2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C) | 978-1-6654-3910-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSA-C52384.2021.00013

A Process Model for Microservices Design and Identification

1st Christoph Schröer
Very Large Business Applications
University of Oldenburg
Oldenburg, Germany
Email: christoph.schroeer@uni-oldenburg.de

2nd Sven Wittfoth
Futures Research
Volkswagen AG
Wolfsburg, Germany

3rd Jorge Marx Gómez
Very Large Business Applications
University of Oldenburg
Oldenburg, Germany

***Abstract*—Microservices have been established as an architectural style for developing modern applications. A major challenge in theory and praxis is to identify a adequate granularity of microservices.**

In order to achieve a consistent and easily understandable process, this paper presents a process model for the design and identification of microservices. This process model should support the development of microservices and the selection of suitable identification approaches. The process model is deductively derived on the basis of existing identification approaches from literature. This serves to classify the approaches into the phases of the presented process model but also to identify research gaps and thus new approaches.

The process model is evaluated first by integrating existing identification approaches into the model. Second, a case study shows that the process model allows for multiple perspectives on microservice architecture and can lead to different architecture alternatives. As a result, microservice architecture decisions can be better justified, compared and derived by software architects in a standardized way. We also show that the process model can be integrated into existing software development processes.

***Index Terms*—Microservices, Process Model, Software architecture, Software design**

I. Introduction

Microservices have been established as the architectural style for the development of modern applications since around 2014. They have frequently mentioned benefits like better scalability, maintainability and faster development cycles and can reduce technical dept in the entire application system [1].

However, there are also challenges in adopting microservice architecture such as initially increased complexity and the risk of data inconsistencies during operations. Also, adequate identification of microservices is still challenging. Here, the experience and knowledge of software architects is necessary. Badly decomposed services can end up in expensive refactorings later. For some years now, automated procedures have also been researched to support software architects during the design phase [2], [3].

The identification of microservices is based on the principles of cohesion and loose coupling. Cohesion means that the responsibility of a specific microservice is well defined along business capabilities. Loose coupling means that microservices can be run and deployed independently. On the one hand, microservices should be highly cohesive and on the other hand loosely coupled. In addition, there are other possible factors that can influence granularity and determine whether microservices are fine-grained or coarse-grained. The question is how a structured process model can support software architects in developing microservice architectures during the design phase [4]–[6].

An ongoing problem is that a uniform and structured process model does not exist to support the development, application and selection of approaches for identifying microservices. The goal of identification approaches is to determine adequate microservice candidates for example on the basis of software artifacts such as source code or logs. To solve this problem, a process model is developed and presented through an argumentative-deductive analysis in this paper. A uniform process model can increase the standardization, common understanding and comparability of microservice identification approaches and architectures. In general, a process model is a "representation of the design/product development/product creation process" [7]. Our process model focuses on and extends the design phase of the software development process. Furthermore, we show how identification approaches can be integrated into the process model. A case study evaluates that the process model enables different perspectives on software architecture in a uniform and structured way.

The paper is structured as follows. First, we summarize the related work in the context of service-oriented architectures in general and microservice identification process models. Second, we explain the research method, the research question and the results. Third, we describe the evaluation of the process models as well by the integration of new identification approaches as by case study. The case study is about an analytics application in the credit scoring domain. Finally, we conclude with the theoretical and practical implications and with the research outlook.

II. Related Work

A. Software development processes and service-oriented architectures

The problem of adequately delimited services is not new for microservices. Service-oriented architectures (SOA) also consider service cohesion and low coupling and determine

978-1-6654-3910-7/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSA-C52384.2021.00013

between fine-grained and coarse-grained services. Comprehensive process models already exist in this area such as the "Service-Oriented Design and Development Methodology" [4], shown in Fig. 1. This one integrates and links the phases of planning, analysis and design, implementation, service provisioning, deployment and operation. Compared to traditional software development processes, both have the design phase in common for example [9]. Additionally, [4] integrate principles of service cohesion and loose coupling into the process.

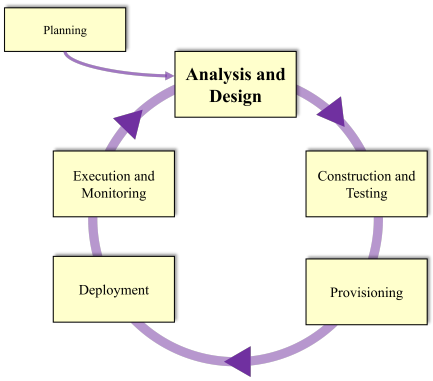

Fig. 1. Service-oriented design and development process model by [4].

The challenges on how to support software architects have already been addressed in SOA. For example, [10] modeled architecture decisions and organized them into a reusable model. Following this, the question of granularity also includes the structure of message exchange and the grouping of operations.

Concrete support regarding automated algorithms and how they can be integrated within a uniform process model is not covered in the mentioned articles. Our paper focuses on the design phase in Fig. 1 since our paper extends this phase by our proposed process model.

B. Process Models for Microservices Identification

Approaches for the automated and semi-automated identification of microservices can be found, for example, in the literature reviews of [6], [11], [12]. Identification approaches aim to determine microservice candidates based on software artifacts such as source code or logs.

Concrete process models exist but they specifically have to be adopted to the respective approach. For example, [13] specifies a "Service decomposition process" that provides the activities (1) create nano-entities, (2) create weighted graphs, (3) calculate candidate cut and (4) visualize candidate cut. Also, [14] presents a "process of proposed dataflow-driven decomposition mechanism for microservices": (1) Specify the use case (Function, Business capabilities), (2) construct data flow (Sentences), (3) identify microservice candidates.

There are also concrete process models for migration to microservice architecture for example in [11], [15], [16]. For example, the proposed activities in [15] are (1) collect information from the monolith, (2) grouping entities of the system into candidate microservices, (3) a visualization of the decomposition as a graph.

The mentioned process models have one limitation that they were very specifically developed for a particular identification approach. There is a large number of such approaches and thus also own process models. Our process model generalizes these concrete ones. This helps software architects to start designing microservices easily.

III. Process Model for the Design and Identification of Microservices

A. Method

For developing a process model, we choose the method of an argumentative-deductive analysis [17]. We combine this method with methodologies from qualitative content analysis defining the search string, inclusion and exclusion criteria to select relevant papers [18].

In general, the goal of the paper is to support software architects in microservices design and identification by a uniformed process. Therefore, we would like to answer the main research question: *How could microservice architectures be uniformly designed and compared?*

To operationalize the research questions, we adopted the search strategy from [12] and applied it in relevant scientific data sources ScienceDirect, IEEE and ACM Digital Library in 2020.

> ALL=((microservice OR microservices) AND (identif* OR extract* OR migrati* OR decompos* OR refactor*)) AND PY >= (2014)

We decided to limit the results regarding the publication year to 2014 since the concept of microservices has been established since 2014 [5]. We define the inclusion criteria according to which the studies must give hints about structured process steps and be written in English. We exclude papers that are out of scope, are grey literature and have no abstracts. Furthermore, we split the set of papers in a set using it for the argumentative-deductive analysis (2014 to 2019 having full years) and a set using it for evaluation (2020). We applied the search string in beginning of 2020 first and then during 2020 continuously.

The applied search string resulted in 101 papers. For the full years 2014 until 2019, we found 81 paper. For 2020, we found 21 paper. For developing the process model, we selected 31 paper as relevant. For evaluation, we selected 20 paper as relevant.

For developing the process model, we focus on methods and artifacts from the papers that could be used during design phase of the software development process. During reading the full texts, we continuously identified concrete process phases in the paper and derived abstract process phases deductively from the paper [12], [18].

B. Overview and Phases of the Process Model

Based on the relevant literature, we have developed a process model for microservice design and identification as shown in Fig. 2. This process model can be integrated into the

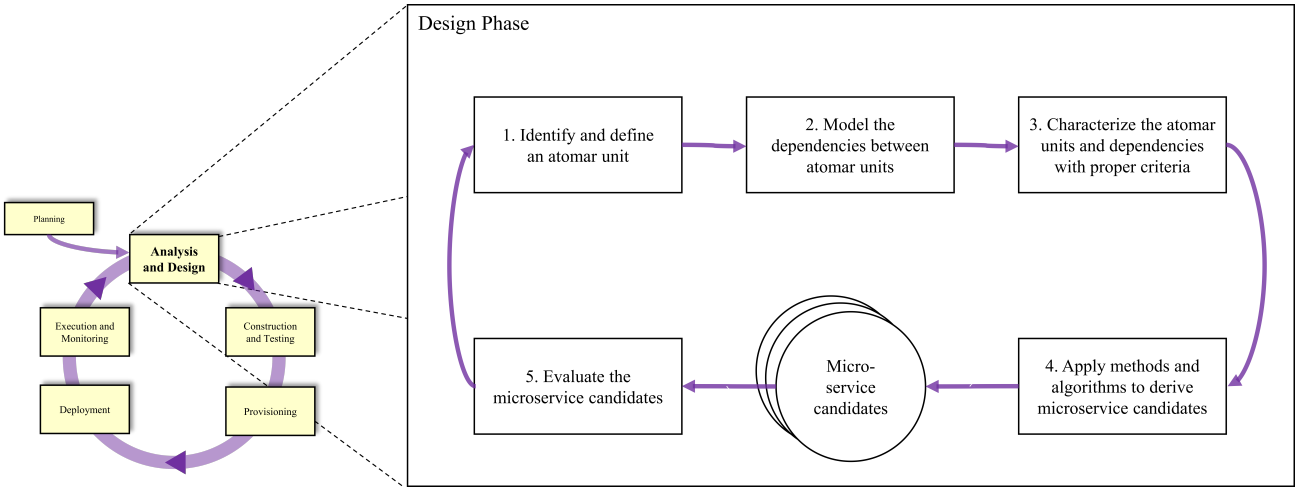

Fig. 2. Proposed process model for design and identification of microservices. This process model can be integrated into the design phase during software development process. Source: Authors.

design phase of Fig. 1 [4], [9] but can also be adopted in other software development processes with a similar design phase. We concretize this phase with methodologies and descriptions on how to conduct the design phase.

Before entering the proposed process model, one prerequisite is finalizing the requirement analysis and planning phase at least once. The requirement analysis is important to gather relevant artifacts for the design phase. In agile projects, the overall software development process can be conducted in multiple iterations. Therefore, our process model could be enriched with more information from passed phases iteratively, like already implemented source code from implementation phase or logs from monitoring phases. Further, different stakeholders can be involved. We aim that the process model supports software architects and developers. Therefore, most tasks will be addressed to them. Nevertheless, business experts or requirements engineers can be involved in the process as well [19].

For selection support of appropriate approaches for microservice identification in combination with this proposed process model, we refer to the literature review of [12]. The phases are described in the following sections.

1) Identify and Define an Atomar Unit (Phase 1): The goal of phase 1 is to identify atomar units. We introduce the concept of atomar units as there are starting points to start the decomposition process. In some papers, atomar units are also known as functional atoms [13]. Also, [11] use the concept of atomar units as granular building blocks of microservice architecture. The output of this phase is a textual specification of the atomar unit.

For example, atomar units could be an entity, a function, a functionality or an interface [12]. Relevant information can be found from artifacts of the requirements analysis or planning phase. Entities can be derived from use case descriptions. Functions are useful, if source code already exists in the context of a migration project.

2) Model the Dependencies between Atomar Units (Phase 2): The goal of this phase is to model the functional dependencies between the atomar units. For the functional requirements, use case descriptions or business processes could be take into consideration. Therefore, the understanding of the business processes and the requirements is essential.

There are different approaches and perspectives on how to model the dependencies: from a logical and from a physical point of view, or from a static and from a dynamic point of view. The modeling language is dependent on the chosen atomar unit. For example, the application can be modeled with an entity-relationship model to model the business entities, their data and their relationships [13]. An alternative is a dataflow-driven approach that models the data exchange between functions and operations [14]. Also, domain-driven design can be adopted in this phase since the bounded can be modeled based on ubiquitous language [20], [21]. The output of this phase is at least one model in an appropriate modeling language (like UML or entity-relationship notation etc.)

3) Characterize the Atomar Units and Dependencies with Proper Criteria (Phase 3): The goal of this phase is to consider important basic conditions both in terms of atomar units and the relationships between them. Besides the functional requirements (phase 2), these framework conditions also have a decisive influence on the software architecture, and are often referred to as non-functional requirements or quality attributes.

These conditions must be integrated into the models from phase 2. It is important to identify and describe the influencing factors as a whole. Business experts, requirements engineers, software architects and developers can be consulted to identify them.

In the microservices context, it is important to consider the principles of coherence as well as low coupling. Metrics can be

defined for this purpose. For example, [13] defines important coupling metrics and scales that can be used to characterize the relationships between atomar units. In the context of data flow diagrams, the relationships between function calls can also be described and quantified with the call frequency and the data volume.

Also the atomar units themselves can be further characterized. For example, the concrete technical attributes of an entity may describe semantic similarities and concepts between entities. Non-functional characteristics like team organization can also be integrated.

Fig. 3 clarifies how quality attributes can extend the model from phase 2 (atomar units and their relationships). The output of this phase is a textual specification of quality attributes regarding the relationships and atomar units.

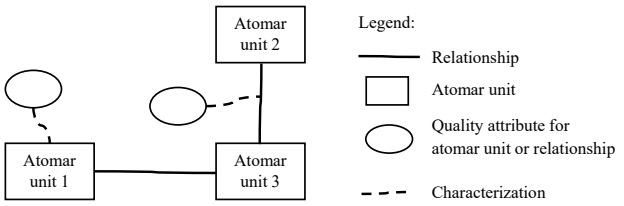

Fig. 3. Abstract model based on atomar units and relationships from phase 2, added with quality attributes for atomar units and relationships in phase 3. Source: Authors.

4) Apply Methods and Algorithms to Derive Microservice Candidates (Phase 4): The goal of this phase is the generation of microservice candidates as proposals and recommendations for software architects.

The previously created models (see Fig. 3) must be converted into a format that can be processed. We found in the paper different approaches like clustering, rule-based and optimization problems. Furthermore, the methods can be automatic, semi-automatic or manually. The output of this phase is an assignment of which atomar units could be grouped into microservices.

Showing how we have derived this phase we use the paper of [13] as an example. We could assign their clustering algorithm to this phase. The clustering algorithm determine microservice candidates. The clustering algorithms requires an entity-relationship model and manual estimates how strongly the entities are related. The former is constructed in phase 2 of our proposed process model. The latter are constructed in phase 3 of our proposed process model. With this example, we provide an insight how we have argumentative-deductively analyzed the paper from the search results and how we assign paper to the phases of the process model.

5) Evaluate the Microservice Candidates (Phase 5): The resulting microservice candidates should be evaluated in this phase.

Basically, case studies or experiments can be used for this purpose. Experiments aim to implementing first prototypes to measure quality metrics. Case studies are a rather inexpensive way to determine metrics at model level. For case studies, possible quality metrics derived from the relevant literature are for example number of resulted microservices candidates [12], [22]. For experiments, possible quality metrics derived from the relevant literature are for example throughput or response time [12], [23]. In addition, evaluation metrics for microservice architecture could also be used for evaluation. [1] present a Microservice Measurement Framework with four metrics: (1) coupling, (2) number of classes per microservice, (3) number of duplicated classes and (4) frequency of external calls.

As a result of the evaluation, the process model can be conducted multiple times to obtain alternative microservice architectures. Several possible microservice architectures with different microservice candidates can then be compared. Furthermore, the microservice architectures can also be evaluated with the experience of software architects, for example with structured interviews.

After selection of a microservice architecture, the further software development process can be followed, like implementation, test, deployment and operation of microservices. During test of implemented microservices, a further result can be that microservices have to be adjusted. Therefore, our proposed process model can be conducted in an incremental-iterative way as well.

IV. Evaluation of Process Model

We evaluate the process model twofold. First, we try to categorize paper from 2020 that we have explicitly excluded during the argumentative-deductive analysis (see Section IV-A). Second, we conduct a case study to apply the process model (see Section IV-B).

A. Evaluation with Approaches Published in 2020

The evaluation with papers from 2020 allows us to show the application and generalization of the process model for new approaches. Totally, we found 20 new approaches with the mentioned search string in 2020.

We show one adopted and evaluated example from [15] in Table I.

With the template from Table I, approaches can be (1) classified into the phases of our process model and (2) be easily compared with other approaches regarding each phase. Also, research gaps can be reached out. For example, giving the same preconditions of [15] until phase 3, we can research alternative clustering algorithms and compare such algorithms with the metrics from phase 5.

B. Case Study - Microservices for an Analytics Application in Credit Scoring

This section introduces a case study which will be used to show how our process model could be applied. The use case is taken from a open-source analytics platform, called KNIME [24]. The main requirement solving by this example is that a bank officer would like to know the credit worthiness of

TABLE I
ADOPTED PROCESS MODEL TO INTEGRATE THE APPROACH FROM "A COMPLEXITY METRIC FOR MICROSERVICES ARCHITECTURE MIGRATION" [15].

Phase	Description of adopted phase
1. Atomar unit:	Domain entity
2. Modeling the dependencies:	Read and write operations made to the domain entities are the dependecies between atomar units.
3. Modeling the influence factors:	Similarity measures and a new developed so called complexity metric are used to model the weights of the dependencies.
4. Algorithm:	A hierarchical clustering algorithm calculate cluster with microservice candidates.
5. Evaluation:	Software architects can use a visualization of the microservice candidates to evaluate the results. Further, several metrics and their correlation are used to evaluate the cluster of candidates.

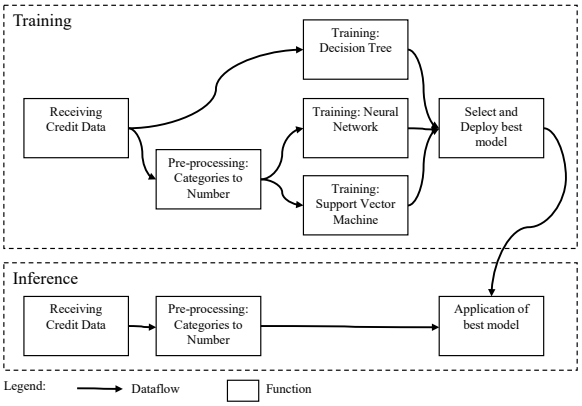

Fig. 4. Functional Worfklow of Credit Scoring Application based on [24]. Such a model can be used before entering the proposed process model.

customers supporting the decision about credit lending. This use case is already implemented in KNIME, but has not been migrated to a microservice architecture yet. A microservice architecture could be suitable to integrate the requirement into existing systems of a bank. For deriving a suitable microservice architecture, we will iterate through our process model three times to identify microservices for this use case.

The use case describes a classification problem from the credit risk domain to determine the credit scoring of customers. Based on selected customer and credit data (features), a credit scoring (good or bad) is calculated. For this purpose, classification models are trained and compared using accuracy. Then, the best model is selected to perform unknown and future credit scorings. The workflow is shown in Fig. 4 [24]. The workflow distinguishes between continuous training and the application of the model. The objective of the case study is to show how our process model works. Therefore, we will identify a suitable microservice architecture for this workflow.

In addition to applying our process model, we use three concrete identification approaches. Therefore, we have three iterations through the process model. First, we apply an identification approach by [14] called "A dataflow-driven approach to identifying microservices from monolithic applications". Second, we apply an approach by [25] "stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines.". Third, we apply an approach by [13] called "Service Cutter: A Systematic Approach to Service Decomposition". The three approaches are selected by the authors. An expanded list of approaches could be found in [12]. Table II shows the summary and comparison of the case study. The evaluation phase is conducted descriptively since the evaluation of the concrete microservice architectures is not focused in this case study.

First iteration based on "A dataflow-driven approach to identifying microservices from monolithic applications" by [14]: In order to migrate a monolith to a microservices architecture, [14] proposed a top-down decomposition approach driven by dataflows of business logics. We would like to use this approach to decompose the workflow of credit scoring. First, we follow the steps of our process model to start. Then, we search in the approach of [14] to find the information we need to start the decomposition process. Table II shows the results. For example, the initial workflow in Fig. 4 is a level 0 dataflow diagram that the approach requires.

In Phase 2 of our process model, we need to transform this model into an abstract format modeled in Fig. 5. A concrete dependency is called a sentence that describes two functions called each other, like $\{$*Data Store 1, P1*$\}$. The functions are modeled as processes. The data are exchanged via data stores. For example, P3 represents the training functions and need preprocessed data from data store 3 and persists a trained model in data store 4. The approach by [14] does not integrate further quality attributes. Therefore, the functional dependency from phase 2 is sufficient.

In phase 4, this model is applied by the algorithm developed by [14] to find one single microservice candidate. The main reason is the high coupling that is modeled due to the workflow.

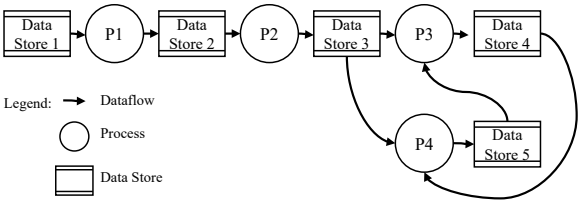

Fig. 5. Dataflow diagram at phase 2 of the proposed process model (first iteration). Notation based on [14].

Second iteration based on "stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines." by [25]: The approach by [25] is similar to the prior approach. First, the atomar units are also functions. Function calls model functional dependencies (see Fig. 3). Therefore, the phases 1 to 3 are almost the same. However, conducting phase 4 of our proposed process model, [25] recommends a microservice candidate per function. Therefore,

five microservice candidates can be inferred. This microservice architecture can be evaluated by measuring the data rate during an experiment, as [25] describes.

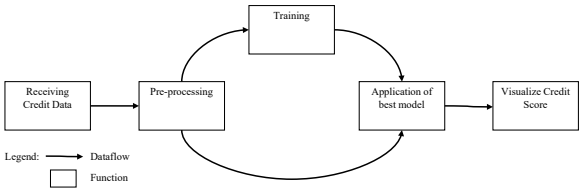

Fig. 6. Function-oriented diagram at phase 2 of the proposed process model (second iteration). Notation based on [25]

Third iteration based on "Service Cutter: A Systematic Approach to Service Decomposition" by [13]: The service cutter is a structured approach based on a criteria-driven method and architectural knowledge. The conducted approach for this case study is listed in Table II. The entities are derived from the dataset and workflow entities from [24].

The Service Cutter approach is based on the atomar units data, operations or artifacts. For our case study we choose data represented by entities. We identified the entities based on our knowledge from the datasets. We distinguish between credit data and credit score due to having features and categories. We do not further concretize the credit data entity, for example customer data, account data, etc. Also, we do not show the attribute of the entities. For this case, the granularity is sufficient to make the case study less complex. The training entity holds information about the used model algorithm and has relationships to the preprocessed credit data. Thus, it points out that the training is based on that data. The entity-relationship model is shown in Fig. 7. We model the relationships between the entities as "has a" or "is based on" dependencies.

[13] lists many coupling criteria. Each dependency from the model of phase 2 has to be estimated regarding relevant coupling criteria by software architects manually on a predefined scale. For example, we can model the dependency between credit data and credit score as a strong dependency.

A hierarchical clustering algorithm calculates clusters for microservice candidates. Therefore a range of microservice candidates is from one single microservice up to seven microservices in this case. This depends on the preference of the software architects how many microservices are suitable or required.

C. Comparison and Discussion

To sum up, the three identification approaches come to different solutions. The proposed process model supports here to make the approaches more comparable. First, the atomar units can be pointed out directly. Atomar units strongly influence the choice of the concrete identification approach [12]. However, the first phase of our process model can be conducted without having selected a concrete identification approach. Following the case study, if only functions as atomar unit are known, the approach by [13] was not appropriate since it requires entities. Furthermore, different ways for modeling are possible

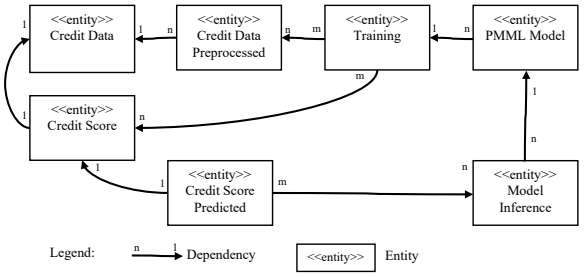

Fig. 7. Entity-relationship model at phase 2 of the proposed process model (third iteration).

and multiple criteria can be used. The proposed process model supports the main activities for the design and identification of microservices.

Assuming that no microservice architecture is suitable after the third iteration, a further iteration could be started. In this iteration, it is also possible to implement a new approach. Thus, the new approach also implies a research gap and could be investigated. Such approaches could also be a combination of existing identification approaches.

V. Conclusion

To achieve a consistent and easily understandable process, this paper presents a process model for the design and identification of microservices. This process model should support the development of microservices and the selection of suitable identification approaches. The process model is deductively derived on the basis of existing identification approaches from literature. This serves to classify the approaches into the phases of the presented process model and also to identify research gaps and new approaches.

The process model is evaluated first by integrating existing identification approaches into the model. Second, a case study shows that the process model allows for multiple perspectives on microservice architecture and can thus lead to different architecture alternatives. As a result, microservice architecture decisions can be better justified and derived by software architects as it allows to compare the design process and the microservice architecture itself in a standardized way. We also show that the process model can be integrated into existing software development processes.

Two limitations of our paper should be mentioned. First, our research methodology only considered literature only from relevant scientific databases, although grey literature like blog posts could also be interesting in future. Second, the process model is developed and also evaluated at least by the authors.

In future, the process model will be evaluated with software architects to show how the process model leads to better comparable microservice architectures and the practical application in other domains or more complex software development projects. Furthermore, future research focuses on developing new identification approaches based on the proposed process model.

TABLE II
ADOPTED PROCESS MODEL FOR THE APPROACHES DURING CASE STUDY.

Iteration: → **Name of Approach: →** **Phase of our process model ↓**	**1** **"A dataflow-driven approach to identifying microservices from monolithic applications" [14]**	**2** **"stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines." [25]**	**3** **"Service Cutter: A Systematic Approach to Service Decomposition" [13]**
1. Atomar unit:	Function	Function	Entity
2. Modeling the dependencies:	A function call to another function is defined as the functional dependency.	A function call to another function is defined as the functional dependency.	We model the relationships between the entities as "has a" or "is based on" dependencies.
3. Modeling the influence factors:	In this approach, additional factors have not been modeled.	In this approach, additional factors have not been modeled.	Each dependency has to be estimated regarding relevant coupling criteria by software architects manually on a predefined scale.
4. Algorithm:	A rule-based method leads to one microservice candidate.	Manually, each function is translated to a microservice candidate.	A hierarchical clustering algorithm calculate clusters for microservice candidates and leads to one up to seven microservices.
Resulted number of microservice candidates	1	5	1 – 7
5. Evaluation:	Following [14], coupling (afferent and efferent coupling, instability) and cohesion (relational cohesion) metrics can be used.	The evaluation is focused on the data rate per minute and also compared to a monolith.	A visualization tool supports the evaluation of the microservice candidates. [13] also points out, how fast microservice candidates could be identified.

This paper has theoretical and practical implications. Theoretically, research gaps could be identified. Practically, the process model allows clear and structured comparisons and validations of different identification approaches and microservice architectures. Thus, architecture decisions could be better contrasted and validated.

REFERENCES

[1] D. Taibi and K. Systä, "A Decomposition and Metric-Based Evaluation Framework for Microservices." in *Cloud computing and services science*, ser. Communications in computer and information science, D. Ferguson, V. Méndez Muñoz, C. Pahl, and M. Helfert, Eds. Cham: Springer, 2020, vol. 1218, pp. 133–149.

[2] H. Zhang, S. Li, Z. Jia, C. Zhong, and C. Zhang, "Microservice Architecture in Reality: An Industrial Inquiry," *IEEE International Conference on Software Architecture (ICSA)*, 2019.

[3] J. Soldani, D. A. Tamburri, and W.-J. van den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, 2018.

[4] M. P. Papazoglou and W.-J. van den Heuvel, "Service-oriented design and development methodology," *Int. J. of Web Engineering and Technology (IJWET)*, 2006.

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

[6] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *Journal of Systems and Software*, vol. 150, pp. 77–97, 2019.

[7] K. Gericke and L. Blessing, "An Analysis of Design Process Models Across Disciplines," *Proceedings of International Design Conference*, vol. 70, pp. 171–180, 2012.

[8] D. C. Wynn and P. J. Clarkson, "Process models in design and development," *Research in Engineering Design*, vol. 29, no. 2, pp. 161–202, 2018.

[9] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[10] O. Zimmermann, *An architectural decision modeling framework for service-oriented architecture design: @Stuttgart, Univ., Diss.,*, 2009.

[11] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From Monolith to Microservices: A Classification of Refactoring Approaches," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, J.-M. Bruel, M. Mazzara, and B. Meyer, Eds., vol. 11350. Cham: Springer International Publishing, 2019.

[12] C. Schröer, F. Kruse, and J. Marx Gómez, "A Qualitative Literature Review on Microservices Identification Approaches," in *Service-Oriented Computing*, ser. Communications in computer and information science, S. Dustdar, Ed. Cham: Springer International Publishing, 2020, vol. 1310, pp. 151–168.

[13] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, "Service Cutter: A Systematic Approach to Service Decomposition," in *Service-Oriented and Cloud Computing*, M. Aiello, E. B. Johnsen, S. Dustdar, and I. Georgievski, Eds., 2016, pp. 185–200.

[14] S. Li, H. Zhang, Z. Jia, Z. Li, C. Zhang, J. Li, Q. Gao, J. Ge, and Z. Shan, "A dataflow-driven approach to identifying microservices from monolithic applications," *Journal of Systems and Software*, vol. 157, pp. 1–16, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121219301475

[15] N. Santos and A. Rito Silva, "A Complexity Metric for Microservices Architecture Migration," *IEEE International Conference on Software Architecture (ICSA)*, pp. 169–178, 2020.

[16] H. Knoche and W. Hasselbring, "Using Microservices for Legacy Software Modernization," *IEEE Software*, vol. 35, no. 3, pp. 44–49, 2018.

[17] T. Wilde and T. Hess, "Forschungsmethoden der Wirtschaftsinformatik: Eine empirische Untersuchung," *WIRTSCHAFTSINFORMATIK*, vol. 49, no. 4, pp. 280–287, 2007.

[18] P. Mayring, *Qualitative content analysis: theoretical foundation, basic procedures and software solution.*, Klagenfurt, 2014.

[19] M. H. Gomes Barbosa and P. H. M. Maia, "Towards Identifying Microservice Candidates from Business Rules Implemented in Stored Procedures," *IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 41–48, 2020.

[20] A. Krause, C. Zirkelbach, W. Hasselbring, S. Lenga, and D. Kroger, "Microservice Decomposition via Static and Dynamic Analysis of the Monolith," *IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 9–16, 2020.

[21] E. Evans, *Domain-driven design: Tackling complexity in the heart of software*. Upper Saddle River, NJ: Addison-Wesley, 2011.

[22] M. Kamimura, K. Yano, T. Hatano, and A. Matsuo, "Extracting Candidates of Microservices from Monolithic Application Code," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018, pp. 571–580.

[23] A. Sriraman and T. F. Wenisch, "μ Suite: A Benchmark Suite for Microservices," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, 2018, pp. 1–12.

[24] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *Data Analysis, Machine Learning and Applications*, ser. Studies in classification, data analysis, and knowledge organization, C. Preisach, Ed. s.l.: Springer-Verlag, 2008, pp. 319–326.

[25] K. Tserpes, "stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines," *ICT Express*, vol. 5, no. 2, pp. 146–149, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405959519301092