IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 July 2024, accepted 30 August 2024, date of publication 9 September 2024, date of current version 19 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3456119

RESEARCH ARTICLE

A Lightweight Ontology for Enterprise Architecture Mining of API Gateway Logs

CARLOS ROBERTO PINHEIRO[1,2], SÉRGIO LUÍS PROENÇA DUARTE GUERREIRO[3,4], AND HENRIQUE S. MAMEDE[2,5]

[1]Escola de Ciências e Tecnologia da, Universidade de Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal
[2]INESC TEC—Institute for Systems and Computer Engineering, Technology and Science, 4200-465 Porto, Portugal
[3]INESC-ID, 1000-029 Lisbon, Portugal
[4]Instituto Superior Técnico, University of Lisbon, 1649-004 Lisbon, Portugal
[5]Department of Science and Technology, Universidade Aberta, 1269-001 Lisbon, Portugal

Corresponding author: Carlos Roberto Pinheiro (carlos.guti@gmail.com)

This work was supported by the National Funds through the Portuguese Funding Agency, Fundação para a Ciência e a Tecnologia (FCT) (https://doi.org/10.54499/UIDP/50014/2020) under Project UIDP/50014/2020.

ABSTRACT Enterprise Architecture (EA) is defined as a set of principles, methods, and models that support the design of organizational structures, expressing the different concerns of a company and its IT landscape, including processes, services, applications, and data. One role of EA management is to automate modeling tasks and maintain up-to-date EA models while reality changes. However, EA modeling still relies primarily on manual methods. Contributing to EA modeling automation, EA Mining is an approach that uses data mining techniques for EA modeling and management. It automatically captures existing information in operational databases to generate architectural models and views. This paper presents an ontology for EA Mining that focuses on generating architectural models from API gateway log files. An ontology defines the concepts and relationships among them to uniquely describe a domain of interest and specify the meaning of the terms. API Gateways are information technology components that serve as a facade between information systems and enterprise business partners. The ontology development methodology followed the SABiO process, whereas the Unified Foundational Ontology provided the foundations of the ontology and OntoUML, the ontology modeling language. An experiment in an e-commerce application scenario was conducted to evaluate the theoretical feasibility and applicability of the ontology. Automatic semantic and syntactic validation tools and semi-structured expert interviews were used to confirm the desired ontology properties. This study aims to contribute to the evolution of the knowledge base of EA Management.

INDEX TERMS Data mining, enterprise architecture management, enterprise architecture mining, ontologies, restful API.

I. INTRODUCTION

A prominent characteristic of the modern dynamic business environment is its continuous change and rapid pace at which companies must adapt to new challenges and scenarios in which new threats and opportunities are constantly arising. A well-defined Enterprise Architecture (EA) model aids in planning and maintaining the enterprise environment under control, guiding strategic decisions, aligning business processes, and integrating information systems (IS). However, the traditional approach to designing EA models, which relies primarily on manual methods involving expert interviews, presents difficulties in terms of agility, accuracy, and timeliness [1]. The speed we have seen in business changes implies that companies must continuously redefine their business goals, demanding that they review and adapt their business processes and evolve their IT infrastructure and Information Systems (IS). In this context, manual EA modeling incurs additional time pressure to deliver EA models in addition to being prone to errors and poor

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

understanding. Moreover, managing outdated models can lead architects to make inadequate decisions that may harm organizations [2].

The business capabilities of modern companies are highly dependent on IS and the underlying Information Technology (IT) infrastructure [3]. Thus, information system integration is essential to foster seamless enterprise business interactions, connecting various business partners, departments, functions, processes, and data. In this context, an API Gateway is an architectural component that sits between a caller and a collection of application services, allowing the collaboration of various organizational actors across the company's IT boundaries [4]. Furthermore, it may be an effective data source for mining cross-organizational interactions as it also monitors traffic on company boundaries and their information systems.

To ensure and optimize the alignment between business and IT, the Enterprise Architecture Management (EAM) creates, maintains, and analyzes models of the current state of EA, and designs the desired future state. These models cover distinct aspects that reflect the perspectives of business and IT, which enterprise architects must constantly update in response to the continuous transformations of the company and business environment. As business complexity increases, EA models grow, making it harder to maintain them because many stakeholders from various sources must contribute by providing relevant knowledge and information to the architecture [3]. In response to these challenges, Perez-Castillo et al. [1] described EA mining as the use of Data Mining (DM) techniques to obtain up-to-date EA models. However, according to Pérez-Castillo et al. [2], EA modeling is still performed with low automation. In addition, EAM literature indicates that maintaining the EA model, particularly documentation through manual activities, is one of the most significant challenges for EAM because of the cost and complexity of these activities [1], [2], [3]. Consequently, there is a latent need for new solutions to accelerate the modeling process and provide reliable data from the current architecture to assist in architectural decision-making.

Given this scenario, this paper refers to automation as the ability to execute tasks through computational resources with no or few manual interventions. Thus, this study proposes and validates an ontology for a data-mining application solution to automatically capture the current EA model based on events collected from the API Gateway logs.

The remainder of this paper presents the motivation for the research, followed by the background, methodology, description of the reference ontology, its implementation in a lightweight ontology, demonstration and validation, discussion, and a conclusion summarizing the main contributions and future work.

II. MOTIVATION

This work is part of an ongoing research aimed at applying EA mining to obtain architectural knowledge using API gateway logs as data sources. A previous publication within the context of this research presented a literature review in EA mining from application logs to automate architecture modeling, identifying some challenges of EA modeling automation [5]. This review reinforced the notion that logs recorded by enterprise applications may serve as an essential source of knowledge. Combined with data mining techniques, it may promote these logs as an enabler for faster EA modeling, reducing manuals and time-consuming tasks and supporting better decision-making for future architecture when providing information and visualizations of the current architectural condition.

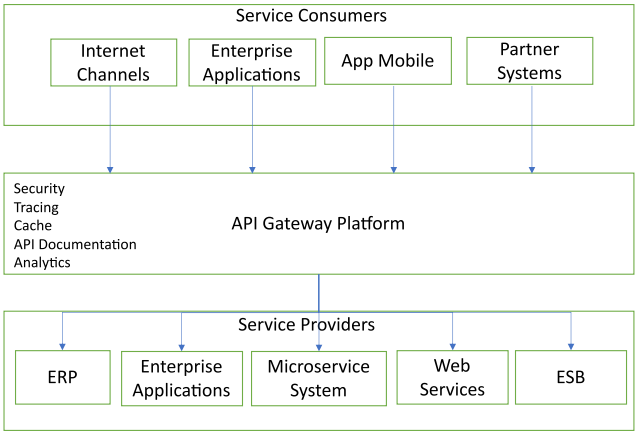

FIGURE 1. API gateway general usage (our own authorship).

There are scenarios in which multiple organizations collaborate and work together to manage shared business processes, such as when different organizations operate the same business process and share common digital experiences, knowledge, or infrastructure. Some aspects of these collaborations can be captured and recorded in event logs [6].

API Gateways are architectural components that mediate calls from clients to a collection of applications, APIs, or services by creating a façade that intercepts API requests to enforce security, validate data, transform messages, throttle traffic, route calls to the backend service or orchestrate requests between multiple backend services, and cache idempotent responses to improve performance [7]. It allows the collaboration of different organizational actors across industrial boundaries [4], [8]. Fig. 1 depicts the general schema and the central capabilities of the API Platform.

An API Management tool is a data source where cross-organizational interaction mining can be performed, because it monitors all traffic on the company's boundary and its information systems by mediating service calls among different systems of an organization. Some API platforms have built-in monitoring and analytics tools that allow the observation of events related to individual API calls and tracking their processing. However, correlating different API invocations and chaining them to identify corresponding business processes remains an open problem. Each API request differs entirely from the others. They have different data structures that may carry the same value and meaning

even with different attribute names, which causes a lack of direct correlation. Moreover, when analyzing processes and enterprise integration data without observing cached responses and blocked requests in the API Gateway, the results may be distorted because some of the calls may not reach the backend service or a transactional database. Therefore, the API gateway may play a significant role in complementing the event logs for EA mining.

In this context, an ontology is essential for clarifying the scope and objectives of solution design. Therefore, this study aims to conceptualize a reference for EA model discovery that uses API Logs as the data source, considering the need to correlate the states of multiple and different API calls, and advances toward a lightweight ontology implementation of the solution to allow obtain automatically a visualization of business process and data relation at EA level of API Interactions that will complement the existing EA mining approaches.

III. RELATED WORK

EA refers to the principles, methods, models, and frameworks applicable to the design of organizational structures. It represents different viewpoints for managing the company's varied concerns and IT landscape to align business and IT initiatives with an organization's strategic objectives, encompassing all of its business activities and capabilities, information, and technology that create the entire infrastructure and governance of an enterprise or one or more specific areas of interest within the enterprise [9], [10]. EA Management is a discipline for managing architectural development across an organization, coordinating people to achieve and maintain IS, and improving business efficiency and effectiveness [11]. This includes constructing architectural models within large enterprises and complex information system landscapes containing many data sources with relevant information for a complete EA description, which may consist of business process models, software documentation, and service documentation, among others [12].

EA Management deals with the general views of an enterprise's business activities and capabilities, information, and technology, seeking to understand and communicate how each business and IT component of an organization is related to each other and its business strategy. One of its goals is to optimize across the enterprise the often-fragmented legacy of processes and IT components into an integrated environment that is responsive to change and supportive of the delivery of the business strategy [10, Sec. 1.1], [13, Sec. 1.3]

The Open Group Architecture Framework (TOGAF) defines a framework and methodology for building EA by identifying relevant building blocks in four domains: business, data, applications, and technological architecture [10]. Winter and Fischer [14] extended the TOGAF domain layers by adding two layers: integration and process architectures. The integration layer describes the possible interfaces between the EA and specialized architectures. This extension was used as the reference. The domain structure is described as follows:

- **Business Architecture**: The Business architecture represents the fundamental organization of an enterprise from a business strategy perspective. This includes product and service categories and groups.
- **Process Architecture**: Process architecture represents the fundamental organization of service development, creation, and distribution in the enterprise context.
- **Integration Architecture**: The integration architecture represents the fundamental organization of information system components in the enterprise context. The design and evolution principles for this layer focus on the agility, cost efficiency, integration, and service speed. Aggregating dependencies and data flows between applications or application components belong to the EA. Detailed interface descriptions for data exchange, such as remote procedure calls, should be maintained using the integration repositories of Enterprise Application Integration (EAI) tools.
- **Software Architecture**: The Software architecture represents the fundamental organization of software artifacts. Detailed descriptions of the data objects are not essential for EA purposes and should be managed using a data modeling tool. In addition, the EA does not cover the structural and behavioral aspects of single software components, which should be managed using software design tools.
- **Technology Architecture**: The Technology architecture represents the fundamental organization of computing components, hardware, and networks. The detailed specifications of the IT components (i.e., hardware units) are irrelevant to the EA. Asset management tools support metadata management, and appropriate interfaces must maintain consistency between different repositories.

EA covers a wide range of issues and views of different stakeholders in these five architectural domains. This study focuses on the integration architecture. It will be helpful for industrial applications and sufficient to prove the idea, in addition to enabling it for generalization.

According to Shvaiko and Euzenat [15], ontologies typically describe concepts and the relationships between them, thereby providing constraints on their interpretation. Using ontologies aids in focusing on which information is relevant to the domain, avoiding irrelevant information from the beginning of domain analysis [16].

The Unified Foundational Ontology (UFO) presents a philosophically and cognitively well-founded formal foundational ontology for ontology development. A foundational ontology defines a range of top-level domain-independent ontological categories to provide a general foundation that supports domain-specific ontologies [17]. The UFO organizes concepts to describe what is real from the perspective of what humans can understand cognitively and

linguistically [17], [18]. Therefore, it focuses on modeling human knowledge of the natural world and its phenomena.

On the top level, UFO presents a fundamental distinction between two main categories: universal (type) and particular (individual) on its top classification. Universals are patterns of features that can be realized in several different particulars. By contrast, a particular is something that represents an entity that exists individually in reality and has a unique identity [19]. From this distinction, UFO presents three main structural fragments based on another essential difference between the Endurant and Perdurant. Endurants are individuals who exist in time with all their parts. They have essential and accidental properties; hence, they can qualitatively change while maintaining their numerical identity (i.e., remaining the same individual). By contrast, a Perdurant is an individual composed of temporal parts. Perdurants occur over time and accumulate temporal characteristics. They are manifestations of dispositions and only exist in the past (i.e., conversation, football game, symphony execution, birthday party, business process) [19]. The list below presents the UFO structure [19], [20].

UFO-A is an ontology of Endurants.

UFO-B is an ontological increment of UFO-A for terms related to the Perdurants.

UFO-C is an ontological increment of UFO-B for terms explicitly related to spheres of intentional and social things, including linguistic terms.

In their study on using ontological-based representations of enterprise models, Sunkle et al. [21] suggested that ontology representation facilitates EA analysis when it aggregates inference functionalities, which are also extensible for more detailed EA analysis.

Some studies have used semantic web technologies, such as RDF (Resource Description Framework) and OWL (*Web Ontology Language)* to represent reference models for EA. Jabloun et al. [22] developed an ontology for change management in enterprise information systems that identifies the relationships between system components by reducing their semantic mismatch. Allemang et al. [23] proposed the use of RDF and OWL to distribute the EA-structured information. They also implemented a web-based system for managing EA Reference Models based on ontology and their reference model. Silva et al. [24] proposed the use of ontology and computational inference features of ontologies to analyze the EA model evolution specified in OWL-DL. Ontology reasoners are used for relation checking, consistency checking, dependency inferring, and completeness checking, which help analyze EA model changes under a specific IT project and their impact on enterprise transformation.

Other studies have focused on the high-level ontological conceptualization of EA. Guédria et al. [25] proposed a set of concepts to identify critical aspects of interoperability and EA and their associations, resulting in a conceptual reference model for integrated EA interoperability based on ArchiMate [26]. Hinkelmann et al. [27] used an ontology representation of EA in ArchiMate to link an enterprise ontology with operational databases distributed over several information systems to ensure the continuous alignment of the information architecture with the business requirements. Hinkelmann et al. [28] developed an approach to identify dependencies among architectural models integrated through information from the databases of different enterprise systems, such as enterprise resource planning (ERP), customer relationship management (CRM), document management systems (DMS), and content management systems (CMS).

Kopp and Orlovskyi [29] developed an approach for web mining to extract a model of the EA landscape from organizational websites. It is an EA mining approach that focuses on detecting business activities by mining hyperlinks in enterprise webpage content. It also uses NPL (Natural Language Processing) techniques to detect business activities and produce a human-readable EA model of these business activities, contributing to the enterprise architectural landscape. Kang et al. [30] built a model based on WordNet that helps define ontologically relevant business terms for EA.

The related works presented here demonstrate how ontologies have been applied to EA management, summarizing some approaches and tools from references to operational ontologies to provide an initial view of ontology applications in the context of this research.

DM is an emerging field that has received increasing attention over the last two decades, with many studies seeking to use a wide variety of applications to discover trends and patterns in vast datasets [31]. Association Rule Mining (ARM) is an unsupervised Data Mining (DM) technique that attempts to identify correlations among items in a database by extracting interesting associations and frequent patterns from itemsets in a transactional database or other data repositories [32]. In general, ARM approaches use support-confidence frameworks. Support (sup) measures the frequency of an item and expresses the popularity of the item set. Confidence (conf) calculates the probability of an item's occurrence and characterizes the strength of a rule [33]. The ARM technique can identify conditions (or antecedents) and results (consequences) that have a conditional connection in a transaction, as defined below [32, p. 3].

A = Antecedent, C = Consequent, T = Transaction

$A \rightarrow C$, "If an event occurs, then an outcome event will happen."

A and C are different sets, thus $A \cap C = \emptyset$

$A \subset T$, then, it is expected that $C \subset T$, "Despite the condition A and the consequent C being different sets, both are contained in the same uniquely identified transaction T."

EAM provides decision support based on organization-wide models. However, creating these models is complex because of the multiple aspects of the organization that must be considered. Thus, it is essential to automate EA modeling because of the harmful effects of decision-making based on outdated architecture models, as manual modeling

is time-consuming, requires specialized skills, and is prone to errors and misunderstandings [2], [34], [35]. Nevertheless, low automation of EA model creation and maintenance is one of the most critical challenges in EA management [1].

Automatic architecture modeling is the capability to generate architectural models directly from data and events captured from various informalized data sources. Several investigations have studied and developed techniques to automate EA modeling to reduce the effort and time in EAM modeling, lead to a better return on investment in EAM, and support better decision making for future architecture.

The estate of the art review has yet to find studies that explicitly address the gathering and usage of API Gateway data for mining and discovering EA models [5]. However, some studies indicate that automated modeling is not an all-or-nothing proposition [34]. This reinforces that logs recorded by enterprise applications may serve as a relevant source of knowledge and contribute to raising the level of automation of EA modeling by providing information related to dynamic information to EA models. Combined with mining techniques, this may promote these logs as enablers for agile EA modeling [1].

IV. METHODOLOGY

SABiO (Systematic Approach to Building Ontologies) describes a method to design ontologies at high-level conceptual references and build ontologies at a more concrete and fully operational level [36]. This ontology development method was used for the solution design in this study. This process encompasses five steps, as illustrated in Fig. 2.

The first two phases of the process focus on building a reference ontology, whereas the remaining three focus on implementing the operational ontologies.

OntoUML provided the ontology modeling language for the reference ontology. It is an ontology language based on UML class diagram fragments that provides an ontology-driven conceptual modeling language for complex domains, whose stereotypes reflect the top ontology UFO [18], [37]. It uses the Visual Paradigm Community Edition v17.1[1] with the OntoUML Plugin for Visual Paradigm[2] as a modeling tool. This tool set supports the theoretical reference for Unified Foundational Ontology (UFO), which is the foundation ontology for OntoUML. It provides model check validators that ensure ontology rules and help detect errors and frequent ontology antipatterns [38]. Furthermore, because the proposed ontology targets the building of EA view models, the concepts were mapped to ArchiMate [26].

V. THE REFERENCE ONTOLOGY

A. PURPOSE IDENTIFICATION AND REQUIREMENTS

More specifically, the proposed ontology should allow building: (i) a business process viewpoint based on the API invocation sequence, and (ii) an architectural view of the relationship of the data objects used through the API domains.

[1]https://www.visual-paradigm.com/editions/community/

[2]https://github.com/OntoUML/ontouml-vp-plugin/

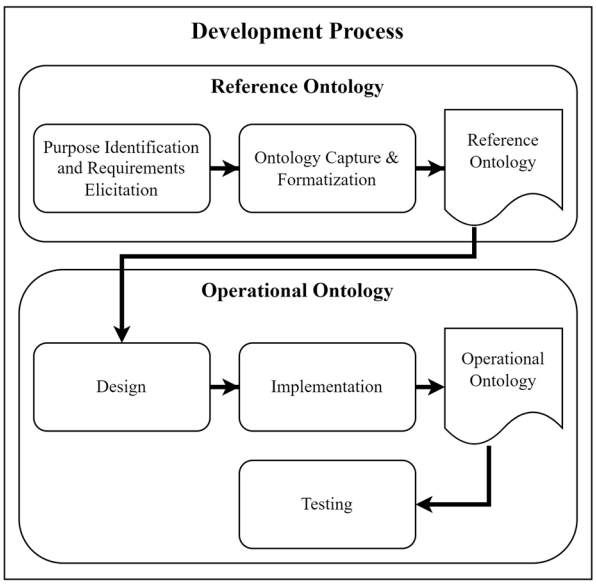

FIGURE 2. SABiO's ontology development process (Adapted from Falbo, 2014).

B. ONTOLOGY CAPTURE AND FORMALIZATION

The ontology is composed of three parts. The first is related to the core ontology concepts represented in the core package, encompassing API-related concepts. The other two parts are extensions that allow modeling a specific data-mining process from the core concepts to build two specific architectural views: process architecture and data architecture views.

1) CORE CONCEPTS

Core Concepts (package core) comprise API-related concepts extracted directly from an API Gateway log and provide structure and data to support the other two packages or extensions of the ontology. Fig. 3 shows the core ontology model of the ontology.

The event API Call is the starting point of ontology. It represents and characterizes each API call as a structural conceptual event that occurs in a specific slice of time, defined by the beginning and end times. It is affordable to reflect the properties of a temporal structure in which the beginning and end of an event define its borders [39]. API Call has the following attributes:

- **request_time**: This timestamp field, stereotyped as ≪begin≫, represents the moment the API Gateway receives an API request, starting the API processing.
- **response_time**: This timestamp field, stereotyped as ≪end≫, represents the moment the API Gateway sends the result of the processing back to the caller, marking the end of the API interaction.

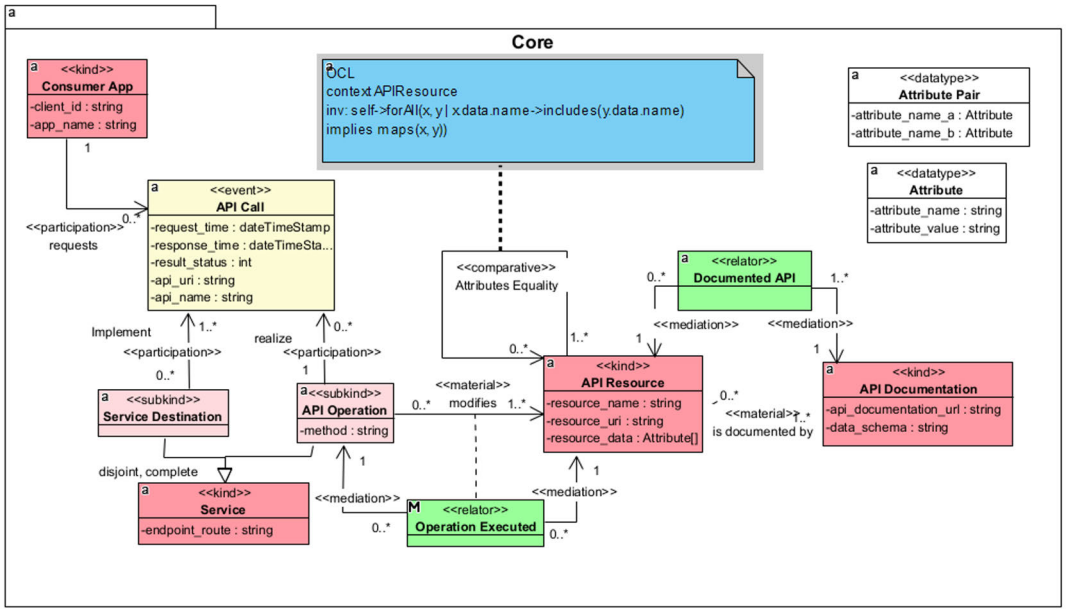

FIGURE 3. Ontology core (our authorship).

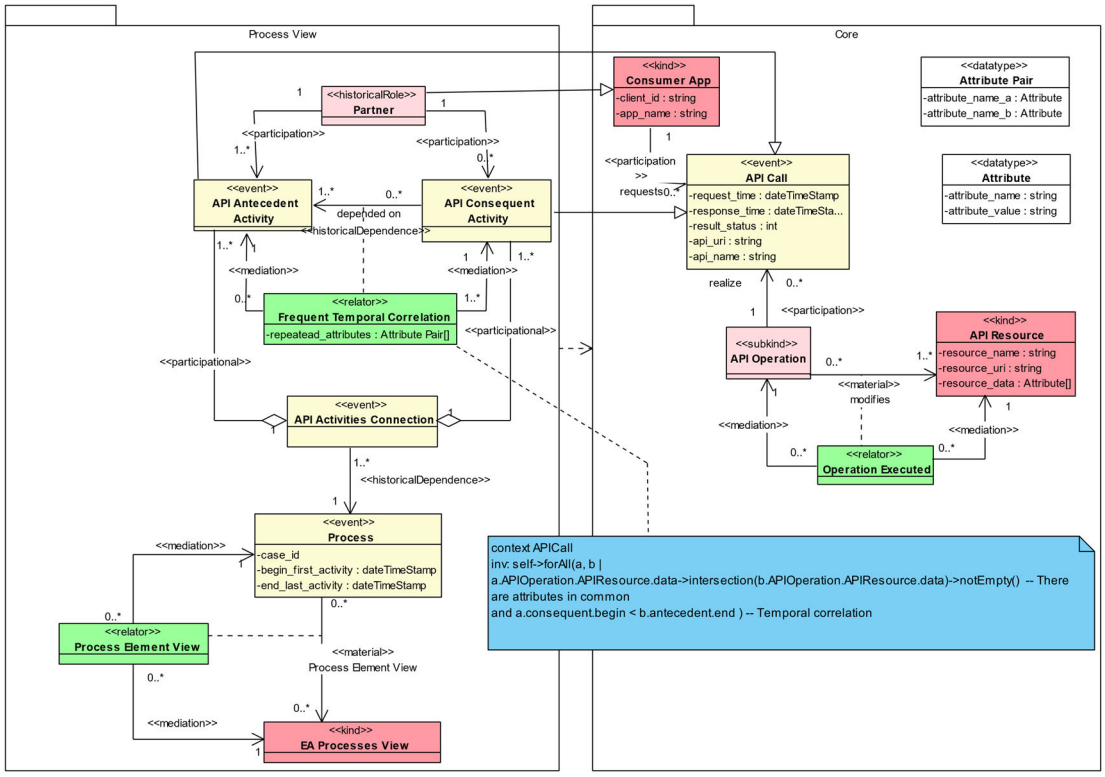

FIGURE 4. Process view mining concepts (our authorship).

- **result_status**: The result of the API call. It indicates a successful or unsuccessful return to the caller.
- **api_name**: is the API name.
- **api_uri:** is the internet address of the API resource called.
- **resource_name**: is the name of the API resource called.

Consumer APP represents applications that consume APIs identified through the attribute client_id. An event API Call is created when a consumer makes an API call. An API Call executes one API Operation, querying, aggregating,

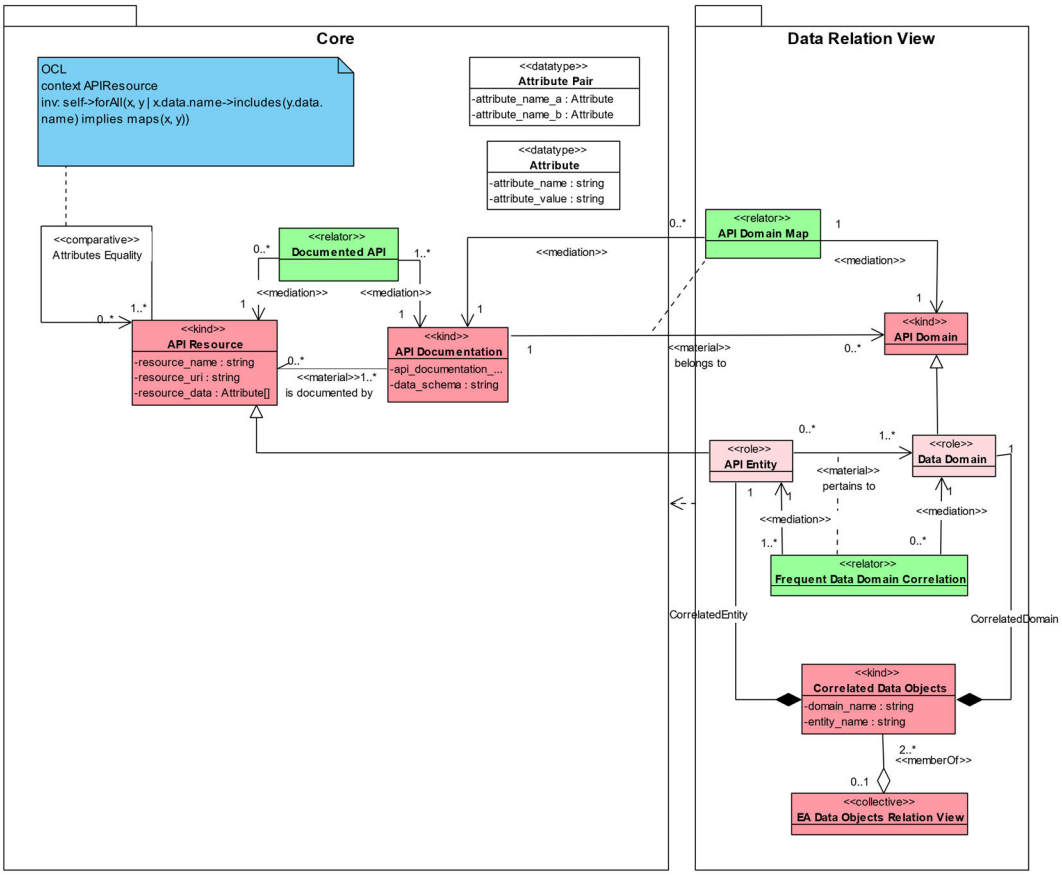

FIGURE 5. Data relation view conceptualization (our own authorship).

or modifying API Resources. URL (endpoint_route) and HTTP methods, such as GET, POST, PUT, and DELETE, characterize the operations that create, change, or delete an API Resource. An API Call may have one or more service destinations, which are services provided by backend applications or other service providers. API Operation and Service Destination extends service that provides its attribute endpoint_route. Thus, a service may be an API operation, API calls from a consumer app, or an API service destination, which are backend services called by one or more API operations. API Resource represents the data entities handled through the API operations. The attribute resource_data corresponds to the data structure of these entities. An API Resource is identified from API request payloads that share a set of attribute names to avoid duplication due to minor differences in the data structures. The relator operation denounces API Operation execution on an API Resource. The documented API relator links an API Resource to its API documentation, which is usually a swagger in a developer portal that describes the API resource and its data schema.

2) PROCESS VIEW CONCEPTS

The Process View package aims to describe concepts to extract a view of the processes supported by the APIs on API Gateway. From this viewpoint, each API Call is a process event classified and specialized as an antecedent or consequent activity by applying temporal data mining techniques to extract activity correlations from API Calls as a sequence of process events grouped in API Activity Connection that creates a chain of activities. The program then identifies and groups the processes in order to build an EA Process View. Fig. 4 depicts the ontology concepts used to build an EA process view by mining the data from the APIs. Regarding the core package representation in this figure, only the concepts that contribute to the EA process view are illustrated.

The event API Activity Connection identifies and aggregates two candidate activities, considering the same partner, the resource id present in the URI, and other shared and repeated attribute values stored in the attribute repeated_attributes in the relator Frequent Temporal Correlation. The consumer app plays the role of a partner when interacting with business processes. Thus, through temporal ARM, it is possible to identify the sequence candidate, assign a process case id, and identify and build individual instances for each process, which is a long-running event composed of a chain of API activity connections, for which is given a process identification stored in the case_id attribute of the process instance.

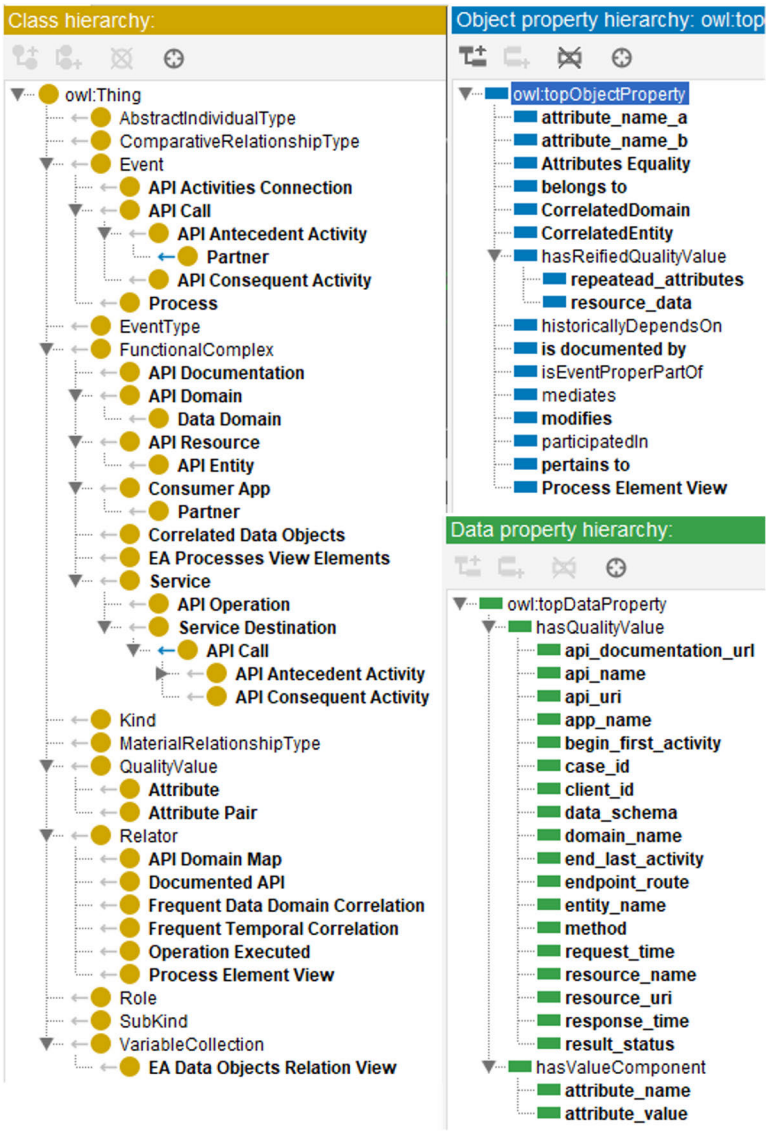

FIGURE 6. OWL structure (our own authorship).

The event process groups the activity connection chain bounded by the first antecedent activity and the last consequent activity. Then, the relator "Process View Element" correlates each process to build the EA process view, which represents the group of processes at a high level and identifies the most frequent chain of events.

3) DATA RELATION VIEW CONCEPTS

The Data Relation View package describes the concept of building a map of data objects and their relations grouped into API Domains. For this, the Frequent Data Domain Correlation identifies Data Elements within the data structures of the API Resource as API Entities. It classifies the entities related to a Data Domain that represents a logic group identified by the relator API Domain. It encompasses different entities, based on the API route and its linked documentation. The entities are then identified and grouped into context domains. The data objects are correlated based on the frequency of their repeated values in the API Resource data attributes, strengthening the relationship and confidence of the data object associations. The Correlated Data Objects group the selected relations to build the visualization by the composition of these associations in the EA Data Objects Relation View. Fig. 5 depicts the concepts involved in extracting the view of the data entity relations mined from the API payloads.

VI. THE LIGHTWEIGHT ONTOLOGY

From the reference ontology previously presented, this section advances to transform the ontology into an operational ontology in the OWL format [40].

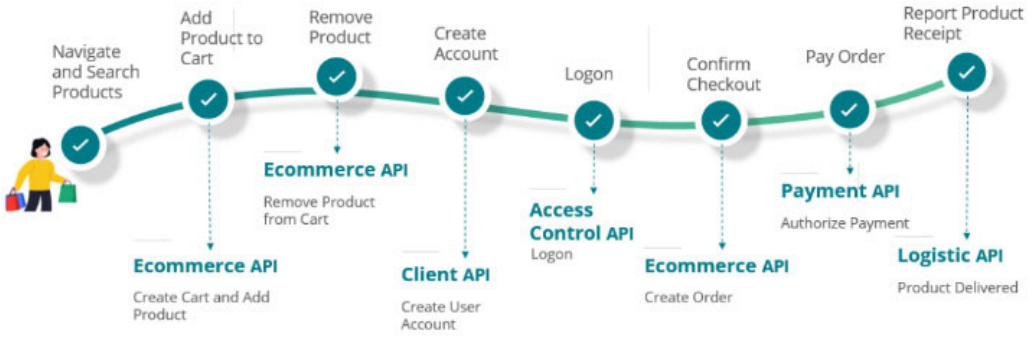

FIGURE 7. Example of e-commerce buying process (our own authorship).

Once a common conceptualization is achieved, a lightweight version of the reference ontology can be created as the objective of ontology engineering. Contrary to reference ontologies, lightweight ontologies are not focused on representation adequacy but on their design attempts to guarantee desirable computational properties of the ontologies [41], [42].

The plugin Visual Paradigm Export to the gUFO (gentle UFO) provided the first step in building lightweight ontology bases. It allows the export of OntoUML models to a gUFO ontology, supporting a lightweight UFO implementation suitable for Semantic Web OWL applications [43], [44]. In addition, ... Barcelos et al. [41] provided an SWRL reference to transform the OntoUML model into OWL to complement the operational concepts. Fig. 6 illustrates the final OWL implementation structure in Protégé, depicting its classes, object properties, and data properties.

VII. VALIDATION

A. DEMONSTRATION

This section demonstrates the applicability of the ontology proposed in this work to prove its logical feasibility and to better understand how it will support architectural view building covering the entire mining process.

This example considers a hypothetical and typical user-buying journey on an e-commerce website as a use case. Fig. 7 illustrates this journey.

First, users search for the products. Once they identified the desired product, they create a shopping cart by adding the first product. Subsequently, users can add or remove products from the created cart. After selecting products to buy, users go to the cart checkout when the application asks them to log in or register as users on the website. Once logged, they pay the invoice. Then, the products are reserved for delivery. Once the payment is approved, the system creates an order and sends it to the logistics system or a partner for delivery.

This study uses the Kong Gateway, a well-known open-source API gateway named leader in the Gartner Magic Quadrant for API Management. The first API request under analysis created a shopping cart. The log snippet below shows an example of the key attributes of an API request for creating a cart and depicts part of the log structure of an API request in the Kong Gateway.

{
"_source": {
"response": {
"status": 200,
“headers": {"date": "Thu, 08 Feb 2024 01:00:15 GMT”,
},
"body": "{\r\n \"cartid\": 1, \r\n ... }"
},
“started_at": {"$numberLong": "1707354014616” },
"consumer": {
“id": "69be440d-5a95-4eda-abd0-0924e4e2f957”,
“username": "acmeapp”,
},
"@timestamp": "2024-02-08T01:00:15.607008096Z",
"request": {
“id": "144a033ccb7cbe6ea757fb160a15e67f”,
“method": "POST”,
“URI": "/sandbox/e-commerce/v1/carts/1”,
"body": "{\r\n \"orderNumber\": 1000, \r\n ... }"
}
}
}

Table 1 lists the mapping from this log to the ontology core package.

TABLE 1. Map ontology x JSON data from Kong API gateway log.

Ontology	JSON Log
Consumer App.client_id	Consumer/id
Consumer App.app_name	consumer/username
API Call.request_time	started_at
API Call.response_time	response/headers/date
API Call.result_status	response/status
API Call.api_uri	request/uri
API Call.api_name	route/name
API Operation.method	request/method
API Operation.route	request/url
Service Destination.route	service/host + service/path
API Resource.resource_name	request/path
API Resource.resource_uri	request/uri
Resource.resource_data	request/body attributes + response/body attributes

The Consumer/id attribute identifies consumers' app individuals. The attribute username can show a friendly description of the app_name of the class Consumer App and client_id receives the attribute consumer/id.

Each API Call is an Endurant individual. It has an OWL object property participatedIn for a Consumer APP, creating a link between the caller and API Call. API Operation and Service Destination also have an object property parcipatedIn linked to each API Call.

The object property ''modifies'' stores the representation of the material relation between an API Operation and an API Resource. Considering semantic versioning in the path, the regular expression r″/v\d+/(. ∗?)/(\d+)″ allows us to obtain the resource name from the request/path. The resource.resouce_data of the conceptual ontology became the Object Property resource_data within the hasReifedQualityValue and hasValueComponent of two Data Properties: attribute_name and attribute_value, fulfilled with the name and value of the attributes present in the request and response body. The individuals in the API Documentation are fulfilled manually because the log has no reference to the location of the API documentation. The first material relation is documented by linking the relation repeated for any other individual of the API Resource with the same name.

1) THE EA PROCESS VIEWS MINING

EA process mining is a sequential correlation extraction process that sequences API Calls to build a visualization of processes supported by APIs. To achieve this, the Frequent Temporal Correlation sequences timely API Calls that share some attribute values and those with the participation of the same partner.

In this case, in addition to the relator Frequent Temporal Correlation being a logical abstraction of the historical dependency between an API antecedent activity and an API consequent activity, it persists repeated attributes between these two API Calls in the object property repeated_attributes and links them by a set attribute_name_a and attribute_name_b within the object property Attribute Pair.

The weight of the correlations is then calculated based on the percentage measure of the attributes repetition. Correlations with at least one attribute with 100% weight are considered candidates for the aggregated event API Activity Connection. This restriction implies that at least one attribute must be present in 100% of the correlations among instances to be part of the same process. It also sets the inverse object property of isEventProperPartOf for the API Antecedent Activity and API Consequent Activity.

Then, a graph is created, extracting the sequence of process events considering that the consequent activity of a process in an API activity connection is an antecedent activity in the following connection in the same process. Fig. 8 illustrates an example of the OWL graph from the process view mining.

Finally, each graph is a process instance assigned to a process case_id. As the interest of this work lies in the

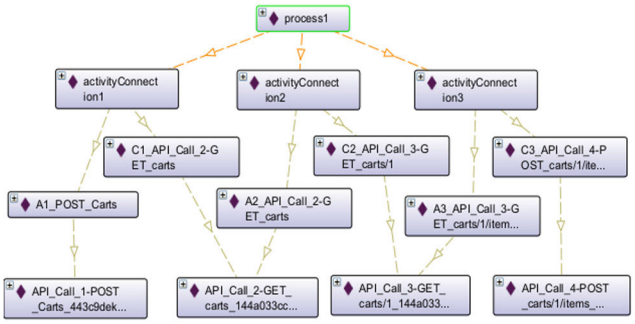

FIGURE 8. OWL of process identification.

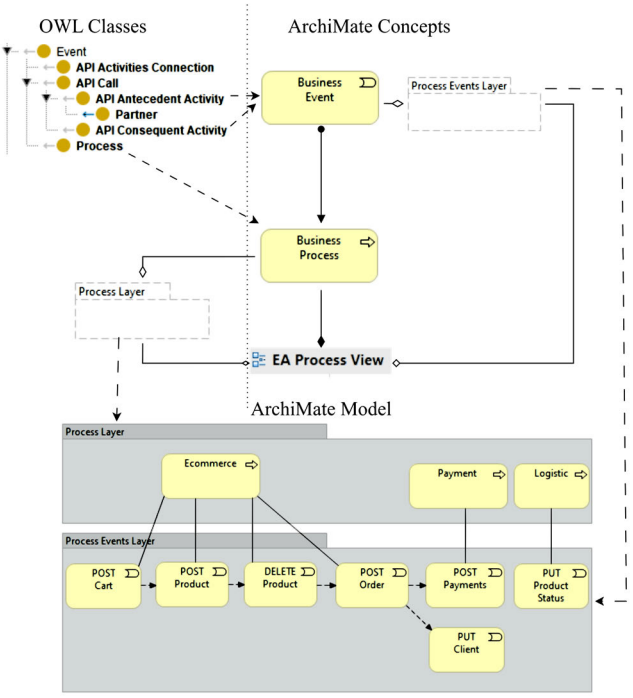

FIGURE 9. ArchiMate process view extraction from OWL (our own authorship).

EA view of the process and not in the process details, it simply relates the Process Instances to the EA Process View, which considers the similarity of graphs regarding their first and last nodes (API Activity Connection). It considers the path variation between these two nodes as internal variations of the same process, owing to the high level of EA.

The processes and events names are obtained from the semantic analysis of the API called URI. In this case, following the semantic pattern http://<domain>/<api-name>/<version>/<resource-name>, the process name comes from the <api-name> label, and the event names come from the operation_route with the verb ''POST, GET, PUT or DELETE'' and the <resource-name>.

The results are used to build ArchiMate Business Process Cooperation. Despite the differences in the individual flows and not precisely corresponding to the same graph, all paths

start with the creation of the cart and finish with the final update of the order status in the sequence depicted in Fig. 9.

2) DATA RELATION VIEW MINING

This topic demonstrates the application of DM techniques to identify frequent associations among data structures composed of Data Elements. Fig. 10 illustrates a fragment of the resulting OWL graph. The data Domain is the API name, identified by the apiName attribute from the API Call. The entities grouped into the Data Domain are API Resources with repeated-value attributes between their data structures from API Calls. It uses API Resource repeated attributes to assess the strength of the association. Only associations with very high confidence, close to 100%, should stay present in the aggregation ''Correlated Data Objects'' from which it is possible to build a visualization of entity relations for each Data Domain, such as the ArchiMate viewpoint of the buying process example illustrated in Fig. 11.

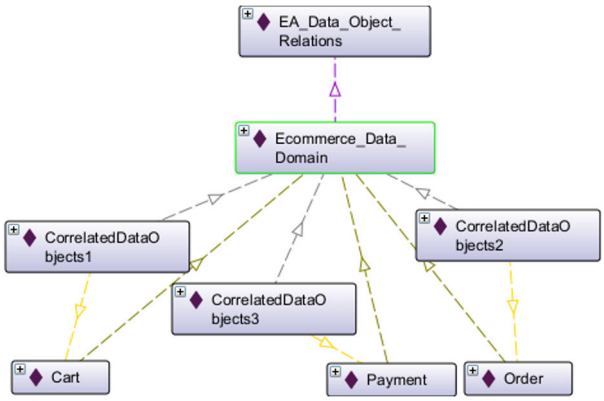

FIGURE 10. OWL of data relation identification (our own authorship).

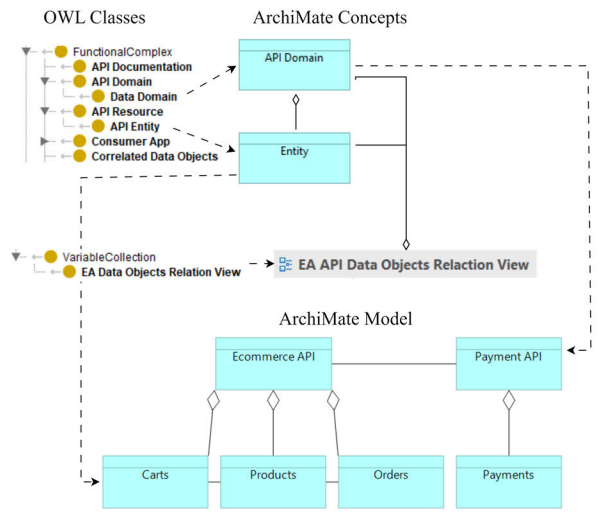

FIGURE 11. ArchiMate data relations view extraction from OWL (our own authorship).

For this view, the names of the API Domains and Entities are obtained from the semantic analysis of the API called URI. Following the semantic pattern http://<domain>/<api-name>/<version>/<resource-name>, the name of the API Domain comes from the label <api-name>, and the entity's names come from <resource-name>. The main difference with the process view extraction is that in this view, the focus is on visualizing the relationship between the data structures rather than the sequence in which the API resources are called.

B. VALIDATION

The verification and validation methods were based on the SABiO methodology [36], which prescribes evaluations from two main perspectives: (i) ontology verification and (ii) ontology validation. Ontology verification ensures that the ontology is built correctly, whereas ontology validation aims to ensure that the ontology fulfills its specific intended purpose of use. According to Falbo [36], the participation of domain experts, ontology experts, and ontology users is essential for ontology validation. Most of the verification for this ontology is performed through automated reasoners: the Chek Model of OntoUML Plugin for Visual Paradigm on reference ontology, and Hermit[3] in Protégé and owl2ready.[4] These reasoners ensure ontology consistency from conceptualization to implementation. Therefore, the verifications were automated as much as possible throughout all phases of the ontology development process. In contrast, a demonstration simulated real-world situations for validation purposes. This validation included the collaboration of an expert researcher in ontologies, and interviews with enterprise architects, API specialists, and data scientists.

1) EVALUATION ROLES

SABiO prescribes some roles in the process of ontology building: (i) domain experts, who are specialists in the ontology domain and provide the domain knowledge that is modeled and implemented in the ontology; (ii) ontology user, representing someone who intends to use the ontology for a given purpose; (iii) ontology engineer, who is responsible for the reference ontology, and thus, is responsible for the initial phases of the ontology development process; (iv) ontology designer, who is responsible for the design of an operational ontology; (v) ontology programmer, who is responsible for implementing an operational ontology; and (vi) ontology tester, who is responsible for testing an operational ontology [36]. The proposed ontology used the roles listed in Table 2 for human evaluation during the ontology development process.

Nine professionals participated in the two validation iterations. Seven specialists from six organizations participated in the final validation cycle. One bank, two universities, an API platform provider, and one IT consulting specialized in EAM and integration architecture. Fig. 12 shows the interviewee's

[3]http://www.hermit-reasoner.com/
[4]https://owlready2.readthedocs.io/en/v0.42/#

TABLE 2. Ontology evaluation roles.

Criteria	Description
Ontology Expert	A recognized expert in ontology theory and design.
Domain Expert	A person who has a background in Enterprise Architecture Management, API Management, Data Science, or Data Mining.
Ontology User	Enterprise Architects.
Tester	Domain expert and ontology users.

background. Most of the respondents had backgrounds in more than one field.

2) EVALUATION MATERIAL

The reference ontology model and a questionnaire form, with questions related to the evaluation criteria, served as the primary evaluation materials. APPENDIX B contains the link for the questionnaire as a supplementary reference. The following sections describe each evaluation criterion, related questions, and results.

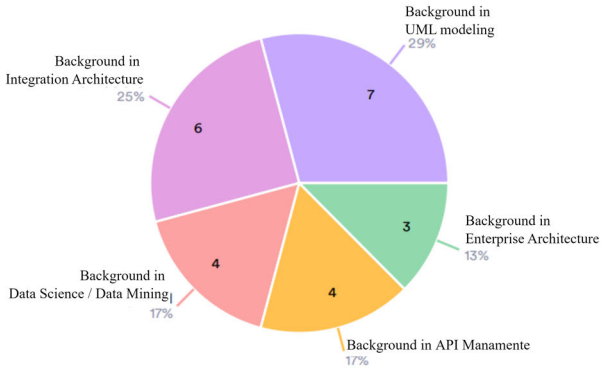

FIGURE 12. Interviewee's background distribution (our own authorship).

3) EVALUATION CRITERIA

Table 3 summarizes the evaluation criteria used to validate the quality of the proposed ontology.

a: FITNESS/COMPETENCY

This criterion checks the ontology fitness for a given task and specific domain [45]. The fitness of an ontology refers to the extent to which it attends to stakeholders' functional requirements. This should be tested and validated using competency questions. Competency Questions (CQ) identify the purpose of ontology and its intended uses, which define the scope of ontology and provide a way for its validation [36]. In this sense, the ontology application demonstration helped assess fitness by answering the following three competency questions.

CQ1: Does it identify business partners that interact with APIs?

TABLE 3. Ontology evaluation criteria.

Criteria	Description
Competency Fitness	It measures how much the ontology attends to the functional requirements and intended use.
Expressiveness	It measures the level of ontology detail to indicate if it is accurate, efficient, and appropriate to model the domain for its intended use.
Coherence	Verify if the ontology followed good practices and remove syntactic errors.
Consistency	Verify the application of ontology cleaning practices to remove semantic errors.
Completeness	Verify that there are no missing concepts from its purpose perspective.
Adaptability	It measures how well the ontology anticipates future uses.
Expandability	It assesses how easy it is to add new definitions and information to the ontology without altering the well-defined properties already provided.

CQ2: Does it map the business processes supported by APIs?

CQ3: Does it map the relations among different data objects of different APIs?

These questions were presented in the questionnaire, asking them to answer how likely they think that the proposed ontology supports the answer to these competency questions when applying the data mining techniques conceptualized to obtain architectural views. The options for each competency question were (i) Very Likely, (ii) likely, (iii) somewhat, (iv) unlikely, and (v) Very Likely.

The bar graph in Fig. 13 shows the percentage distribution of the answers. Almost all participants answered Very Likely for all three questions, and only one answered the Likely option related to competency in mapping business processes. This result confirms the fitness of the ontology.

b: EXPRESSIVENESS (EX)

Expressiveness measures the level of ontology detail to indicate whether it is accurate, efficient, and appropriate for modeling the domain for its intended use. Hence, the evaluation of the ontology expressiveness considers two perspectives. First, using a formal ontology language, such as OntoUML, ensures expressiveness regarding concepts and relational representations that other ontology builders can easily understand. Second, the final objective of ontology is to support the building of EA views. For this purpose, the following two questions were asked.

Expressiveness 1: How well does the ontology communicate the purpose and intended meanings?

Expressiveness 2: The level of ontology detail is sufficient to express all the concepts necessary for the purpose.

For Expressiveness 1, the questionnaire presented the following options: (i) it communicates well, (ii) Somewhat, and (iii) it does not communicate. For Expressiveness 2, the options were: (i) the level of detail is good, (ii) the level

of detail is sufficient, but there is some lack regarding its purpose, and (iii) the level of detail is insufficient.

Regarding the first question, 86% of the respondents agreed that it communicated its purpose and intent well, whereas 14% (one respondent) answered that it communicated somewhat. For question 2, 100% of the participants agreed that the level of detail was good. These answers indicate that ontology supports its purpose, intended meaning, and intended use in generating the proposed architectural views.

c: COHERENCE AND CONSISTENCY (CC)

These criteria aim to verify the application of ontology-cleaning practices to remove semantic errors. In this sense, McDaniel and Storey [46] discussed two types of errors in ontology design: syntax and semantic errors. They suggested using different criteria in the ontology for syntax errors, such as checking recursive definitions and missing inverse relationships. By contrast, for semantic errors, they described merging different concepts in the same class, checking missing equivalent properties and disjointness relationships, creating synonyms as new classes, and swapping intersections and unions. This evaluation considers coherent ontologies to be those without syntactic errors and consistent ontologies to be those without semantic errors. For this verification, an automatized process uses the OntoUML Plugin for Visual Paradigm, syntactical verification, model checking, detection and correction, and validation of parthood relations and ontology patterns [38]. The Hermit Reasoner in Protégé and owl2ready also helped to ensure ontology quality and consistency.

d: COMPLETENESS (CP)

This criterion verifies that no concepts are missing from a purposeful perspective. According to McDaniel and Storey [46], this refers to when all that is supposed to be in the ontology is explicitly set out or can be inferred using other definitions and axioms. Ontology completeness was evaluated using two methods. First, the OntoUML plugin for the Visual Paradigm provided validator model completeness through its Check Diagram and Check Model features. Second, the demonstration and questionnaire helped to evaluate how the ontology covered enough concepts to build all the proposed architectural views. The following question was asked in the questionnaire to evaluate this criterion.

Completeness 1: How completely does the ontology cover enough concepts to build both the process view and the data object relation view?

It has three options: (i) it covers the two views well; (ii) it partially covers the construction of the two views; and (iii) it does not cover the two views. All respondents answered that the ontology model covered all intended views well.

e: ADAPTABILITY AND EXPANDABILITY (AE)

Adaptability refers to how well an ontology anticipates its future use, and expandability is related to how easy it is to add new definitions and information to an existing definition without altering the set of well-defined properties that are already guaranteed. Both address the level of modularity of the ontology and its capability to be divided into smaller and self-contained modules, facilitating its understanding, application, reuse, evolution, and extension [46]. The ontology design followed the FAIR principles to ensure the adaptability and expandability of the ontology. These principles improve ontology findability, accessibility, interoperability, and reusability [47].

In addition, the questionnaire included the following two questions to validate and confirm ontology adaptability and expandability:

Adaptability and Expandability 1: How well is the ontology extensible to other uses and contexts?

Adaptability and Expandability 2: How much does ontology seem adaptable to anticipating future uses?

The answer options for the first question were: (i) it seems to be quite extensible, (ii) it seems to be partially extensible, and (iii) it does not seem to be extensible. For the second question, the options were (i) adaptable, (ii) somewhat adaptable, and (iii) not adaptable.

For both questions, six participants (86%) answered that they seemed extensible and adaptable. One participant answered that it seems to be partially extensible and one that it seems somewhat adaptable. In this case, the interviewee criticized the fact that, from the perspective of future use, the presented ontology does not explicitly integrate machine learning, generative AI, and other trends. Despite this criticism providing insight into evolution and future work, the results confirm that the ontology is extensible and adaptable.

VIII. DISCUSSION

Cross-organizational process models describe processes spanning two or more organizations. They also refer to process interactions or collaborations that cross the boundaries of two or more enterprise information systems of an organization. The focus of the solution in this research is not to map each business process in detail but to provide a view of the interactions in processes that cross the boundaries of the enterprise or its internal areas, capturing events from its information systems integration.

In this context, API gateways play a crucial role in complementing event logs at the EAM level. However, correlating different API calls is not trivial because each request is completely different from the others with data structures that are not directly correlated. Moreover, in general, APIs are open to consumers, who call them in a sequence that implements their processes as needed. Consequently, API requests do not have an explicit call sequence, which makes the use of the API gateway as a data source for EA mining challenging. The traceable points in an API Gateway range from when the consumer application requests an API to when the API returns a response to the caller. The entire process of the consumer application before and after the request and all backend processing between its service call and return are unknown by the API Gateway. This aspect limits the

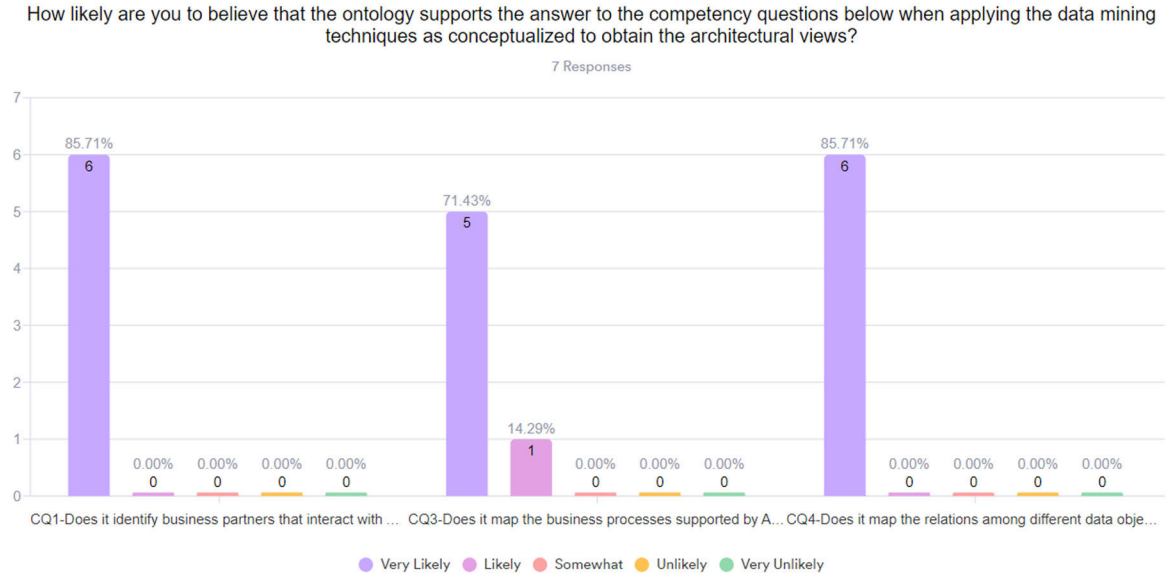

FIGURE 13. Answer distribution for the competency questions (our own authorship).

contribution of miming API Gateway logs to some specific architectural views and indicates that it is inadequate for fine-grained process details. However, it also contributes to specific architectural views at the enterprise level, such as those presented in this study.

The literature identifies some issues and challenges that must be overcome to automate EA modeling. It remains difficult to guarantee data and model accuracy, solve conflicts, and deal with missing heterogeneity, fragments, truthfulness, elements, false negatives, and false positives. Most studies have used a single data source, and even when they used multiple data sources, they reported challenges in dealing with these difficulties. Most of these studies highlight opportunities to improve the current approaches, particularly in describing how to effectively aggregate different data sources in the context of EA mining, such as an API gateway. It also reinforces that there is no one-size-fits-all solution for EA mining, and the evolution of EA mining relies on the use of various solution building blocks to compound the EAM automation landscape. Different views and viewpoints are built through different methods. Thus, EA mining from API gateway logs can deliver just some views that complement the EA models. However, it is a building block that contributes to the evolution of EA mining, as indicated in some papers, whose development has not been found in the literature.

For the business architecture domain, mining correlated events and process flows that occur among the business processes within the EA may assist in identifying hidden process flows and adapting business process views in the architecture to these actual flows. It allows quick modeling and drives corrective action courses to improve business processes. Moreover, in mapping business processes, temporal-related mining may identify patterns of business interactions from inter- or intra-organizational processes and show how partners interact with the company's businesses. Based on these data, enterprises and business architects may plan a course of action to improve the interaction between business partners and the company's services.

Search for patterns of processing inefficiencies, such as finding frequent disruptions that may indicate bad architectural design or event problems related to gaps in data and absent or noisy information. Using ARM approaches to mine API logs can contribute to demonstrating how to address stakeholder concerns by collecting indicators for these concerns from events as close as possible to real time. For instance, key EA stakeholders may be concerned about monitoring customer success. In this sense, EA mining can correlate customer behaviors with product or service details that positively or negatively impact customer interactions through enterprise digital interfaces such as enterprise APIs.

Exploring EA mining techniques in a bottom-up modeling strategy to discover enterprise architectural viewpoints can bring economic benefits to organizations, consuming less specialized time than gathering information based on interviews and inspections to discover and maintain the current architecture. It reduces the risk of errors caused by manual modeling and misunderstanding and supports better and faster architectural decision-making [1].

This study presented the conceptualization of a solution as a reference ontology, and advanced the solution to the implementation of a lightweight ontology. Its development followed the SABiO, a well-established process for ontology development. It complies with FAIR principles to ensure its findability, accessibility, interoperability, and reusability [47]. The developed ontology supports the data extraction

from APIs to automatically build two EA views in ArchMate, simplifying the AS-IS modeling of these views by eliminating the manual modeling from technical interviews and manual API inspections of the APIs published in an API Gateway.

Although it was not possible to carry out a comparative study with other ontologies, it would contribute to greater robustness of the validation, the demonstration confirmed the feasibility of the ontology, and the validation assessment proved its correction and usefulness based on ontology competence questions and quality criteria: fitness, expressiveness, coherence, consistency, completeness, adaptability, and expandability. However, from the perspective of knowledge discovery, no other works seek to capture knowledge about EA models from API log data. Conversely, from the EA perspective, the solution focuses on only two EA views, a business process view and a data view, a small set of possible EA views already well-defined in the ArchiMate language. Therefore, EA ontology is not comparable to the presented solution.

The main limitation of the present study is that the proposed method was demonstrated and validated using a controlled case. It still requires implementation in a complex case, which will be the focus of the research sequence, including advances in testing with larger volumes of data and advances in the logic and choices of ARM algorithms. The presented solution was not designed to extract architectural views from continuous data streaming. Therefore, the application of the solution requires the selection of a sample of data, e.g., within temporal limits, to obtain a snapshot of the architecture at a given moment, which may limit its application. On the other hand, EA models are relatively static and do not need minute-to-minute updates, which justifies the first focus on working with sample data instead of data streaming. Nonetheless, more frequent updates are crucial for IT infrastructure operation, business process management, and other use cases. Tracking minor changes at the detailed level of each IT component or instance of processes should consider data and processing distribution aspects that this development has not addressed and may be an opportunity for future developments.

Another aspect that limits the current proposal is that syntactic and semantic matching depend very heavily on a specific API design pattern, so the solution may not be sufficient for the comprehensive generation of EA models, especially in cases of more complex data relationships where there are semantic equivalences that the simple evaluation of API resources is not able to detect. Incorporating semantic matching techniques in future development may improve accuracy and correlation in more complex cases, with variable API design patterns or even less structured API URL patterns.

Furthermore, there is a non-explored space in the solution to apply other AI approaches, which raises the question of how the latest advances in generative AI can impact the field of EA mining and contribute to improving the presented solution.

IX. CONCLUSION AND FUTURE WORK

This study presented an ontology for an EA mining process that considers API Gateway logs as a data source. It advances from previous work [48] by (i) transforming and testing the reference ontology to an operational ontology in OWL, (ii) applying a more robust validation aggregating the participation of an ontology expert in the process, and (iii) discussing the results of API, EA, and Data Mining with specialists using semi-structured interviews.

Ontology development followed SABiO, a well-founded ontology development process that covers reference ontologies and operation ontology implementation [36]. Its accordance with FAIR principles ensures ontology findability, accessibility, interoperability, and reusability [47].

The validation confirmed the ontology competency fitness, expressiveness, coherence, consistency, completeness, adaptability, and expandability. At the same time, it provides insights into future evolution. It raised the question of how the latest advances in the use of generative AI impact the field of EA mining and can contribute to improving the solution presented so far, especially after ChatGPT was released at the end of 2022.[5]

The demonstration considered the case of REST APIs with an entity representational state in JavaScript Object Notation (JSON) format [49]. However, there are other types of APIs handled by the API Gateway, such as GraphQL and gRPC, which can be explored to extend the presented solution capabilities.

Future works should compare the performance of data-mining algorithms to contribute to the accuracy of the EA model resulting from mining processing. It must also be advanced to implement association rule mining algorithms and to apply and test solutions in one or more real cases.

APPENDIX A
gUFO ONTOLOGY FILE

The lightweight ontology in turtle format is available on https://gitlab.com/carlospinheiro/lightweight_onto_ea_mining_i3ea/-/blob/main/EA_Mining_OntoUML_Test_V1.ttl

APPENDIX B
QUESTIONNAIRE'S REPORT WITH PERSONAL DATA OBFUSCATED

https://gitlab.com/carlospinheiro/lightweight_onto_ea_mining_i3ea/-/blob/main/Last_Survey_Report.pdf

ACKNOWLEDGMENT

The authors would like to thank Dr. Pedro Paulo F. Barcelos for the collaboration in reviewing and validating the reference ontology regarding the best use of the UFO and OntoUML concepts and the transformation to the gUFO operational ontology.

[5]https://openai.com/blog/chatgpt

REFERENCES

[1] R. Perez-Castillo, F. Ruiz-Gonzalez, M. Genero, and M. Piattini, "A systematic mapping study on enterprise architecture mining," *Enterprise Inf. Syst.*, vol. 13, no. 5, pp. 675–718, May 2019, doi: 10.1080/17517575.2019.1590859.

[2] R. Pérez-Castillo, D. Caivano, F. Ruiz, and M. Piattini, "ArchiRev—Reverse engineering of information systems toward ArchiMate models. An industrial case study," *J. Softw. Evol. Process*, vol. 33, no. 2, p. e2314, Feb. 2021, doi: 10.1002/smr.2314.

[3] M. Farwick, C. M. Schweda, R. Breu, and I. Hanschke, "A situational method for semi-automated enterprise architecture documentation," *Softw. Syst. Model.*, vol. 15, no. 2, pp. 397–426, May 2016, doi: 10.1007/s10270-014-0407-3.

[4] M. M. Herterich, C. Dremel, J. Wulf, and J. vom Brocke, "The emergence of smart service ecosystems—The role of socio-technical antecedents and affordances," *Inf. Syst. J.*, vol. 33, no. 3, pp. 524–566, May 2023, doi: 10.1111/isj.12412.

[5] C. R. Pinheiro, S. Guerreiro, and H. S. Mamede, "Automation of enterprise architecture discovery based on event mining from API gateway logs: State of the art," in *Proc. IEEE 23rd Conf. Bus. Informat. (CBI)*, vol. 2, Sep. 2021, pp. 117–124, doi: 10.1109/CBI52690.2021.10062.

[6] W. van der Aalst et al., "Process mining manifesto," in *Proc. Bus. Process Manage. Workshops*, in Lecture Notes in Business Information Processing, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Berlin, Germany: Springer, 2012, pp. 169–194, doi: 10.1007/978-3-642-28108-2_19.

[7] B. De, "API management," in *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Berkeley, CA, USA: Apress, 2017, pp. 15–28, doi: 10.1007/978-1-4842-1305-6_2.

[8] C. Bonina, K. Koskinen, B. Eaton, and A. Gawer, "Digital platforms for development: Foundations and research agenda," *Inf. Syst. J.*, vol. 31, no. 6, pp. 869–902, Nov. 2021, doi: 10.1111/isj.12326.

[9] D. Greefhorst and E. Proper, "The role of enterprise architecture," in *Architecture Principles: The Cornerstones of Enterprise Architecture* (The Enterprise Engineering Series). Berlin, Germany: Springer, 2011, pp. 7–29, doi: 10.1007/978-3-642-20279-7_2.

[10] Open Group. (2022). *The Open Group, The TOGAF® Standard, 10th Edition*. Accessed: Dec. 31, 2023. [Online]. Available: https://publications.opengroup.org/c220

[11] S. Weiss, S. Aier, and R. Winter, "Institutionalization and the effectiveness of enterprise architecture management," in *Proc. ICIS*, Dec. 2013. [Online]. Available: https://aisel.aisnet.org/icis2013/proceedings/GovernanceManagement/9

[12] W. Chen, C. Hess, M. Langermeier, J. Stuelpnagel, and P. Diefenthaler, "Semantic enterprise architecture management," in *Proc. 15th Int. Conf. Enterprise Inf. Syst.*, Dec. 2023, pp. 318–325. Accessed: Dec. 3, 2023. [Online]. Available: https://www.scitepress.org/Link.aspx?doi=10.5220/0004445003180325

[13] Open Group. (2018). *The TOGAF® Standard, Version 9.2*. [Online]. Available: https://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html

[14] R. Winter and R. Fischer, "Essential layers, artifacts, and dependencies of enterprise architecture," in *Proc. 10th IEEE Int. Enterprise Distrib. Object Comput. Conf. Workshops (EDOCW)*, Oct. 2006, p. 30, doi: 10.1109/EDOCW.2006.33.

[15] P. Shvaiko and J. Euzenat, "Ten challenges for ontology matching," in *On the Move to Meaningful Internet Systems: OTM* (Lecture Notes in Computer Science), R. Meersman and Z. Tari, Eds., Berlin, Germany: Springer, 2008, pp. 1164–1182, doi: 10.1007/978-3-540-88873-4_18.

[16] L. P. Branquinho, M. B. Almeida, and R. M. A. Baracho, "Ontologies in support of data mining based on associated rules: A case study in a diagnostic medicine company," in *Proc. CEUR Workshop*, vol. 1442, 2015, p. 6.

[17] G. Guizzardi and G. Wagner, "Towards ontological foundations for agent modelling concepts using the unified fundational ontology (UFO)," in *Agent-Oriented Information Systems II* (Lecture Notes in Computer Science), P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, and M. Winikoff, Eds., Berlin, Germany: Springer, 2005, pp. 110–124, doi: 10.1007/11426714_8.

[18] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, and T. P. Sales, "UFO: Unified foundational ontology," *Appl. Ontol.*, vol. 17, no. 1, pp. 167–210, Mar. 2022, doi: 10.3233/ao-210256.

[19] G. Guizzardi. (Oct. 2005). *Ontological Foundations for Structural Conceptual Models*. Accessed: Jan. 21, 2023. [Online]. Available: https://research.utwente.nl/en/publications/ontological-foundations-for-structural-conceptual-models

[20] G. Guizzardi, R. Falbo, and R. S. S. Guizzardi, "Grounding software domain ontologies in the unified foundational ontology (UFO): The case of the ODE software process ontology," in *Proc. Conferencia Iberoamericana de Softw. Eng.*, 2008, pp. 127–140.

[21] S. Sunkle, V. Kulkarni, and S. Roychoudhury, "Analyzing enterprise models using enterprise architecture-based ontology," in *Model-Driven Engineering Languages and Systems* (Lecture Notes in Computer Science), A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. Clarke, Eds., Berlin, Germany: Springer, 2013, pp. 622–638, doi: 10.1007/978-3-642-41533-3_38.

[22] M. Jabloun, Y. Sayeb, and H. Ben Ghezala, "Enterprise ontology oriented competence: A support for enterprise architecture," in *Proc. 3rd Int. Symp. ISKO-Maghreb*, Nov. 2013, pp. 1–8, doi: 10.1109/ISKO-Maghreb.2013.6728115.

[23] D. Allemang, I. Polikoff, and R. Hodgson, "Enterprise architecture reference modeling in OWL/RDF," in *The Semantic Web—ISWC* (Lecture Notes in Computer Science), Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, Eds., Berlin, Germany: Springer, 2005, pp. 844–857, doi: 10.1007/11574620_60.

[24] N. Silva, M. Mira da Silva, and P. M. M. V. A. d. Sousa, "Modelling the evolution of enterprise architectures using ontologies," in *Proc. IEEE 19th Conf. Bus. Informat. (CBI)*, vol. 1, Jul. 2017, pp. 79–88, doi: 10.1109/CBI.2017.17.

[25] W. Guédria, K. Gaaloul, H. A. Proper, and Y. Naudet, "Research methodology for enterprise interoperability architecture approach," in *Proc. Adv. Inf. Syst. Eng. Workshops*, in Lecture Notes in Business Information Processing, X. Franch and P. Soffer, Eds. Berlin, Germany: Springer, 2013, pp. 16–29, doi: 10.1007/978-3-642-38490-5_2.

[26] Open Group. *ArchiMate® 3.1 Specification*. Accessed: Apr. 15, 2022. [Online]. Available: https://pubs.opengroup.org/architecture/archimate3-doc/

[27] K. Hinkelmann, M. Maise, and B. Thönssen, "Connecting enterprise architecture and information objects using an enterprise ontology," in *Proc. 1st Int. Conf. Enterprise Syst. (ES)*, Nov. 2013, pp. 1–11, doi: 10.1109/ES.2013.6690088.

[28] K. Hinkelmann, E. Merelli, and B. Thönssen, "The role of content and context in enterprise repositories," *Framework*, vol. 18, p. 10, Jan. 2010.

[29] A. Kopp and D. Orlovskyi, "Towards the enterprise architecture Web mining approach and software tool," in *Proc. Inf. Technol. Implement. CEUR Workshop*, vol. 3347, Kyiv, Ukraine, 2022, pp. 256–268. [Online]. Available: https://ceur-ws.org/Vol-3347/Paper_22.pdf

[30] D. Kang, J. Lee, S. Choi, and K. Kim, "An ontology-based enterprise architecture," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1456–1464, Mar. 2010, doi: 10.1016/j.eswa.2009.06.073.

[31] D. Menaga and S. Saravanan, "GA-PPARM: Constraint-based objective function and genetic algorithm for privacy preserved association rule mining," *Evol. Intell.*, vol. 15, no. 2, pp. 1487–1498, Apr. 2021, doi: 10.1007/s12065-021-00576-z.

[32] X. Liu, X. Niu, and P. Fournier-Viger, "Fast top-K association rule mining using rule generation property pruning," *Int. J. Speech Technol.*, vol. 51, no. 4, pp. 2077–2093, Apr. 2021, doi: 10.1007/s10489-020-01994-9.

[33] S. Datta and K. Mali, "Significant association rule mining with high associability," in *Proc. 5th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, May 2021, pp. 1159–1164, doi: 10.1109/ICICCS51141.2021.9432237.

[34] P. Johnson, M. Ekstedt, and R. Lagerstrom, "Automatic probabilistic enterprise IT architecture modeling: A dynamic Bayesian networks approach," in *Proc. IEEE 20th Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Sep. 2016, pp. 1–8, doi: 10.1109/EDOCW.2016.7584351.

[35] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke, "Modeling and analyzing systems-of-systems in the multi-attribute prediction language (MAPL)," in *Proc. IEEE/ACM 4th Int. Workshop Softw. Eng. Systems-of-Systems (SESoS)*. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 1–7, doi: 10.1145/2897829.2897830.

[36] R de Almeida Falbo, "SABiO: Systematic approach for building ontologies," in *Proc. Onto. Com/Odise@Fois*, 2014.

[37] G. Guizzardi, G. Wagner, J. P. A. Almeida, and R. S. S. Guizzardi, "Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story," *Appl. Ontol.*, vol. 10, nos. 3–4, pp. 259–271, Dec. 2015, doi: 10.3233/ao-150157.

[38] C. M. Fonseca, G. Guizzardi, J. P. A. Almeida, T. P. Sales, and D. Porello, "Incorporating types of types in ontology-driven conceptual modeling," in *Conceptual Modeling* (Lecture Notes in Computer Science), J. Ralyté, S. Chakravarthy, M. Mohania, M. A. Jeusfeld, and K. Karlapalem, Eds., Cham, Switzerland: Springer, 2022, pp. 18–34, doi: 10.1007/978-3-031-17995-2_2.

[39] J. P. A. Almeida, R. A. Falbo, and G. Guizzardi, "Events as entities in ontology-driven conceptual modeling," in *Conceptual Modeling* (Lecture Notes in Computer Science), A. H. F. Laender, B. Pernici, E.-P. Lim, and J. P. M. de Oliveira, Eds., Cham, Switzerland: Springer, 2019, pp. 469–483, doi: 10.1007/978-3-030-33223-5_39.

[40] *OWL Web Ontology Language Overview*. Accessed: Feb. 18, 2024. [Online]. Available: https://www.w3.org/TR/owl-features/

[41] P. P. F. Barcelos, V. A. dos Santos, F. B. Silva, M. E. Monteiro, and A. S. Garcia, "An automated transformation from OntoUML to OWL and SWRL," *Ontobras*, vol. 1041, pp. 130–141, Sep. 2013.

[42] G. Guizzardi, "On ontology, ontologies, conceptualizations, modeling languages, and (meta)models," in *Proc. Conf. Databases Inf. Syst. Sel. Papers 7th Int. Baltic Conf. (DB&IS)*. Amsterdam, The Netherlands: IOS Press, Jun. 2007, pp. 18–39. Accessed: Feb. 18, 2024. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2121ac26804a68b1bf4b7c1fdb5f7a6d1dfdede0#page=29

[43] J. P. A. Almeida, R. A. Falbo, G. Guizzardi, and T. P. Sales. *GUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO)*. Accessed: Oct. 14, 2023. [Online]. Available: http://purl.org/nemo/doc/gufo

[44] P. P. F. Barcelos, T. P. Sales, E. Romanenko, J. P. A. Almeida, G. Engelberg, D. Klein, and G. Guizzardi, "Inferring ontological categories of OWL classes using foundational rules," in *Formal Ontology in Information Systems*. Amsterdam, The Netherlands: IOS Press, 2023, pp. 109–124, doi: 10.3233/FAIA231122.

[45] O. Tatarintseva, V. Ermolayev, B. Keller, and W.-E. Matzke, "Quantifying ontology fitness in OntoElect using saturation- and vote-based metrics," in *Information and Communication Technologies in Education, Research, and Industrial Applications* (Communications in Computer and Information Science), V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky, and G. Zholtkevych, Eds., Cham, Switzerland: Springer, 2013, pp. 136–162, doi: 10.1007/978-3-319-03998-5_8.

[46] M. McDaniel and V. C. Storey, "Evaluating domain ontologies: Clarification, classification, and challenges," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–44, Jul. 2020, doi: 10.1145/3329124.

[47] P. P. F. Barcelos, T. P. Sales, M. Fumagalli, C. M. Fonseca, I. V. Sousa, E. Romanenko, J. Kritz, and G. Guizzardi, "A FAIR model catalog for ontology-driven conceptual modeling research," in *Conceptual Modeling* (Lecture Notes in Computer Science), J. Ralyté, S. Chakravarthy, M. Mohania, M. A. Jeusfeld, and K. Karlapalem, Eds., Cham, Switzerland: Springer, 2022, pp. 3–17, doi: 10.1007/978-3-031-17995-2_1.

[48] C. Pinheiro, S. Guerreiro, and H. Mamede, "Towards an ontology to enforce enterprise architecture mining," presented at the *Proc. 25th Int. Conf. Enterprise Inf. Syst.*, Oct. 2023. Accessed: Accessed: Oct. 8, 2023. [Online]. Available: https://www.scitepress.org/PublicationsDetail.aspx?ID=fc3DK2dsnQ0=&t=1

[49] T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, document RFC 8259, Internet Eng. Task Force, Request for Comments, Dec. 2017. [Online]. Available: 10.17487/RFC8259

CARLOS ROBERTO PINHEIRO received the M.Sc. degree in information and enterprise systems from Universidade Aberta, Portugal. He is currently pursuing the Ph.D. degree in science and web technology from Universidade de Trás-os-Montes e Alto Douro, Portugal. He joined INESC-ID, Information and Decision Support Systems Laboratory (IDSS), in 2021, and has been developing research in enterprise architecture automation since then. He has more than 20 years of experience as an IT Consultant for large companies in the oil and gas industry, banks, insurance, energy, retail, and other sectors. His research interests include enterprise architecture, data mining, and system integration.

SÉRGIO LUÍS PROENÇA DUARTE GUERREIRO received the M.Sc., Engineering, and Ph.D. degrees in information systems and computer engineering from Instituto Superior Técnico (IST), University of Lisbon (UL), Portugal, in 2012. He joined the Department of Computer Science and Engineering, IST, UL, as an Assistant Professor of information systems scientific, in January 2017, and also joined INESC-ID, Information and Decision Support Systems Laboratory (IDSS), as an Integrated Researcher. Previously, he was the Senior IT Project Manager of Sonae.com and an Assistant Professor with Lusófona University (ULHT) and Universidade da Beira Interior (UBI). He has published more than 100 scientific papers and has been involved with scientific events in modeling, architecture, ontology, and information systems. His research interests include enterprise engineering, aiming to find solutions to control business transaction operations and identify and solve run-time misalignments. It includes the specification languages of BPMN, ArchiMate, DEMO, UML, SysML, WOSL, and EPC; the mechanisms of process mining, and business analytics; and implementing solutions for governance, risk, and compliance.

HENRIQUE S. MAMEDE received the B.Sc. degree in computer science and engineering, the M.Sc. degree in informatics, and the Ph.D. degree in information systems and technologies. He is currently an Associate Professor (Habilitation) with the Department of Sciences and Technology, Portuguese Open University. He is an Invited Professor with the Information Management School, New University of Lisbon. He is a Senior Researcher with INESC TEC, Human ISE Center. He has published more than 100 scientific papers in conferences and journals. He is a specialist in the digital transformation of organizations and information systems. He has more than 30 years of experience in academia and as a consultant, having worked in management in the corporate sector.