Towards an Ontology-driven Approach to Model and Analyze Microservices Architectures

Gabriel Morais, Dominik Bork, and Mehdi Adda

To appear in:

MEDES'21: 13th International Conference on Management of Digital EcoSystems

Final version available via DOI (to be added once published):

www.model-engineering.info

Towards an Ontology-driven Approach to Model and Analyze Microservices Architectures

Gabriel Morais
Département de mathématiques, d'informatique et de génie, Université du Québec à Rimouski (UQAR)
Lévis, Québec, Canada
gabrielglauber.morais@uqar.ca

Dominik Bork
TU Wien, Business Informatics Group
Vienna, Austria
dominik.bork@tuwien.ac.at

Mehdi Adda
Département de mathématiques, d'informatique et de génie, Université du Québec à Rimouski (UQAR)
Lévis, Québec, Canada
mehdi_adda@uqar.ca

ABSTRACT

Microservices Architectures (MSAs) are continuously replacing monolithic systems toward achieving more flexible and maintainable service-oriented software systems. However, the shift toward an MSA also requires a technological and managerial shift for its adopters. Architecting and managing MSAs represent unique challenges, including microservices' identification, interoperability, and reuse. To handle these challenges, we propose an Ontology-driven Conceptual Modelling approach, based on the Ontology of Microservices Architecture Concepts (OMSAC), for modelling and analyzing microservices-based systems. We show, how OMSAC-based conceptual models, stocked in a Stardog triple store, support Stakeholder-specific communication, documentation, and reuse. This paper reports on the application of our approach in three open-source MSA systems with a focus on microservices' discovery based on similarity metrics. Eventually, we compare the extracted similarity metrics derived from the application of machine learning techniques to the OMSAC models with a manual analysis performed by experts.

ACM Reference Format:
Gabriel Morais, Dominik Bork, and Mehdi Adda. 2021. Towards an Ontology-driven Approach to Model and Analyze Microservices Architectures. In *International Conference on Management of Digital EcoSystems (MEDES '21), November 1–3, 2021, Virtual Event, Tunisia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3444757.3485108

1 INTRODUCTION

Microservices Architecture (MSA) is a recent architectural style, considered as a Cloud-native architecture [7] that has met significant success in the industry and a growing interest in academia [8, 10]. MSAs handle complexity by decomposing large systems and by bringing modularity "to the next level," [10] facilitating components' reuse [7, 28]. Indeed, architecting MSA-based systems means the adoption of a compositional design approach. Thus, the resulting architecture is a set of microservices composed to meet business requirements.

This compositional approach encourages reuse which facilitates systems development. Designers composing a microservices based system should be able to identify existing reusable microservices that could be used as-is or extended to compose new systems, leading to faster systems development [3, 4] and increased quality. Thus, identifying similar microservices is a prerequisite to enable microservices reuse.

MSA changes the technology stack impacting the design, development, deployment, and operation tasks. In the design phase, architects should be aware of the system's functional, technological, operational, and organizational aspects. For instance, some components can be developed and operated by different teams, coded in different languages and use different platform-provided services. Consequently, the question of how to efficiently model and analyze knowledge about MSA-based systems arises.

We believe that an ontological approach could address the stressed challenges, and, consequently, present an Ontology-driven Conceptual Modelling approach based on the Ontology of Microservices Architecture Concepts (OMSAC) [25] to describe MSAs in a holistic form. Our approach aims to support MSA-based systems modelling and improve automated analysis of large MSAs by providing a tailored domain ontology. Our approach's fundamental characteristics are flexibility and extension. It allows representing MSA systems holistically or decomposed into individual viewpoints that meet different stakeholders' specific information needs. All views can be derived from the single underlying OMSAC ontology, represented with the Web Ontology Language (OWL2 DL) [40]. Eventually, the OMSAC ontology can be automatically processed e.g. to efficiently identify similar microservices in large knowledge bases.

We evaluate our approach's feasibility in the form of a use case, using three open-source MSA systems to discover microservices based on similarity metrics. The resulting similarities are expressed as a synthetic metric that allows modellers to rely on it when analyzing microservices interchange or replacement. Ultimately, we compare the extracted similarity metrics derived from applying machine learning techniques to the OMSAC models with a manual analysis performed by experts. The evaluation material can be found on this paper's companion source code repository [26].

The remainder of this paper is structured as follows: Section 2 briefly introduces the relevant background before we introduce our new approach in Section 3. Section 4 presents the use case based evaluation. In Section 5 we discuss the findings before we close with a conclusion and future work directions in Section 6.

MEDES '21, November 1–3, 2021, Virtual Event, Tunisia

ACM ISBN 978-1-4503-8314-1/21/11…$15.00
https://doi.org/10.1145/3444757.3485108

2 BACKGROUND

In the following, we briefly introduce the foundations of microservice architectures (Section 2.1), conceptual modelling and ontology-driven conceptual modelling (Section 2.2), and provide a concise review of related works (Sections 2.3 and 2.4).

2.1 Microservices Architecture

Microservices are small, autonomous services that work together and are "focused on doing one thing well" [28]. They are the components that form a microservice architecture, which is "a distributed application where all modules are microservices" [10]. The most recurrent Microservice properties in the literature are: Independency (share nothing), modularity, organization around a specific functionality, and single responsibility.

The inherent complexity of distributed architectures establishes a major challenge faced by industry when architecting and maintaining MSAs [8, 10]. Shared persistence, monitoring, exchange mechanisms, security and modularization challenges seem to be exacerbated in microservices-based architectures. Likewise, defining the appropriate size and number of microservices, correctly bound business contexts, manage polyglotness, and adequate skill and expertise make up challenges directly related to MSA principles [19].

2.2 Ontology-driven Conceptual Modelling

Ontologies in systems engineering are "an explicit specification of conceptualization" [36]. In software engineering, they are models used to represent and codify a subject of matter. Those models are descriptive and normative; they aim to formalize domains into knowledge structures by generically describing relevant concepts, properties, and relationships. They include a vocabulary of related terms and their significance, allowing the structure to ease knowledge sharing [36]. Also, they can be used to support other AI technologies [39], which on the other hand, can extend ontologies' capabilities [9]. Concretely, the knowledge representation is based on named graphs which are data models for objects and their relationships. In these graphs, objects are represented as nodes, object relationships as edges, and knowledge is organized through subject-predicate-object statements [2]. Compared to other knowledge structures like relational databases, an ontology allows high expressiveness and reasoning capabilities.

Conceptual modelling "focuses on communication, learning and problem solving among human users" [38]. Using ontologies extends this notion to machine understanding, as ontologies are understandable by humans and machines [36]. Applying ontologies in conceptual modelling thus enables several benefits, such as a formal specification of the semantics [5], bringing reasoning on the content of a conceptual model, improving domain-specific knowledge reuse, and enhancing the domain's structural and behavioural description [13, 38].

Guizzardi et al. [18] define Ontology-driven Conceptual Modelling (ODCM) as a discipline that applies ontological theories to develop engineering artifacts "for improving the theory and practice of conceptual modelling." Indeed, ontologies (e.g. foundational and domain ontologies) have also been used to improve semantic integration to facilitate the interoperability of models, methods, languages, and paradigms [37]. ODCM is also suitable for handling complexity in modelling large systems and describing complicated domain aspects. Likewise, ODCM improves reusability, reliability, and domain understanding [38]

2.3 Modelling Microservices

We believe modelling has great potential to address some of the challenges stressed at the outset. However, there is a "lack of conceptual models able to support engineers since the early phases of MSA development" [22], and a lack of "a uniform way to model autonomous and heterogeneous microservices at a level of abstraction that allows easy interconnection through dynamic relation" [22].

Various approaches to represent microservices architectures have been proposed: Informal drawings [1], UML based diagrams [20], Domain-specific Languages (DSL) [6], directed graphs [1], programming languages [17], and modelling languages [30]. However, modelling microservice-based systems using these approaches does not address common challenges in software modelling like the analysis and exploration of multiple viewpoints [34] and modelling in different granularity levels [29], because in these approaches, each viewpoint remains a separate model. Thus, the modelled viewpoints are, if at all, analyzed and explored separately.

2.4 Measuring Microservices' Similarity

Calculating similarity is paramount to address challenges like reuse, interoperability, and interchangeability. However, establishing similarity metrics in MSA is a challenge on its own because of the variety of criteria that could be considered.

While the importance of identifying similarities in MSA seems intuitive, existing works on microservices' similarity are limited. Some of them had mentioned similarity metrics used when decomposing a monolithic application into microservices [11]. Others had investigated mechanisms to identify [23] and establish variability [4] among microservices for reuse purposes. Microservices identification and variability remain as open research challenges [3]. Benni et al. [4] proposed a set of four factors to establish whether a microservice is interchangeable: *sharing feature sets*, *interaction compatibility*, *communication compatibility*, and *no coevolution*.

On the other hand, several similarity measurement approaches have been proposed to enhance services' reuse in web services architectures, which are compositional architectures. These approaches mainly use the structure of web services (extracted from the source code based on programming language standards or derived from functional descriptions), semantic analysis, and a combination of the two. They rely on web services functional descriptions formalized using the Web Service Description Language (WSDL) and its extensions [15]. However, these approaches have limitations inherent to the quality and form of the descriptions using WSDL [31]. Ontological approaches have been suggested to address those limitations [21, 33]. Despite the fact that ontologies perform better when identifying similarities [31], these benefits remain marginal because of the costs imposed and the ontological expertise required when implementing them [15, 33].

Consequently, extending similarity approaches from the web service context without an existing ontology to microservices would be challenging. Firstly, the absence of a standard language to describe microservices concomitantly with the diversity of programming

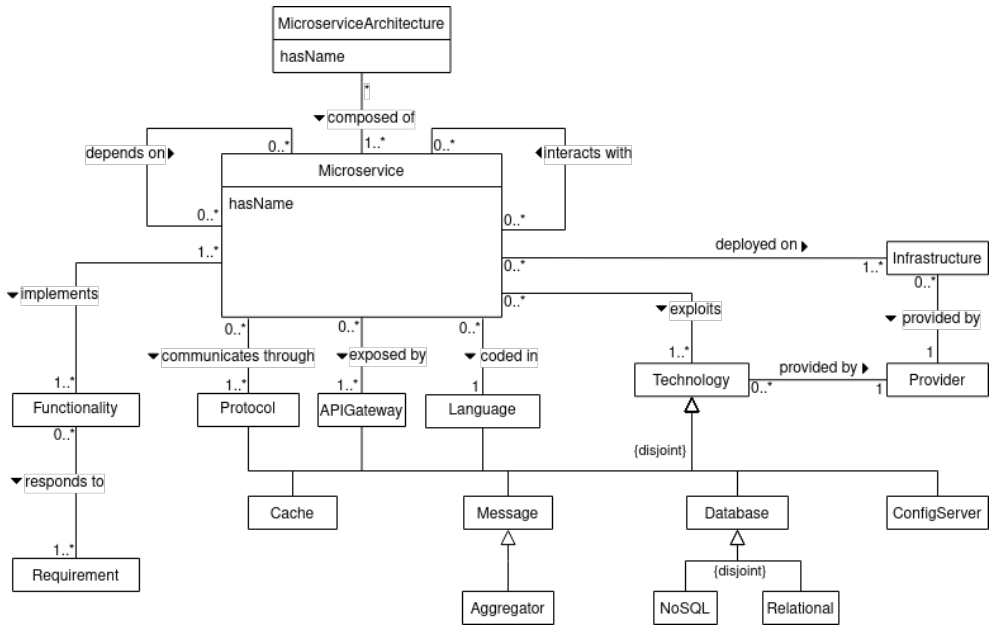

Figure 1: Excerpt of the OMSAC's ontology (derived from [25])

languages used to code them are barriers to establish similarity using structural approaches. Lastly, establishing similarity in the microservices domain calls for using multiple viewpoints and aspects other than functional; thus, using ontological approaches "will be more reliable" than the lexical dictionaries used in common web services similarity approaches [14].

The holistic representation of the MSAs proposed in this article could provide modellers with the necessary tools to describe microservices and handle MSA modelling and similarity metrics challenges, allowing comparison of microservices based both on their structure and semantics.

3 THE ONTOLOGY OF MICROSERVICE ARCHITECTURE CONCEPTS – OMSAC

OMSAC [25] is a domain ontology focusing the microservices architecture domain, formalized in OWL2 DL. OMSAC aims to support modelling, exploring, understanding, sharing knowledge and using MSA concepts to build MSA-based systems. It enables: representing MSA (anti-)patterns; cognifying MSA's supporting tools to improve designer understanding of MSA's principles; describing and stocking models to manage MSAs; building machine learning models for MSAs' classification and analysis.

During MSA-based systems design, OMSAC allows modellers to represent various conceptual aspects of these systems used in development life cycle tasks. Our approach has explored this OMSAC capability by applying a subset of its terminological component (TBox), covering the concepts needed to model an MSA system. Fig. 1 offers a UML view of an OMSAC ontology's excerpt to support the comprehension of the subsequent use case. For a detailed description of OMSAC's TBox, see [25]. Once the system is modelled, the created assertion component (ABox) can be explored using a triple-compliant query language and an inference engine.

4 USE CASE

In this section, we show how OMSAC can be applied to model (Section 4.2) and analyze (Sections 4.3 and 4.4) MSA-based systems. We have modelled three MSA-based systems using OMSAC concepts presented in Fig.1. These three systems are *Hipster Shop*, *eShopOnContainers*, and *Vert.x*. They contain between seven and ten microservices developed in different programming languages (C#, Go, Java, Javascript, Python), implementing a set of (common and shared) features, and deployed in containers – for more details on these systems please see [3]. We have chosen these systems because of their technological variety and the number of shared functionalities they implement. Using these systems allowed us to analyze them by applying different techniques and measurements, identify similarities between them, and discover potentially interchangeable microservices based on functional and technical aspects.

4.1 Preparatory Steps

We have primarily relied on the results from Benni et al. [4] and Mendonça et al. [23] to represent the three selected systems using OMSAC. Those works have analyzed the three systems in-depth to extract all functional aspects and some technical aspects from the microservices composing them. Despite the detailed information provided by these works, we have identified a lack of operational aspects while modelling these three systems with OMSAC.

Consequently, we have explored the files related to these systems' deployment processes from their source code repositories and extract the deployment dependencies and the platform-provided services they use. We have analyzed mainly the Dockerfiles, which allowed us to expose deployment dependencies among microservices and services sharing. The latter has exhibited dependencies that the previous works did not analyze and unveiled deployment aspects that should be considered when analyzing microservices for interoperability or reuse. Table 1 provides a sample of the data

Table 1: A sample of the Basket Microservice's analyzed data

Features	Interactions	Coevolution	Services	Communication
Create Cart Get Cart Add item Update cart Checkout Cart	Catalog Microservice Ordering Microservice Identity Microservice	Catalog Microservice Ordering Microservice	Cache Redis	gRPC

extracted for the Basket Microservice, one of the microservices composing the *"eShopOnContainers"* system.

We relied on Protégé [27] version 5, an open-source ontology editor, to model the systems and build the ABox, and on Stardog [35] version 7.5.1 (under academic licensing) and Docker version 20.10.5 to create the knowledge base by uploading the OMSAC's TBox and the ABox containing our models. The created knowledge base would then be ready to be explored in an environment supporting triple-queries and inferences; for this purpose, we relied on the Startdog Studio tool to build and run SPARQL [41] queries. Stardog offers machine learning capabilities, which we used for computing similarity metrics.

4.2 Modelling MSA-based Systems with OMSAC

Once we collected and analyzed the data, we mapped the identified concepts to the respective classes and relations in OMSAC's TBox, presented in Fig 1. Then, we modelled the systems by creating individuals and linking them following the identified concepts and unveiled relations from the analysis step. The outcome is an ABox containing the three systems modelled using OMSAC's TBox and the OWL2 language. Fig. 2 shows the OMSAC-based conceptual model of the *Basket Microservice* in a UML fashion while Listing 1 shows this model in OWL2.

```
@prefix omsac:
<http://www.semanticweb.org/ontologyOfMicroservicesArchitecture#>.
@prefix msa:
<http://www.semanticweb.org/WebBasedMicroservicesModels#>.
{ msa:BasketMicroservice a omsac:Microservice;
    omsac:constitutes msa:eShopOnContainers;
    omsac:deployableOn msa:AKS;
    omsac:implements msa:CheckoutCart, msa:CreateCart,
                     msa:GetCart, msa:UpdateCart;
    omsac:codedIn msa:CSharp ;
    omsac:interactsWith msa:CatalogMicroservice,
                        msa:OrderingMicroservices,
                        msa:IdentityMicroservice;
    omsac:exploits msa:RedisCacheEShop ;
    omsac:exposedBy msa:APIGatewayShoppingMobile,
                    msa:APIGatewayShoppingWeb,
                    msa:ShoppingAggregatorMobile,
                    msa:ShoppingAggregatorWeb;
    omsac:dependsOn msa:CatalogMicroservice,
                    msa:OrderingMicroservices;
    omsac:communicatesThrough msa:gRPC;
    omsac:hasName "Basket_Microservice"@en.}
```

Listing 1: OWL2 specificaiton of the Basket Microservice in OMSAC

4.3 Analyzing MSA-based Systems modelled with OMSAC

We have relied on the following competency questions (CQ) listed in Table 2 to explore and analyze the modelled systems - an approach widely adopted in ODCM (cf. [12, 16, 38]). Here, we present the analysis of the *eShopOnContainers* system and its Basket Microservice. These CQ are answered by SPARQL queries, which extract sub-graphs from the knowledge base. As various criteria can be used in these queries, this mechanism allows extracting different information to meet different stakeholders' needs. Also, we relied on these sub-graphs to measure similarities among the Basket Microservices and other microservices from all three systems.

Results for CQ1 to CQ5 are sub-graphs that focus on the system's functional aspects; thus, representing functional models. Fig. 3 shows the CQ3 sub-graph which can meet system analysts' needs. CQ6 and CQ7 focus on technical aspects; thus, the sub-graphs extracted represent technological models that could address operators' and integrators' needs. Finally, for CQ8 and CQ9, the extracted sub-graphs contain multiple microservices' aspects, addressing multiple needs. Table 2 shows the OMSAC's classes and relations used for each query; complete queries are available on this article source code companion [26].

4.4 Measuring Similarity

To explore the knowledge base for identifying similar microservices, we relied on three different approaches: Stardog's ML similarity model, EdgeSim [24], and manual analysis by an expert. Below, we introduce the three methods, their execution results are discussed in Section 5.

4.4.1 Using Stardog. The similarity model proposed by Stardog is an ensemble model based on three different approaches: Syntactic, semantic, and structural. Syntactic similarity is measured on the labels' characters by applying e.g. "edit distance, fuzzy string matching or trigram cosine similarity". Semantic similarity is measured on the meaning of the labels by exploiting "a manually curated lexical database (e.g. WordNet) or a separately trained word embedding model". Structural similarity is measured through the schema's structure, which detects "relationships having the same source and target types" [32].

```
INSERT { graph spa:model { :simModel a spa:SimilarityModel;
         spa:predict ?microservice;
         spa:arguments (?features ?protocols
                        ?interactions ?coevolutions).}}
WHERE { SELECT (spa:set(?f) as ?features)
        (spa:set(?p) as ?protocols)
        (spa:set(?i) as ?interactions)
        (spa:set(?c) as ?coevolutions)
        ?microservice
{ ?microservice omsac:implements ?f.
  OPTIONAL{?microservice omsac:interactsWith ?i.}
  OPTIONAL{?microservice omsac:dependsOn ?c.}}
  OPTIONAL{?microservice omsac:communicates ?p.}
GROUP BY ?microservice}
```

Listing 2: Creation of a Stardog similarity model using criteria proposed in [4]

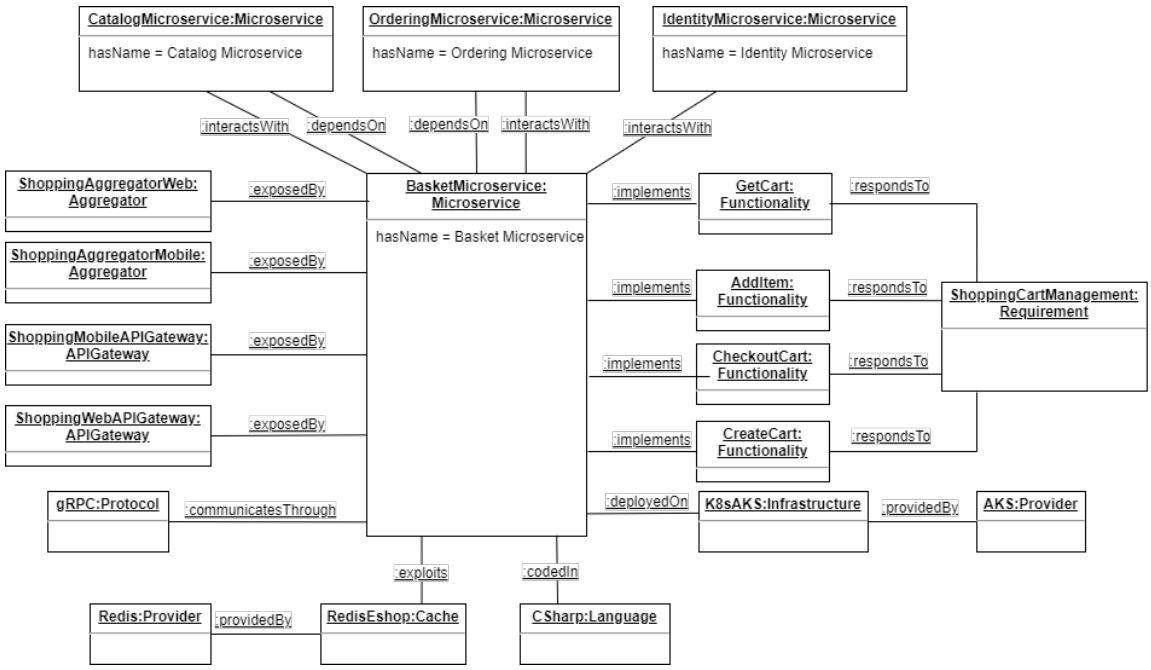

Figure 2: OMSAC model of the Basket Microservice

Table 2: Use of OMSAC elements to respond to the Competency Questions

Competency Questions	OMSAC Classes	OMSAC Relations
CQ1 - Which microservices compose a system?	MicroserviceArchitecture Microservice	isComposedOf
CQ2 - Which features are implemented by a microservice?	Microservice Functionality	implements
CQ3 - Which MSA requirements are met by a microservice?	Functionality Requirement Microservice	implements respondsTo
CQ4 - Which features are related to a given feature?	Functionality	interactsWith, dependsOn
CQ5 - What is the shortest path between two features?	Functionality Microservice	all
CQ6 - Which are the technical dependencies of a microservice?	Microservice Technology	exploits, exposedBy communicatesThrough
CQ7 - Where is a microservice deployed?	Microservice Infrastructure	deployedOn
CQ8 - Which interconnections exist between microservices?	Microservice	interactsWith, dependsOn
CQ9 - Which microservices are similar?	Microservice Functionality Technology	implements, interactsWith communicatesThrough dependsOn, codedIn

We have created three different models: Based on the criteria from Benni et al. [4], based on functional attributes, and based on technological attributes. These models have been created by executing SPARQL "insert" queries where we declared the features and the prediction variables, and provided a graph extracted from the knowledge base, which is used as training data. Listing 2 shows the SPARQL query for the first model.

4.4.2 Using EdgeSim. The EdgeSim metric was proposed by Mitchell et al. [24] to calculate similarity among graphs, taking into account the internal and external edges between entities, which can be weighted. This metric has been used to calculate the similarity between two clusters in software decomposition approaches. The general formula of the EdgeSim metric is:

$$EdgeSim(A, B) = \frac{Y}{E} \quad (1)$$

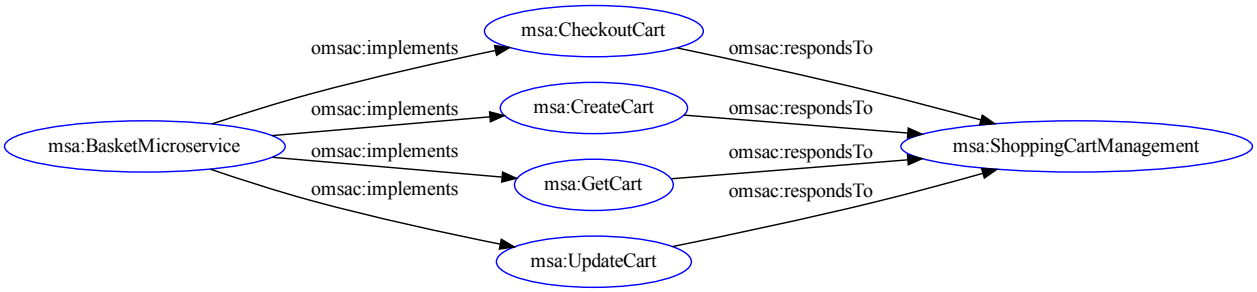

Figure 3: Basket Microservice Functional Model

Where A, B are graphs, Y is the sum of the edges' weights that are of the same type in both graphs (inter and intra-edges), and E is the sum of the weight of all edges in the graph, when all edges have the same weight, we set it to 1. EgdeSim works with clusters having the same edge numbers. In our case, there may be differences in the number of relations each microservice has. Thus, here, we considered only the number of relations of the Basket Microservices to the computation of E, and gave each relation the same weight. Then we have implemented it using SPARQL queries. Listing 3 shows the request used to calculate the sum of Y, for the complete version of this implementation see [26].

```
SELECT ?microservice
       ?similarMicroservice
       ((?yFeature+?yInteracts+?yProtocol+?yDependsOn) as ?Y)
{ SELECT ?similarMicroservice
(count(distinct ?feature) as ?yFeature)
(count(distinct ?interaction) as ?yInteracts)
(count(distinct ?protocol) as ?yProtocol)
(count(distinct ?dependencies) as ?yDependsOn)
WHERE { VALUES ?microservice {msa:BasketMicroservice}
?feature ^omsac:implements ?microservice, ?similarMicroservice .
OPTIONAL{?microservice omsac:dependsOn ?anotherMicroservice.
    ?anotherMicroservice omsac:implements ?dependencies.
    ?similarMicroservice omsac:dependsOn ?anotherMicroservice2.
    ?anotherMicroservice2 omsac:implements ?dependencies.}
OPTIONAL{?microservice omsac:interactsWith ?otherMicroservice.
     ?otherMicroservice omsac:implements ?interaction.
     ?similarMicroservice omsac:interactsWith ?otherMicroservice2.
     ?otherMicroservice2 omsac:implements ?interaction.}
OPTIONAL{?protocol ^omsac:communicates
         ?microservice, ?similarMicroservice.}
} GROUP BY ?similarMicroservice }
```

Listing 3: EdgeSim metric based on the criteria introduced in [4]

4.4.3 Manual/Informal Approach. The manual-informal analysis of these systems was done by a practitioner using the data from the first step of our approach, which we cleaned to enhance understanding. We have limited the analysis to the microservices implementing the cart management functionality and limited the information provided to the features, interactions, coevolution, platform-provided services, and communication technologies related to them.

5 EVALUATION

OMSAC-based models support information needs by allowing the description of various aspects of microservices-based systems at the same place. These models' ontological nature makes them intelligent, as we can make inferences on the knowledge they represent, and accessible to machines and humans, as both can explore these models. When describing MSAs using OMSAC, we create a unique model containing all the relevant information, which means having the functional, technological, operational and organizational information into the same knowledge base, which can be queried to respond to specific information needs. As shown above, we can extract different sub-graphs from the knowledge base to meet different stakeholders' needs. It addressed challenges in the analysis and exploration of multi-viewpoints and modelling in different granularity levels [29] because, in OMSAC-based models, all viewpoints are derived from one single model, which allows analyzing and exploring them at once.

We can use the OMSAC-based models as input to measure similarity metrics. This measure can be computed based on the extracted sub-graphs. As shown in the previous section, we can apply different metrics to the same data. Further, the similarity identification can also facilitate from the application of machine learning techniques as such techniques extend the similarity computation to hidden relations and consider models' semantics and structure.

5.1 Comparison of similarity identification approaches

To evaluate the different similarity approaches we defined a scenario in which a similar microservice, suitable to replace the *Basket Microservice* of the *eShopOnContainers* system, needed to be identified among all microservices of all three use case systems, composing 25 microservices. We applied Stardog, EdgeSim, and the manual approach. With respect to the manual approach, we were further interested in which aspects the expert considered the most and the less relevant for the similarity identification. We translated the relevance into weights to normalize the perceived similarity.

The expert analysis outcome is that the *Shopping Cart Microservice* from the *Vert.x* system is the most similar one in terms of features. However, technological differences are perceived as a barrier to interchange them. Thus, *Hipster Shop* system's *Cart Service* seems to be more suitable to replace the *Basket Microservice*. Table 3 summarizes the results.

Table 3: Manual-informal expert's similarity analysis

Microservice	Basket	Cart	Shopping Cart
Functional	9/9	2/9	5/9
Technical	2/2	2/2	0/2
Expert Score	100,00%	61,11%	27,78%

Table 4 shows the metrics obtained with each technique for the Basket Microservice using the criteria proposed in Benni et al. [4]. The expert score served as a baseline to compare them. The results show that the similarity obtained using Stardog's machine learning approach performs better than the EdgeSim metric and is close to the expert's.

Table 4: Comparison of the similarity metrics approaches.

Microservice	Expert	Stardog SM	EdgeSim
Basket	100,00%	100,00%	100,00%
Cart	61,11%	56,57%	22,22%
Shopping Cart	27,78%	33,54%	27,78%

5.2 Discussion

Using a manual-informal approach based on expert analysis to calculate similarity among microservices is time-consuming. Experts need to explore raw data gathered from different documents as source code, models and architectural artifacts. Calculating the EdgeSim metric, the semi-automatic approach needs an in-depth knowledge of the different hidden relations between microservices to build accurate queries which provide the input for similarity calculation. These queries should include extensive relations considering both direct and hidden relations. Likewise, queries ignore semantic aspects, as query projections will consider specific individuals in a relationship that limits similarity metrics' accuracy. Consequently, using only queries could lead to passing by commonalities.

For instance, when querying the knowledge base using criteria from Listing 2 but without the Stardog similarity model, the results differed for interaction and coevolution. The projection considered two individuals as strictly different even if they were close considering feature implementation and technology stack. Consequently, we have changed the query to compare features implemented by microservices sharing interaction and coevolution relationships.

Finally, the machine learning (ML) approach using Stardog Similarity Model has had results close to the manual-informal approach. It has performed similarity identification with short analysis time, as we can focus on relevant relations, which eliminates the need for in-depth knowledge of the knowledge base's structure. Beyond, the Stardog Similarity Model applied in our use case shows the perspective of using semantic comparison, the distance between individuals, and the knowledge base's structure to measure similarity. Furthermore, the advantage of using ML techniques in this context grows following the amount and the diversity of microservices in the knowledge base.

This research of course also comes with limitations. The most obvious one concerns the generalizability (external validity) of the evaluation results. We successfully applied OMSAC to model and analyze the three presented systems, while future research needs to apply it to more MSAs. Moreover, the similarity metrics were calculated based on a rather narrow complexity with 25 microservices. However, this was a purposeful decision as we wanted to involve the manual approach by a human expert who would not be able to manually comprehend and analyze larger models. We thus expect that our approach is even more valuable when the complexity of the MSA exceeds the cognitive abilities of humans.

6 CONCLUSION

This paper presented an ontology-driven conceptual modelling approach illustrated by a use case that demonstrated the relevance of modelling MSA-based systems using a unified semantic description language which allows discovery and comparison of microservices. OMSAC-based models represent MSA systems holistically but enable efficient derivation of stakeholder-specific viewpoints that meet specific information needs.

These models support the identification of existing microservices and provided services, which could be reused as-is or in an extended version and highlight technical and platform-driven concerns, which are mandatory for efficient design. We showed, that this approach features similarity metrics using various criteria, which support microservices identification for reuse. Likewise, we demonstrated that using machine learning techniques can simplify the computation of such metrics.

Future work will focus on establishing the most suitable properties to be used in a machine learning model to classify components of an MSA-based system and build variability metrics. We also plan to develop a domain-specific language (DSL) to encapsulate OWL2 and SPARQL queries to enhance OMSAC implementation, providing a simplified vocabulary shaped for domain experts. Besides, we plan to develop supporting tools and an intelligent model projection mechanism based on OMSAC and machine learning techniques.

ACKNOWLEDGEMENTS

We acknowledge the support of Natural Sciences and Engineering Research Council of Canada (NSERC) grant number 06351.

REFERENCES

[1] Nuha Alshuqayran, Nour Ali, and Roger Evans. 2016. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 44–51.
[2] G. Antoniou, P. Groth, F. van Harmelen, and R. Hoekstra. 2012. *A Semantic Web Primer*. MIT Press.
[3] Wesley KG Assunção, Jacob Krüger, and Willian DF Mendonça. 2020. Variability management meets microservices: six challenges of re-engineering microservice-based webshops. In *Proceedings of the SPLC (A)*. 22.1–22.6.
[4] Benjamin Benni, Sébastien Mosser, Jean-Philippe Caissy, and Yann-Gaël Guéhéneuc. 2020. Can microservice-based online-retailers be used as an SPL? a study of six reference architectures. In *Proceedings of the SPLC (A)*. 24.1–24.6.
[5] Domenik Bork and Hans-Georg Fill. 2014. Formal aspects of enterprise modeling methods: a comparison framework. In *2014 47th Hawaii International Conference on System Sciences*. IEEE, 3400–3409.
[6] Antonio Bucchiarone, Kemal Soysal, and Claudio Guidi. 2019. A model-driven approach towards automatic migration to microservices. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 15–36.
[7] C. Davis and G. Kim. 2019. *Cloud Native Patterns: Designing Change-tolerant Software*. Manning Publications.
[8] P. Di Francesco, P. Lago, and I. Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (2019), 77–97.
[9] Zhongli Ding and Yun Peng. 2004. A probabilistic extension to ontology language OWL. In *37th Hawaii International Conference on System Sciences*.
[10] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*. Springer, 195–216.
[11] Sinan Eski and Feza Buzluca. 2018. An automatic extraction approach: Transition to microservices architecture from monolithic application. In *Proceedings of the*

19th International Conference on Agile Software Development: Companion. 1–6.
[12] Ricardo de Almeida Falbo, Fabiano Borges Ruy, Giancarlo Guizzardi, Monalessa Perini Barcellos, and João Paulo Andrade Almeida. 2014. Towards an enterprise ontology pattern language. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. 323–330.
[13] Frederik Gailly and Geert Poels. 2010. Conceptual modeling using domain ontologies: Improving the domain-specific quality of conceptual schemas. In *Proceedings of the 10th Workshop on Domain-Specific Modeling*. 1–6.
[14] M. Garriga. 2018. Towards a taxonomy of microservices architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10729 LNCS (2018), 203–218.
[15] Martin Garriga, Alan De Renzis, Ignacio Lizarralde, Andres Flores, Cristian Mateos, Alejandra Cechich, and Alejandro Zunino. 2018. A structural-semantic web service selection approach to improve retrievability of web services. *Information Systems Frontiers* 20, 6 (2018), 1319–1344.
[16] Michael Grüninger and Mark S Fox. 1995. The role of competency questions in enterprise engineering. In *Benchmarking—Theory and practice*. Springer, 22–31.
[17] Claudio Guidi, Ivan Lanese, Manuel Mazzara, and Fabrizio Montesi. 2017. Microservices: a language-based approach. In *Present and Ulterior Software Engineering*. Springer, 217–225.
[18] Giancarlo Guizzardi. 2012. Ontological foundations for conceptual modeling with applications. In *International Conference on Advanced Information Systems Engineering*. Springer, 695–696.
[19] Holger Knoche and Wilhelm Hasselbring. 2019. Drivers and barriers for microservice adoption-a survey among professionals in Germany. *EMISAJ* 14, 1 (2019), 1–35.
[20] Zhiyi Ma, Jinyang Liu, and Xiao He. 2018. An approach to modeling microservice solutions. In *International Conference on Information Science and Applications*. Springer, 533–542.
[21] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L McGuinness, Evren Sirin, and Naveen Srinivasan. 2007. Bringing semantics to web services with OWL-S. *World Wide Web* 10, 3 (2007), 243–277.
[22] Manuel Mazzara, Antonio Bucchiarone, Nicola Dragoni, and Victor Rivera. 2020. Size matters: Microservices research and applications. In *Microservices*. Springer, 29–42.
[23] Willian DF Mendonça, Wesley KG Assunção, Lucas V Estanislau, Silvia R Vergilio, and Alessandro Garcia. 2020. Towards a Microservices-Based Product Line with Multi-Objective Evolutionary Algorithms. In *2020 IEEE Congress on Evolutionary Computation*. 1–8.
[24] Brian S Mitchell and Spiros Mancoridis. 2001. Comparing the decompositions produced by software clustering algorithms using similarity measurements. In *Proceedings IEEE International Conference on Software Maintenance*. 744–753.
[25] Gabriel Morais and Mehdi Adda. 2020. OMSAC-Ontology of Microservices Architecture Concepts. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 0293–0301.
[26] Gabriel Morais, Dominik Bork, and Mehdi Adda. 2021. Companion source code repository. https://gitlab.com/galbuque/code-companion-omsac_modelling.
[27] Mark A. Musen. 2015. The protégé project: a look back and a look forward. *AI Matters* 1, 4 (2015), 4–12. http://protege.stanford.edu/
[28] Sam Newman. 2015. *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc.
[29] Mert Ozkaya and Ferhat Erata. 2020. Understanding Practitioners' challenges on software modeling: A survey. *Journal of Computer Languages* 58 (2020), 100963.
[30] Florian Rademacher, Jonas Sorgalla, Sabine Sachweh, and Albert Zündorf. 2019. Specific Model-Driven Microservice Development with Interlinked Modeling Languages. In *Proceedings SOSE*. IEEE, 57–5709.
[31] Juan Manuel Rodriguez, Marco Crasso, Alejandro Zunino, and Marcelo Campo. 2010. Improving Web Service descriptions for effective service discovery. *Science of Computer Programming* 75, 11 (2010), 1001–1021.
[32] Evren Sirin, Michael Howard Grove, Kendall Grant Clark, and Pedro Carvalho De Oliveira. 2020. System and method for providing prediction-model-based generation of a graph data model. US Patent 10,599,719.
[33] Eleni Stroulia and Yiqiao Wang. 2005. Structural and semantic matching for assessing web-service similarity. *International Journal of Cooperative Information Systems* 14, 04 (2005), 407–437.
[34] Rafika Thabet, Dominik Bork, Amine Boufaied, Elyes Lamine, Ouajdi Korbaa, and Hervé Pingaud. 2021. Risk-aware Business Process Management using Multi-View Modeling: Method and Tool. *Requirements Engineering* 26, 3 (2021), 371–397. https://doi.org/10.1007/s00766-021-00348-2
[35] Stardog Union. 2020. Stardog. https://www.stardog.com/
[36] Mike Uschold and Michael Gruninger. 1996. Ontologies: principles, methods and applications. *The Knowledge Engineering Review* 11, 2 (1996), 93–136.
[37] Michaël Verdonck, Frederik Gailly, Sergio de Cesare, and Geert Poels. 2015. Ontology-driven conceptual modeling: A systematic literature mapping and review. *Applied Ontology* 10, 3-4 (2015), 197–227.
[38] Michaël Verdonck, Frederik Gailly, Robert Pergl, Giancarlo Guizzardi, Beatriz Martins, and Oscar Pastor. 2019. Comparing traditional conceptual modeling with ontology-driven conceptual modeling: An empirical study. *Information Systems* 81 (2019), 92–103.
[39] Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. 2019. Informed Machine Learning–A Taxonomy and Survey of Integrating Knowledge into Learning Systems. *arXiv* (2019), arXiv–1903.
[40] W3C OWL Working Group. 2012. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. https://www.w3.org/TR/owl2-overview/
[41] W3C SPARQL Working Group. 2013. *SPARQL 1.1 Overview*. https://www.w3.org/TR/sparql11-overview/