but not firm enough to create an inter-domain system. Therefor, a manual approach utilizing applicable principles of service-orientation was developed.

In service-orientation, the following principles for service design are often listed (e.g., by Rosen et al. [3] or Huns and Singh [8]):

- Isolation of responsibilities - a service is responsible for a specific task and is the only service responsible for that task.
- Loose coupling - services are as independent as possible of each other.
- Encapsulation - the interface of a service is strictly decoupled from the implementation.
- Modularity - services are self-contained and can be combined to create new workflows.
- Autonomy - a services lifecycle is independent of other services.
- Statelessness - services are without state.

Following these principles allows for high flexibility in software architectures. By applying principles of service orientation to a manual workflow, changes (which often are necessary within a research context) to each step of the workflow should be able without impacting the project at large.

In the following sections, it will be shown how the principles were applied to the design of the different tasks needed to create data for the decision support tool and evaluate whether the application of the principles improved the workflow flexibility.

IV. Data acquisition

The first step in the workflow is the acquisition of data necessary for simulations. To simulate inhabitant behavior, and in turn traffic resulting from the behavior, exhaustive data about geography (i.e. roads, buildings, etc.), inhabitant distribution, and other structural information (distribution of living spaces vs. office buildings/other industries, logistical points of interest, etc.) was needed, as shown in Table I.

TABLE I. Key data for simulation of urban logistic behavior.

Categories	Key data
Traffic	1. road maps, 2. velocity limits, 3. number of vehicles, 4. level of service, 5. modal split
Area usage	1. public, living, industrial, retail areas, 2. coordinates
Public transport	1. network, 2. coordinates
Districts	1. Borders, 2. number of buildings, 3. number of inhabitants, 4. demographics

While most of the data were provided by the city administration of Hannover, further studies and statistics have been analyzed to complete the necessary database. The database provides the information to other parts of the workflow.

V. Simulation models

The second step in the workflow is the simulation of novel logistics concepts utilizing the previously described data as inputs. Since multiple logistics concepts needed simulation (the number of which was not pre-defined), the simulations were designed with common inputs and outputs as to flexibly interchange different models within the same workflow. The models were created using AnyLogic, a proprietary, java-based simulation tool, which provides extensive libraries and multi-method-simulation. A combination of discrete event simulation (DES) and Agent-Based Simulation (ABS) was chosen as the development methodology [9]. The combination allows the development of flexible models, enabling the simulation of different logistics concepts using the same base model.

In total, two base models were developed which can be configured to simulate six logistics concepts. While breaking with the principle of Isolation of responsibilities, since a single model may simulate different concepts, this choice was made due to software inflexibility within the simulation toolkit. The first model, CEP, simulates courier, express and parcel delivery services and provides the base for four of the concepts. The second model, E-Grocery, simulates food delivery in the research area via food fulfillment centers (DCenter) or traditional supermarket shopping. Both simulation models

 ISBN: 978-1-61208-844-0

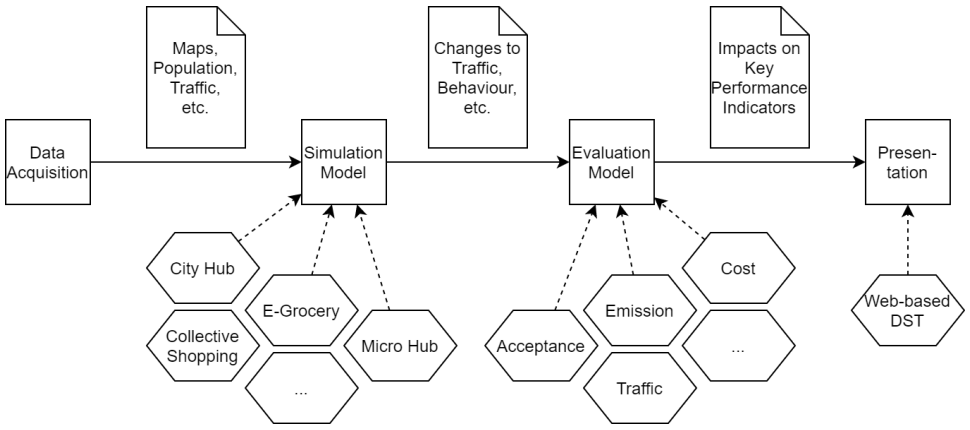

Fig. 1. Workflow of the project USEfUL.

utilize the same database, containing information about roads, buildings, inhabitants, etc. about the research areas.

TABLE II. SCENARIOS SIMULATED VIA ANYLOGIC MODELS.

Scenario	Description
Micro-Hub	The population is supplied by micro-hubs in the inner city area. A supply chain is created across different logistics levels.
White Label	The population of the CEP population is supplied by bundling orders from several CEP service providers in a common distribution center on the outskirts of the city.
City Hub	A stationary, inner-city transshipment point will be built, which will be used by several CEP service providers for last-mile distribution.
Parcel Pickup Locations	CEP service providers now only deliver via unattended services, in which orders are delivered to customers exclusively at stations/stores or via a drop-off location.
Online Grocery Shopping	Customers order consumer goods such as food and drug-store items from a local supplier with a specific delivery window to their desired location.
Neighborhood Logistics	Neighborhoods organize their mobility-triggering activities by linking and optimizing their routes through division of labor. Preferably, one neighbor does several activities for another neighbor (e.g. shopping activities).

In the following section, the E-Grocery model will be presented, which comprises the most comprehensive tool in particular with regard to the logistical complexity (time window routing).

A. E-Grocery model

This simulation model is a consideration of the real world problem of last mile grocery delivery. Within the e-grocery base model, the basic logic of the food delivery process (e-grocery) was mapped and contrasted with the classical purchasing process. This delivery concept was chosen because it is one of the most common in Germany and is used by our partner company. The delivery module of the simulation model shows accruing routes through grocery deliveries to the pilot neighborhoods from a distribution center (DCenter). The deliveries were route-optimized to achieve the highest possible degree of realism.

The inputs for the simulation model utilize publicly available data such as OpenStreetMap locations or anonymized data from the city of Hannover, municipalities or other external partners. All simulation models utilize the same base model of the research areas as well as the population living in the research areas. The simulation of different logistic scenarios is achieved by configuring the simulation model via parameters such as participation rate, consolidation of orders or delivery locations. Important input parameters such as the size of the delivery fleet, the order volume, the type of purchase (bulk purchase or small purchase), time window of the order (depending on the customer type) as well as the shopping behavior, the travel speed and the route guidance were parameterized in order to be able to analyze the model flexibly depending on different behavior and circumstance scenarios and to produce results that are as realistic as possible.

The classic shopping model is based on data on shopping and mobility behavior from the MID study [10] and provides reference values and logics for comparison between classic shopping and grocery deliveries. A more detailed description of our model can be found in [9]. In the publications [11] and [12], supplementary, later extensions of our model are shown, which consider the neighborhood types of the pilot area and the downstream supply chain of the eGrocery scenario in more depth.

Since influences of e-grocery on traffic are mainly determined by shopping behavior, different, behavior-oriented comparison scenarios were defined (see Figure 2).

Depending on their characteristics, these lead to different kilometers driven, a different number of start/stop operations, a different working time, and different emissions after interface transfer.

The simulation output is realised trough the creation of Excel-files which contain information common to all simulation models as well as some scenario-specific information like the success rate of delivering within specified time windows. A wide variety of simulation experiments were conducted for each scenario and over 1000 simulation iterations were performed.

 ISBN: 978-1-61208-844-0

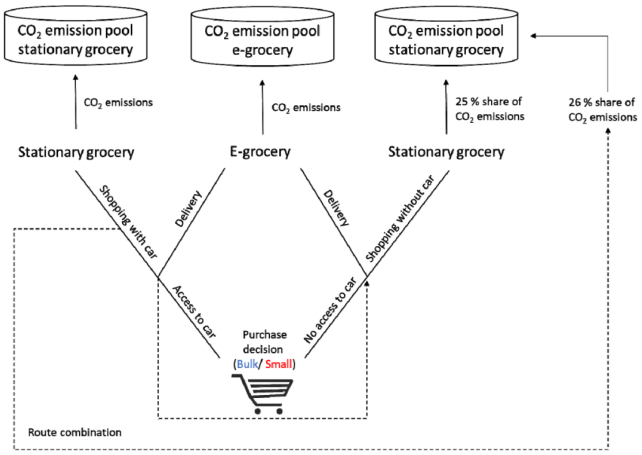

Fig. 2. Comparison of CO_2 emissions for e-grocery and normal grocery shopping [9].

The final Excel spreadsheet contains all iteration results for each scenario, shown in Table III.

TABLE III. OUTPUT VALUES OF THE E-GROCERY MODEL.

Data	origin	Measured variable
Mileage	AnyLogic	Total kilometers driven per agent type
Process times	AnyLogic	Duration tour / per vehicle type Utilization of vehicles or buildings Start/stop ratios
Number of tours	AnyLogic	Number of tours per vehicle type Deliveries made Returns (false acceptances)
Scenarios Information	AnyLogic	Scenario no. Iteration run & Simulation timestamp

Within the simulation model loose coupling as well as modularity is achieved through the definition of inputs and outputs. While some modifications needed to be made to accommodate each scenario, file structure is mostly identical between the models. This allows some interchangeability of simulation models in respect to data im-/export, reducing development time of interfaces in the database or the following evaluation models. In the context of the larger project, defining clear interfaces supported the parallel development of multiple simulation models as well as the database and the evaluation models, reducing overall communication workload. After common interfaces were defined only big changes had to be coordinated between all project teams, while small changes between two teams did not cause issues for other teams.

VI. ANALYSIS OF THE SIMULATION DATA

The next step within the workflow is the evaluation of the effects of logistics concepts on key performance indicators such as emissions, area use, costs and traffic behavior. The evaluation was done by comparison with the current traffic situation within the research areas, as logistics concepts influence multiple key performance indicators at once (Allen et al. [4]). For that a base case was defined and simulated to give a detailed overview about the actual situation.

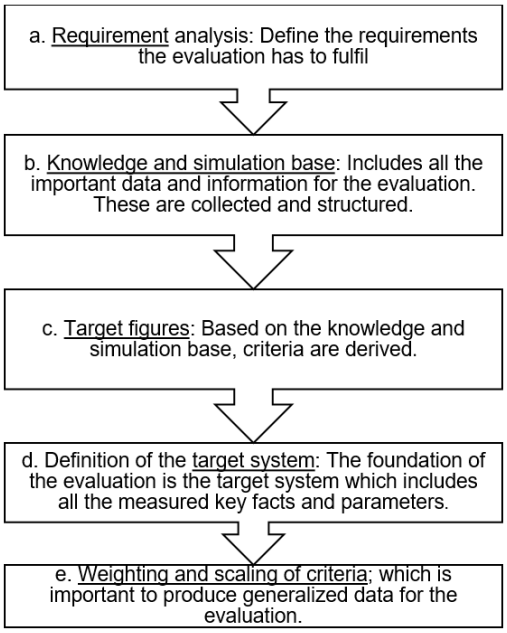

Fig. 3. Evaluation procedure modified from Drews and Hildebrand [13].

Every evaluation comprised of multiple steps as shown in Figure 3. Firstly, requirements have to be defined to figure out the main results of the analysis and the structure of the evaluation model. With the knowledge and simulation base a data base is created to collect specific data and structure them. The results of the data base are one part of the definition of the target figures. They are roughly defined before the execution of the AnyLogic simulation and specified with the knowledge base. The other part of the data base is the input for the target systems. The target systems are built to work up the simulation input. Each target figure gets its own system. At last, all the scenarios and models are summed up and scaled to make them comparable.

Before the modeling of the evaluation system some goals have to be set and defined. In the following sections each step is described in further detail.

A. Requirements

To model a significant evaluation system some core requirements have been set. The aspects shall guarantee a structured modeling process and an easy way to extend it once new logistic concepts should be implemented. In addition, the linking of the parameters and scenarios is desirable to release a comparable output.

The following aspects have to be taken into account while executing this evaluation modeling:

- transparent and replicable evaluation
- generalized assessment
- automation possible
- cross-district valuation based on the different criteria

 ISBN: 978-1-61208-844-0

B. Simulation and knowledge base

As seen in Figure 4 two different input types are used for the evaluation model: the research input and the simulation input. The research input provides information about the logistic concepts and the variation of data needed to evaluate them. The sources of the research input have been the data described in Section IV. Also, data from logistic companies were taken into account. The simulation output of the AnyLogic workflow and the research have been combined to build the basic structure of the target system. In addition, some of the research input also influences the specific definition of the target figures. Some of the most important inputs and outputs are listed in Figure 4.

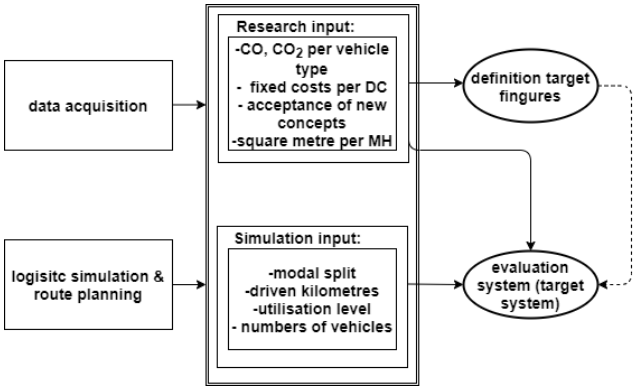

Fig. 4. Simulation output analysis.

C. Target figures

As shown in Table IV, the target figures are categorized into core targets and derivation targets. The core targets are emissions, costs, traffic and area savings.

TABLE IV. TARGET FIGURE CATEGORIES.

Core [unit]	Derivation 1^{st} [dependence]	Derivation 2^{nd} [dependence]
emissions [CO_2 equivalent]	ecologic BEP [CO_2 equivalent per €]	implementation potential [{CO_2/€;€;m^2;∅ km/h}]
costs [€ per day]	economic efficiency [profit (€) per day]	acceptance [{CO_2/€;€;m^2;∅ km/h}]
area savings [m^2]		
traffic [∅ km/h]		

Emissions Goal of the target figure is the reduction of CO_2 emissions and noise.

Costs For the last mile delivery a cost model is created. The balances are measured to the base case which provides information about the economic effects.

Traffic The overall intention is to reduce the traffic activity and congestion. One idea is to substitute individual traffic to commercial transport.

Area savings This target figure deals with the reduction of exploited economically used areas in the urban surrounding.

Due to the mutual influences (proportional, neutral, reciprocal) the core target values are connected. The derived target figures are deduced out of the core targets. Whereas the ecologic BEP (Break-even point) and the economic efficiency are calculated out of the core target figures directly, the acceptance and the implementation potential needed qualitative input which is challenging to evaluate.

1) Target system: Each target figure has got its' own analysis system. In this case the cost model is presented more detailed:

$$K_{total} = k_{fleet} + k_{DC} + k_{log}$$

The shown formula describes the three main modules of the cost model. The parts "fleet costs" (k_{fleet}) and "distribution center costs" (k_{DC}) are essential for all presented concepts. Due to changes in the supply chain in some concepts the third part "logistic costs" (k_{log}) is adapted, especially in the CEP delivery services.

With the following formula of the distribution center costs (k_{DC}) the process and the handling of the data should be emphasized:

$$k_{DC} = ((n_{DC} * (k_{pbs} + k_{st})) * c) * l$$

where n_{DC} is the number of distribution centers-output of the simulation, k_{pbs} are the fixed asset costs to run the distribution center (defined via research input), k_{st} are staff costs (defined via research input), c is the factor to define the capacity of the DCs-output of the simulation and l is a location factor defining cost changes based on ground values (defined via research input). The total costs of the last mile delivery are all compared to the firmly defined base case (BC). In case of the different CEP concepts, all concepts are more expensive compared to the base case. As seen in Figure 5 the White-Label (WL) concept and the central pick-up stations (PS) are slightly higher positioned while the Micro Hub concept (MH) and the City-Hub concept (CH) doubled or rather tripled the costs. In this project the costs are spread on four different quarters of the city. This was clarified by the fixed numbers of parcels per quarter. With this output of the target system the overall scaling is possible.

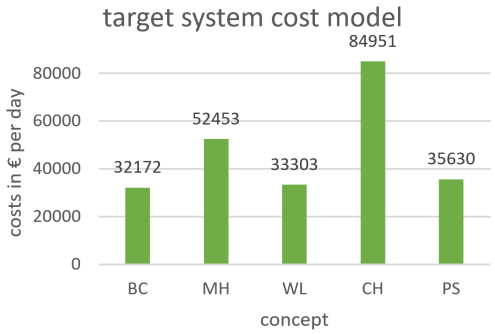

Fig. 5. CEP cost comparison.

 ISBN: 978-1-61208-844-0

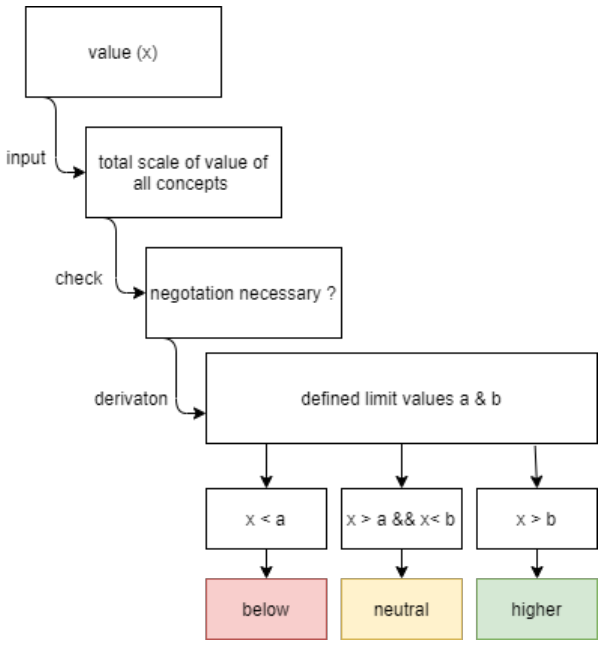

Fig. 6. Scaling system.

2) Weighting and scaling of criteria: With the valuation of the simulation all the concepts have to be comparable to give a clear statement about the tendencies and the possible changes. All values for a target figure have been collected to scale the values. Another important point has been the possible negotiation of some of the target figures like emissions, costs, area savings, ecological BEP and acceptance. With the categorization with limit values a tendency is shown as a result of the analysis as shown in figure 6.

As the entire process of evaluation is inherently independent of previous evaluations, the application of statelessness was trivial. Furthermore, evaluations are modular, since the evaluation of a single key performance indicator is independent form the evaluations of other KPIs Different KPI-Evaluations can be easily combined to create a clearer picture about each logistic concept, as each concept might affect different KPIs.

VII. Web-based decision-support-tool

Lastly, evaluation data was made available to users in the form of a web-based decision support tool. The tool presents evaluation results as well as information about logistic concepts, research areas and the project itself. The main design goal of the web-based decision support tools was ease of use. A user should be able to utilize the tool quickly e.g. in a meeting with other decision makers to discuss the impacts of novel logistic concepts on a given area. Through expert workshops the following requirements were refined:

- Present Information about:
  1) the project USEfUL
  2) research areas (districts)
  3) novel logistic concepts

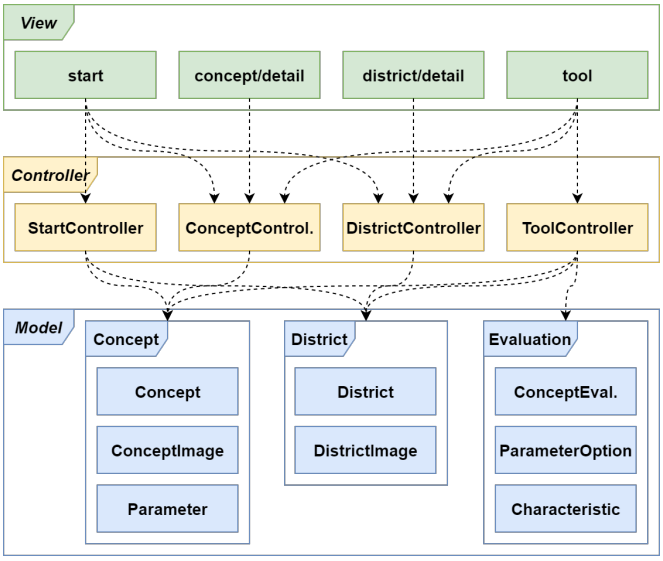

Fig. 7. Rough design of the web-based decision support tool.

- Allow users to view and export the evaluations of the concepts.
- Allow the modulation of concepts through the selection of different parameters.
- Compare the evaluation results of multiple concepts within a research area/across research areas.

The rough design of the web-based decision support tool is shown in Figure 7. Based on the industry standard model-view-controller-pattern, four different views present the user the most important information. The start page, showing rough overviews over logistics concepts, as well as research areas, serves as a landing page. From this page, the user can navigate to detail pages for concepts ans research areas (districts) or the decision support tool. Detail pages show images and in-depth information about concepts or districts and can be used to thoroughly understand the presented evaluations. The tool-page allows a user to select the combination of research area and logistic concepts, configure the concept with the parameters defined for the simulation models. After the user makes a selection, the evaluation results for the chosen combination are presented to the user.

The application was built utilizing the Laravel Framework which in turn required the use of php, javascript and a database (e.g. mysql). Docker was utilized to decrease setup times and increase productivity. Data import is handled by manually converting Excel files from the evaluation into RFC-compliant comma-separated-values, which in turn are imported into the database of the tool. If the web-tool is viewed as a service, this results in a violation of the statelessness of services. However, as the workflow is not fully automated, storing the results in the web-tool is necessary, as the results can not be recreated on-the-fly. Furthermore, execution of simulations takes a long time, violating the design goal of quickly presenting a user with the desired information.

 ISBN: 978-1-61208-844-0

VIII. Discussion

With the service principles originating from well structured workflows within industry solutions [3], the reverse application of the principles to a manual workflow as seen in this article was expected to benefit the project.

The principles of service design could easily be utilized as guidelines for the steps of a non-digital workflow and provided different benefits. The loose coupling of the different steps allowed each team to draw upon its expertise in the domain of the step (e.g. traffic analysis, simulation), while reducing communication needs. By applying encapsulation and defining data exchange formats before implementation of the tools used in the different steps, work could be parallelized within the overall project. In combination with statelessness, the encapsulation also supported the interchangeability of different models, e.g. the model to simulate the e-grocery-concept could be easily exchanged with the model for city hubs.

However, not all the service principles, which are usually applied to services, could also be applied to all steps of the workflow. While simulation and analysis were stateless steps, producing outputs only dependent on the inputs, data collection and presentation of results could not be implemented in a stateless manner, as the state of data is the main driving factor. Furthermore applying service principles to a manual workflow is inferior to a complete automation if the processes are to be executed repeatedly. However, within the context of research, where software is often a tool used a limited amount of times to generate data, the reduced expertise in the computer science domain necessary to create a partially automated workflow is advantageous for budget constrained projects.

Overall, the project benefited from aligning the workflow with service orientation. Other projects, which do not utilize a workflow that consists of multiple steps, each clearly confined to a different domain with own tools, might not benefit from the application of service orientation. E.g. a project that aims to create a software according to a users needs might profit more from an agile workflow allowing for many feedback loops.

IX. Conclusion and Future Work

The paper presented a novel application of service principles by focusing on a partially manual workflow instead of completely automated software solutions. The workflow of the project USEfUL was presented, which aims to create a web-based decision support tool for urban planners. To create the web-based decision support tool a multi-domain workflow was utilized to combine the expertise of different research teams. With each step focusing on a single domain, the application of the principles of service orientation was chosen to refine the workflow of the project. Through this service oriented workflow a decision support tool for urban planners was created to assist the evaluation and selection of novel logistic concepts for the development of urban spaces.

Applying the principles of service oriented software design to the partially automated workflow of the project, provided multiple positive effects on the projects efficiency. Modularity and encapsulation not only allowed interchangeability of models but also increased development speed by reducing communication needs. However the created solution is inferior to a fully automated software when repeated process use is a major goal.

In future work the construction of a fully automated service oriented system is the next logical step for the project USEfUL, since a fully automated system is often faster and more reliable than manual processes.

Acknowledgment

This work was supported by the Federal Ministry of Education and Research of Germany (project USEfUL, grant no. 03SF0547). We would like to thank our colleagues from the other institutions and the City of Hannover.

References

[1] B. für Bildung und Forschung, "Future cities strategic research and innovation agenda (zukunftsstadt strategische forschungs- und innovationsagenda)," https://www.bmbf.de/upload_filestore/pub/Zukunftsstadt.pdf, 2015, accessed: 2021-04-01.

[2] A. Lagorio, R. Pinto, and R. Golini, "Research in urban logistics: a systematic literature review," *International Journal of Physical Distribution & Logistics Management*, 2016.

[3] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer, *Applied SOA: service-oriented architecture and design strategies*. John Wiley & Sons, 2012.

[4] J. Allen, G. Thorne, and M. Browne, "Bestufs good practice guide on urban freight transport," 2007.

[5] C. Guerlain, S. Cortina, and S. Renault, "Towards a collaborative geographical information system to support collective decision making for urban logistics initiative," *Transportation Research Procedia*, vol. 12, pp. 634–643, 2016. [Online]. Available: https://doi.org/10.1016%2Fj.trpro.2016.02.017

[6] C. de Stasio, D. Fiorello, F. Fermi, A. Martino, G. Hitchcock, and S. Kollamthodi, "On-line tool for the assessment of sustainable urban transport policies," *Transportation Research Procedia*, vol. 14, pp. 3189–3198, 2016.

[7] R. Bozzo, A. Conca, and F. Marangon, "Decision support system for city logistics: literature review, and guidelines for an ex-ante model," *Transportation Research Procedia*, vol. 3, pp. 518–527, 2014.

[8] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet computing*, vol. 9, no. 1, pp. 75–81, 2005.

[9] M. Trott, M. Auf der Landwehr, and C. von Viebahn, "E-grocery of tomorrow - home delivery of food between profitability, customer acceptance and ecological footprint," *World Review of Intermodal Transportation Research (in press)*, pp. 10–, 2020.

[10] infas Institut für angewandte Sozialwissenschaft, "Mobility in germany: traffic - structure - trends (Mobilität in Deutschland: Verkehrsaufkommen – Struktur – Trends)," Deutsches Zentrum für Luft- und Raumfahrt e.V, IVT Research GmbH, infas 360 GmbH, Tech. Rep., September 2019, http://www.mobilitaet-in-deutschland.de/pdf/infas_Mobilitaet_in_Deutschland_2017_Kurzreport.pdf, last visit: 17.07.2019.

[11] M. Auf der Landwehr, M. Trott, C. von Viebahn, M. Putz, and A. Schlegel, "E-grocery in terms of sustainability-simulating the environmental impact of grocery shopping for an urban area in hanover," *Simulation in Produktion und Logistik*, pp. 87–96, 2019.

[12] M. Auf der Landwehr, M. Trott, and C. von Viebahn, "Simulation-based assessment of grocery shopping in urban areas," *Simulation News Europe, 30(4)*, pp. 145–158, 2020.

[13] G. Drews and N. Hillebrand, *Encyclopedia of project management methods (Lexikon der Projektmanagement-Methoden)*. Haufe-Lexware, 2007.

 ISBN: 978-1-61208-844-0