Towards an Ontology to Enforce Enterprise Architecture Mining

Carlos Roberto Pinheiro[1,3][a], Sérgio Guerreiro[2,3][b] and Henrique São Mamede[3,4][c]
[1]*Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal*
[2]*Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal*
[3]*INESC-ID, Rua Alves Redol 9, 1000-029 Lisbon, Portugal*
[4]*INESC TEC, Department of Science and Technology, Universidade Aberta, Lisbon, Portugal*

Keywords: Enterprise Architecture Management, Enterprise Architecture Mining, Automatic Architecture Modelling, Ontology, API.

Abstract: Enterprise Architecture (EA) is a coherent set of principles, methods, and models that express the structure of an enterprise and its IT landscape. EA mining uses data mining techniques to automate EA modelling tasks. Ontologies help to define concepts and the relationships among these concepts to describe a domain of interest. This work presents an extensible ontology for EA mining focused on extracting architectural models that use logs from an API gateway as the data source. The proposed ontology was developed using the OntoUML language to ensure its quality and avoid anti-patterns through ontology rule checks. Then, a hypothesized scenario using data structures close to the real is used to simulate the ontology application and validate its theoretical feasibility as well as its contribution to the Enterprise Architecture Management field.

1 INTRODUCTION

Enterprise Architecture (EA) models represent viewpoints of different concerns of the company and its IT landscape, such as processes, services, and information systems (The Open Group, 2018). It is essential to automate the EA modelling due to the harmful effects on organizations of decision-making based on outdated architecture models, as manual modelling is prone to errors and misunderstanding, besides being time-consuming (Pérez-Castillo et al., 2021). Nevertheless, the literature indicates low automation of EA model creation and maintenance, it is one of the most critical challenges for EA management (Perez-Castillo et al., 2019).

An API Gateway is an architectural component that works as a proxy, mediating calls between API and services. It allows the collaboration of different organizational actors across industrial boundaries (Bonina et al., 2021; Herterich et al., 2022). As it monitors the traffic on the company's boundary and its information systems, it is an essential data source for mining cross-organizational interactions.

An ontology typically describes concepts, relationships between these concepts, and constraints that indicate how to interpret these relationships. Thus, it provides a taxonomy that describes a domain of interest and specifies the meaning of terms (Shvaiko & Euzenat, 2008; Trojahn et al., 2022). Ontologies are used in complex problems to separate essentials from implementation details.

This work is part of ongoing research to automatically apply Data Mining (DM) techniques to obtain EA view models. Specifically, this paper presents an ontology to support the EA mining from an API Gateways log.

In the sequence, section 2 describes the problem, section 3 presents the related works, section 4 describes the methodology, section 5 presents the proposed ontology, section 6 demonstrates its application, section 7 discusses the results, and section 8 presents a conclusion and future works.

[a] https://orcid.org/0000-0002-8687-7027
[b] https://orcid.org/0000-0002-8627-3338
[c] https://orcid.org/0000-0002-5383-9884

Pinheiro, C., Guerreiro, S. and Mamede, H.
Towards an Ontology to Enforce Enterprise Architecture Mining.
DOI: 10.5220/0012032300003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 2*, pages 660-668
ISBN: 978-989-758-648-4; ISSN: 2184-4992

2 THE PROBLEM

The API Management tool's usage as a data source for EA Mining has yet to be found in the literature. Even for other data mining uses, correlating different API calls is a significant challenge because each call is entirely different from the others and may lack a direct correlation. To ensure the feasibility of the research, clarify its scope and objectives, and explain how to build the solution, its conceptualization is necessary. Therefore, the specific problem intended to investigate in this work is conceptualizing a consistent and extendable process for EA mining, considering using data from an API Log. This conceptualization requires an ontology to drive a straightforward solution design.

3 RELATED WORKS

As Enterprise Architecture models expand, it becomes harder to keep them up-to-date. As companies continuously redefine their business goals, it requires a continuous process of reviewing and adapting their architecture (Farwick et al., 2016).

EA Mining is the usage of Data Mining techniques to get updated views of EA models (Perez-Castillo et al., 2019). Pérez-Castillo et al. (2021) developed a reverse engineering approach for AchiMate model generation based on static source code analysis, database schema and textual analysis. Simović et al. (2018) presented an approach to extract business information from enterprise application logs based on a domain-specific language definition. Antonello et al. (2021) prosed a data-driven method for identifying functional dependencies in Complex Technical Infrastructures (CTI) based on Association Rule Mining (ARM) to identify component dependencies using alarm messages collected through sensors and monitoring technologies, which allows detecting a local malfunction propagating through groups of dependent components. Alfonso-Robaina et al. (2020) analyzed the main variables in terms of causes and effects to assess the impact of multifactorial elements impregnated with uncertainty that affects the EA using fuzzy relation logic. However, their method is based on knowledge obtained by grouping and translating information given by experts into dependence rules.

Ontologies typically describe concepts and the relationships among them, providing constraints on how to interpret them (Shvaiko & Euzenat, 2008). One of the main advantages of using ontologies is excluding irrelevant information from the beginning of the analysis (Branquinho et al., 2015). OntoUML is a well-founded ontology language based on UML class diagram fragment that provides an ontology-driven conceptual modelling language for complex domains whose stereotypes reflect the top ontology UFO (Guizzardi et al., 2015, 2022). In their study on using ontological-based representations of enterprise models, Sunkle et al. (2013) suggest ontology representation facilitates EA analysis mixing representation and inference functionalities that are also extensible for more detailed EA analyses.

Some studies focus on representing reference models for EA through semantic web technologies such as RDF and OWL. Jabloun et al. (2013) developed an ontology for change management in enterprise information systems that helps identify system components and their relationships by reducing semantic mismatch among them. Allemang et al. (2005) proposed using RDF and OWL to distribute structured information to allow the reference model to be extended while keeping the consistency of those extensions. They implemented a web-based system for managing EA Reference Models based on ontology and their reference model.

Some other studies focused on using ontology for a more high-level conceptualization of EA. Based on ArchiMate (The Open Group, 2019) which is a well-known language for EA modelling, Guédria et al. (2013) proposed a set of concepts to identify critical aspects of interoperability and EA and their associations, which results in a conceptual reference model for integrated enterprise architecture interoperability.

Kang et al. (2010) built a model to ontologically define business terms based on WordNet, an ontology database to define terms in English. Hinkelmann et al. (2010) developed an approach to operationalize the dependencies identification among architectural models integrated through information from different sources, such as enterprise resource planning (ERP), customer relationship management (CRM), document management system (DMS), and content management systems (CMS). Hinkelmann et al. (2013) used ArchiMEO, an ontology representation of EA in ArchiMate, to link an enterprise ontology with operational databases distributed over several information systems that ensure continuous alignment of the information architecture to the business requirements. Silva et al. (2017) proposed using ontology and computational inferences features of ontologies to analyze the EA model evolution specified in OWL-DL. The ontology reasoners are used for relation checking, consistency checking, dependency inferring, and completeness checking, which helps to analyze EA model changes under a specific IT project and its impact on enterprise transformation.

In this section, we presented works that support the conceptual ontology to mining data from API gateway logs, such as UFO and OntoUML. Other works demonstrate the applications of ontology to solve different aspects of Enterprise Architecture modelling. Despite some promising approaches, none of these works provided a specific and extendable ontology for extracting enterprise architecture models directly from information systems logs.

4 METHODOLOGY

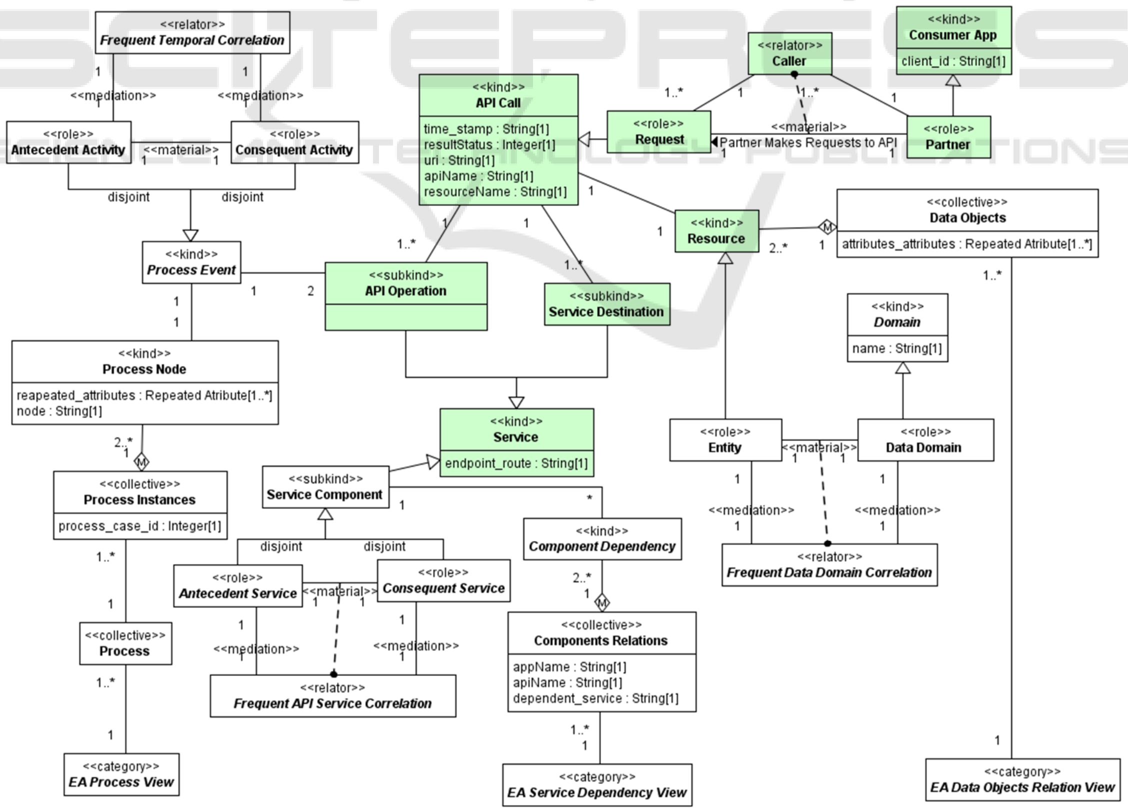
The ontology's construction followed a simplified systematic Approach to Building Ontologies (SABiO) (Falbo, 2014). It prescribes seven steps to build a fully operational web ontology. However, some steps and roles were not applied due to the objective of developing a reference ontology on a more conceptual level.

OntoUML provided the ontology modelling language and OLED (OntoUML Lightweight Editor) the modelling tool. It supports the theoretical reference for the Unified Foundational Ontology (UFO), the foundation ontology for OntoUML. It provides a wide variety of validators to ensure the ontology rules and helps to detect pitfalls and some frequent ontology anti-patterns (Guerson et al., 2015). Furthermore, as the proposed ontology targets building EA view models, the concepts are mapped to ArchiMate.

5 THE ONTOLOGY FOR EA MINING

The Application Usage viewpoint aims to describe the usage of the application by business processes and other applications (The Open Group, 2019). Figure 1 presents the proposed ontology.

The ontology is composed of four parts. The first one is related to the core concepts of ontology, which Figure 1 highlight in green. It comprises concepts that should be extracted directly from the API Gateway



<<relator>> Frequent Temporal Correlation
<<role>> Antecedent Activity
<<role>> Consequent Activity
<<material>>
<<mediation>>
disjoint
<<kind>> Process Event
<<kind>> Process Node
reapeated_attributes : Repeated Atribute[1..*]
node : String[1]
<<collective>> Process Instances
process_case_id : Integer[1]
<<collective>> Process
<<category>> EA Process View
<<kind>> API Call
time_stamp : String[1]
resultStatus : Integer[1]
uri : String[1]
apiName : String[1]
resourceName : String[1]
<<subkind>> API Operation
<<subkind>> Service Destination
<<kind>> Service
endpoint_route : String[1]
<<subkind>> Service Component
<<role>> Antecedent Service
<<role>> Consequent Service
<<relator>> Frequent API Service Correlation
<<kind>> Component Dependency
<<collective>> Components Relations
appName : String[1]
apiName : String[1]
dependent_service : String[1]
<<category>> EA Service Dependency View
<<relator>> Caller
<<kind>> Consumer App
client_id : String[1]
<<role>> Request
<<role>> Partner
Partner Makes Requests to API
<<kind>> Resource
<<collective>> Data Objects
attributes_attributes : Repeated Atribute[1..*]
<<kind>> Domain
name : String[1]
<<role>> Entity
<<role>> Data Domain
<<relator>> Frequent Data Domain Correlation
<<category>> EA Data Objects Relation View

Figure 1: Ontology for EA Mining.

Log and provide data that supports the rest of the ontology. The other three parts are extensions that allow modelling a specific data mining process from the core concepts to build a specific architecture view : (i) the EA Process view, (ii) the EA service Dependency View, and (iii) the EA Data Object Relation View.

The relator Frequent Temporal Correlation targets to build the EA Process View. To do that, it associates the Process Events and classifies each Process Event as an Antecedent Activity or Consequent Activity. Thus, creating a chain of activities by applying temporal data mining techniques to extract activity correlations from API Calls as a sequence of process events and then identifying processes and grouping them in the collective Process, from where the EA Process View is then constructed. The dependency between two Service Components is associated with Component Dependency. Then the set of this dependency is mapped in the collective Component Relations that provide de data to build the view. Similarly, to build the EA Service Dependency View, the Frequent API Service Correlation targets to identify each Services Dependency. Finally, the relator Frequent Data Domain Correlation is used to build the EA Data Objects Relation View by applying ARM techniques to discover frequent relations on the API data payloads. These data payloads are related to the API Resources and the names and values of their attributes whenever they carry on an entity data structure. These entities are grouped in the collective Data Objects that provide the data to build the EA Data Objects Relation View.

The kind API Call, on ontology core, characterizes each API call as its objects. The APIs are called by an authorized Consumer APP representing a Partner identified through the attribute client_id. The relator Caller represents Partners that make HTTP requests (role Request) to APIs.

An API Call has one or more API Operations, characterized by the method (GET, POST, PUT, DELETE) and the endpoint_route. It also has one or more Service Destination, the service called in the backend or other external or internal service provider, usually through REST APIs or SOAP Webservices. API Operation and Service Destination also extend the kind of Service identified by its endpoint_route. Thus, a Service should be an API Operation which is the API Calls from the Caller, or an API Service Destination, which are the backend services called by one API Operation on the API Gateway.

The kind Resource represents the class of objects handled through the API. It can be seen as entity data structures within the API domain. For instance, an API Call may request the creation of an object of the class client.

To build the EA Process View, the relator Frequent Temporal Correlation represents the data mining process that correlates Process Events in a temporal sequence considering each API request as an atomic event and identifying in the sequence what event represents the Antecedent Activity and what is the Consequent Activity. The candidate activities are identified and grouped considering the same Consumer App, the resource id present in URI, and other shared and repeated attributes values in the API Calls payload. This lasts a collection of attributes represented by the collective Activities Attributes. Thus, through temporal ARM, it is possible to identify the sequence candidate and assign to each sequence a case id, identifying and building individual instances for each Process, which is composed of an activities chain with antecedents and consequents under the same case_id. The Process Instances may be grouped based on the first and last activities. Thus, the inner different path sequences may be seen as variations of the same process group in the collective Process. Each Process may be categorized as a component in an EA Process View.

The Frequent Temporal Correlation associates a sequence of process events that are API Operations in the core of the ontology. These correlations are mapped as process nodes that persist the link from Antecedent Activity to Consequent Activity. Based on these Process's Nodes is built a graph of relations mapping their data values similarities and grouping in process instances. It is given a process_case_id for each process instance. The process instances with the same starting and ending events are grouped in the collective Process. Then, a view based on the Application Usage viewpoint is created in an ArchiMate with the processes mapped to Business Process into this view. It identifies the most frequent chain of events. It maps the events in this path to Business Event elements, which are linked by creating an ArchiMate Assignment Relationship between the Process and the Events.

The EA Service Dependency View maps the relations among API Services exposed in an API Gateway and the services called by these APIs, which may be a REST API, a SOAP web service, etc. To build this map, the "Frequent API Service Correlation" identify dependencies between Services. A Service may be an API Operation that represents the route that allows calling an API or a Service Destination, which is the route to a Service. These Services may also make requests to an API exposed on API Gateway. Thus, this service is called a Service Component acting as Antecedent Service and

Consequent Service that allows identifying the Service that makes calls and the Service called.

The Frequent API Service Correlation represents the ARM technique applied to identify frequent associations between APIs exposed in the API Gateway and Services API exposed by applications, which are backend services called by APIs. It correlates two Service Component identifying the Antecedent Service that frequently calls a Consequent Service. Thus, the relations between these services are individuals of the Kind Component Dependency. The collective Component Relations represent the collection that groups these dependencies in a context based on the appName and apiName, which identify the Caller and the API. Thus, an EA Service Dependency View is created for this context as an ArchiMate Application Usage viewpoint. From the attribute dependent_service in Components Relations is extracted an ArchiMate element Application Service that is grouped into the Backend Service Layer. The APIs linked to these backend services are mapped to Application Interfaces grouped into API Layer. The relation between the Application Services and Application Interfaces creates an assignment relation in the ArchiMate model. Then the same logic is applied to the attribute appName mapped to a Business Actor.

The EA Data Object Relation View aims to show a map of data object relations grouped by Domains. To identify these relations, the Frequent Data Correlation applies data mining techniques to identify Data Elements within the data structures of API payloads. The Data Element may play two roles, Entity that is identified by the API REST resource, or Data Domain that represents a logic group identified through the base path name of the API present in its route as well as in its URI. In this case, it may be essential to clarify the concept of Data Domain used in this ontology and differentiate it from the Web Domain, which is also part of the URI of API and Service routes. The last one is the web domain that allows a web resource to be recognized and reached from any computer on the Internet. The concept for this ontology is a logic groping that creates a context of data usage. Thus, this context encompasses different entities based on the API route and its linked documentation, excluding the web domain until the part of URI that identifies the different resources that an operation over API creates, modifies, deletes, and so on. Once identified the entities and groped to a context (Domain), the data object is correlated based on their dependency based on the frequency of its repeated values in its attributes.

In this part of the ontology, the Frequent Data Domain Correlation represents the ARM technique that extracts frequent data correlations based on the repeated value of its attributes. The Data Objects are directly related to the data structures of the API calls identified by the kind Resource. Thus, the Frequent Data Domain Correlation identifies the peace of data that plays the role of Entity and the peace that identifies the Data Domain. The repeated_attributes of Data Objects are used to identify the strength of the relation. Then, the Data Object classified as Entities are mapped to an application Data Object in ArchiMate. The related Domain is mapped to another ArchiMate Data Object, but this one aggregates the other one.

6 EXAMPLE OF APPLICATION

This section exemplifies the applicability of the ontology proposed in this work to prove its logical feasibility and better understand how it will support the architecture view building covering the whole Process.

The example considers a simplified and hypothetical user-buying journey on an e-commerce site. Firstly, users search for products. Once they identify the product, they put it in a shopping cart created when the first product is added. Then, users add and remove products from the cart. Once satisfied, users check out the cart when they are asked to create an account or log in. Once logged, they pay, and after the payment is approved, the e-commerce system creates an order that is sent to the logistics sector for delivery. In the end, the logistics and the user confirm the product delivery and its reception by the user.

The first API call occurs when the user adds the first product to the cart, creating the shopping cart. The key attributes this API call based on Sensedia API Gateway, which is used for through this example:

- **clientId:** identifies the Consumer App individuals.
- **apiName:** friendly name description of API
- receivedOnDate: time that the call hits the API Gateway.
- **resultStatus:** the HTTP result of API processing with success or error.
- **uri:** the internet address of the API.
- **trace:** a string with the record of some aspects of the API call. It carries its header and body payload.

An API Call should be one kind of Service that is identified by its URL route: an API Operation, which is the API Calls from the Caller, or an API Service Destination which are the backend services called by one API Operation on the API Gateway.

From "trace" string, it is possible to extract the API Operation Route and the Service Destination Route.

The operation name in API Operation identifies the operation over an API to create or change a resource. It results from string concatenation of the value of method, apiName and Resource.

Because the called URLs carry the concrete identifiers of resource instances, it was necessary to check the description of the routes in the swagger. The swagger describes some relevant attributes of the API. In the case of Sensedia's API Platform, its documentation is linked to the API Gateway.

6.1 Frequent Temporal Correlation

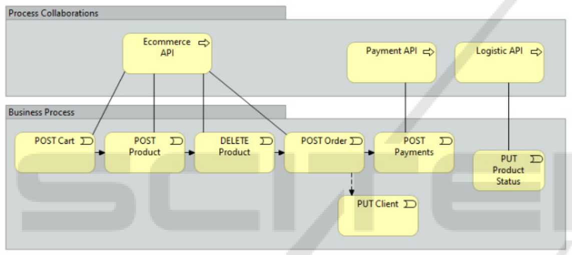


Figure 2: ArchiMate Process View.

Frequent Temporal Correlation is the sequential correlation mining extraction process that sequences timely API Calls that share some attribute values and groups those with the same Caller. The attribute names and its value are stored in the repeated_attributes of Process Node for each Process Event, combining two API Calls, the older as the Antecedent Activity and the younger as the Consequent Activity. Then the weight of the Process Event is calculated based on the percentual measure of the attributes repetition. The Process Events that do not have at least one attribute with 100% of the weight are eliminated, presuming that to make part of the same Process, at least one attribute must be present in 100% of correlations among its instances.

Then, a graph is created, extracting the sequence of process events considering that a Consequent Activity of a Process Event is an Antecedent Activity in the following Process Event. In the end, each graph is a process instance assigned a "process_case_id". As the interest of this work lies in an enterprise architecture view of the Process, and not in process details, it simply groups the Process Instances into collective Processes that consider the similarity of graphs regards its first and last node. It considers the variations path between these two nodes as internal variations of the same Process due to the high level of EA.

The resulting model is used to build an ArchiMate Business Process Cooperation viewpoint. Despite some differences in the sequences and do not correspond precisely to the same graph, all paths will start with the creation of the cart and finish with the final update of order status with the sequence depicted in Figure 2.

6.2 Frequent API Service Correlation

Frequent API Service Correlation is the data mining processing that correlates two Services. The first is an API Operation playing the Antecedent Service role that calls the Consequent Service, which is the Service Destinations of an API Call. Its relation is an instance of Service Dependency where the "api" attribute comes from the API Call "apiName", and the dependent service comes from service route. The collective stores a collection of the related mining grouped by API. Thus, it makes it possible to depict what an API and its resources call a dependent service. This view may help to get an impact analysis for an API Change.

Thus, the EA Service Dependency View may be built to show these dependencies at a high level, such as the ArchiMate Application Cooperation viewpoint depicted in Figure 3.

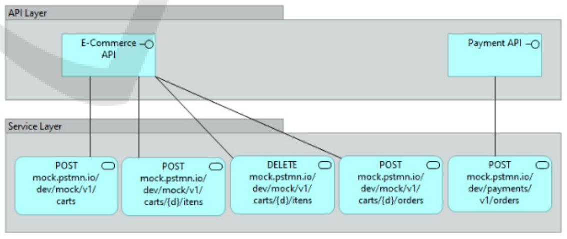

API Layer
E-Commerce API
Payment API
Service Layer
POST mock.pstmn.io/dev/mock/v1/carts
POST mock.pstmn.io/dev/mock/v1/carts/{id}/itens
DELETE mock.pstmn.io/dev/mock/v1/carts/{id}/itens
POST mock.pstmn.io/dev/mock/v1/carts/{id}/orders
POST mock.pstmn.io/dev/payments/v1/orders

Figure 3: ArchiMate Service Dependency.

6.3 Frequent Data Domain Correlation

It is the application of the Data Mining Technique to identify frequent associations among data structures composed of Data Elements. These Data elements may be Data domains or Entities. From this perspective, we consider the Data Domain as the API Name. Thus, it is filled by the API Name attribute from the API Call. The entities grouped into a Data Domain are the resources. Then the Entities are filled with the resource_name of the API. The Entity Attributes, in this case, are the repeated value attributes between the data structures. Then it uses entity attributes to assess the strength of the association. Only associations with very high confidence, close to 100%, should stay present in the collective "Data Objects Domain" from what it is possible to build a visualization of Entities relations for each data Domain, such as the ArchiMate viewpoint illustrated in Figure 4.

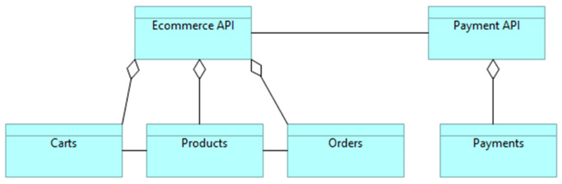


Figure 4: Data Objects API Dependency.

7 DISCUSSION

This proposal focuses on conceptualizing an EA mining extraction from a particular technology, API Gateway, which is a widely used integration tool. Even though many companies may also have been not using it, limiting the industrial application of the study, API gateway logs are vital to complement the EA mining field once it provides a kind of service facade for the information systems of an organization and its partners, capturing and recording some behaviours and data structures that may not be present in operational databases. This approach and focus have yet to be found in the literature.

Tracing a unique API request through the network and following the components triggered by it is not a big deal. However, the proposed ontology focuses on correlating different requests that initially have no physical correlations but that share logical contexts relevant to understanding a part of the EA.

The extracted business process view allows an overlook of an API relations with the execution of the main tasks of a business process. Another view allows identifying and tracking API dependencies with various operations and the backend services that support these operations. It helps impact analysis of changes in APIs or services that can impact the global EA level. Likewise, a view of data objects related to APIs and processes helps to identify the relationship between data entities and their usage through enterprise APIs. All these extracted views aim to contribute to the overall architecture. Despite having some limitations regarding building a complete EA, such as the business architecture, it will contribute with complementary views to accelerate the architecture analysis and modelling.

For this ontology, attributes with repeated values between the data structures of API calls are used to give weight to the relationship between process activities and to associate data objects. Based on this data, it is possible to apply other process mining techniques to help discover different aspects and patterns related to business processes. Another way to extend this ontology is through further analysis of this data to verify the cohesion of the data domains related to the name of the attributes between the data objects.

The proposed ontology had its consistency and quality checked by tools. It condenses three different applications following an ontological pattern for three distinct views, demonstrating that the model can be extended to build other architectural views not developed in this work and to apply other APIs integrations tools beyond API Gateway. Furthermore, it can allows instantiation and refinement into a more operational ontology, such as RDF or OWL.

The proposal's threat so far is that despite presenting a specific domain ontology, this ontology is still a relatively high conceptual reference model. Although OLED allows exporting OntoUML models to other more operational languages for ontology instantiation, the high level of detail, given the objectives of this work, may reduce the expressiveness of the ontology when applied to these more operational languages, such as OWL and RDF, requiring adaptation of the derived models.

8 CONCLUSION AND FUTURE WORK

This research aims to build a DM model to automate the modelling of the current views of EA. For that, the ontology presented in this paper focused on conceptualizing the process of building EA mining.

The application of the proposed ontology was hypothesized and exemplified as a possible approach for capturing EA model changes. It allows tracking the current EA and provides complementary views to support EAM.

It was only partially implemented in an automatized form to extract and correlate data to build the EA Process View. Much of the data provided in the example was simulated, which implies a challenge to solve in future work. Even though the simulation matches the semantics and attributes in an actual API gateway log file, it still needs to be programmed and proven in a real case.

Also, the best data mining techniques for each architectural view must be analysed to build the complete model.

ACKNOWLEDGMENT

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

REFERENCES

Alfonso-Robaina, D., Díaz-Moreno, J. C., Malleuve-Martınez, A., Medina-Moreno, J., & Rubio-Manzano, C. (2020). Modeling Enterprise Architecture and Strategic Management from Fuzzy Decision Rules. In L. T. Kóczy, J. Medina-Moreno, E. Ramírez-Poussa, & A. Šostak (Eds.), *Computational Intelligence and Mathematics for Tackling Complex Problems* (pp. 139–147). Springer International Publishing. https://doi.org/10.1007/978-3-030-16024-1_18

Allemang, D., Polikoff, I., & Hodgson, R. (2005). Enterprise Architecture Reference Modeling in OWL/RDF. In Y. Gil, E. Motta, V. R. Benjamins, & M. A. Musen (Eds.), *The Semantic Web – ISWC 2005* (pp. 844–857). Springer. https://doi.org/10.1007/11574620_60

Antonello, F., Baraldi, P., Shokry, A., Zio, E., Gentile, U., & Serio, L. (2021). Association rules extraction for the identification of functional dependencies in complex technical infrastructures. *Reliability Engineering & System Safety*, *209*, 107305. https://doi.org/10.1016/j.ress.2020.107305

Bonina, C., Koskinen, K., Eaton, B., & Gawer, A. (2021). Digital platforms for development: Foundations and research agenda. *Information Systems Journal*, *31*(6), 869–902. https://doi.org/10.1111/isj.12326

Branquinho, L. P., Almeida, M. B., & Baracho, R. M. A. (2015). Ontologies in support of data mining based on associated rules: A case study in a diagnostic medicine company. *CEUR Workshop Proc*, *1442*, 6.

Falbo, R. de A. (2014). SABiO: Systematic Approach for Building Ontologies. *Onto. Com/Odise@ Fois*.

Farwick, M., Schweda, C. M., Breu, R., & Hanschke, I. (2016). A situational method for semi-automated Enterprise Architecture Documentation. *Software & Systems Modeling*, *15*(2), 397–426. https://doi.org/10.1007/s10270-014-0407-3

Guédria, W., Gaaloul, K., Proper, H. A., & Naudet, Y. (2013). Research Methodology for Enterprise Interoperability Architecture Approach. In X. Franch & P. Soffer (Eds.), *Advanced Information Systems Engineering Workshops* (pp. 16–29). Springer. https://doi.org/10.1007/978-3-642-38490-5_2

Guerson, J., Sales, T. P., Guizzardi, G., & Almeida, J. P. A. (2015). OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies. *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, 144–147. https://doi.org/10.1109/EDOCW.2015.17

Guizzardi, G., Botti Benevides, A., Fonseca, C. M., Porello, D., Almeida, J. P. A., & Prince Sales, T. (2022). UFO: Unified Foundational Ontology. *Applied Ontology*, *17*(1), 167–210. https://doi.org/10.3233/AO-210256

Guizzardi, G., Wagner, G., Almeida, J. P. A., & Guizzardi, R. S. S. (2015). Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology*, *10*(3–4), 259–271. https://doi.org/10.3233/AO-150157

Herterich, M. M., Dremel, C., Wulf, J., & vom Brocke, J. (2022). The emergence of smart service ecosystems—The role of socio-technical antecedents and affordances. *Information Systems Journal*, *n/a*(n/a). https://doi.org/10.1111/isj.12412

Hinkelmann, K., Maise, M., & Thönssen, B. (2013). Connecting enterprise architecture and information objects using an enterprise ontology. *Proceedings of the First International Conference on Enterprise Systems: ES 2013*, 1–11. https://doi.org/10.1109/ES.2013.6690088

Hinkelmann, K., Merelli, E., & Thönssen, B. (2010). The role of content and context in enterprise repositories. *Framework*, *18*, 10.

Jabloun, M., Sayeb, Y., & Ben Ghezala, H. (2013). Enterprise ontology oriented competence: A support for enterprise architecture. *2013 3rd International Symposium ISKO-Maghreb*, 1–8. https://doi.org/10.1109/ISKO-Maghreb.2013.6728115

Kang, D., Lee, J., Choi, S., & Kim, K. (2010). An ontology-based Enterprise Architecture. *Expert Systems with Applications*, *37*(2), 1456–1464. https://doi.org/10.1016/j.eswa.2009.06.073

Pérez-Castillo, R., Caivano, D., Ruiz, F., & Piattini, M. (2021). ArchiRev—Reverse engineering of information systems toward ArchiMate models. An industrial case study. *Journal of Software: Evolution and Process*, *33*(2), e2314. https://doi.org/10.1002/smr.2314

Perez-Castillo, R., Ruiz-Gonzalez, F., Genero, M., & Piattini, M. (2019). A systematic mapping study on enterprise architecture mining. *Enterprise Information Systems*, *13*(5), 675–718. https://doi.org/10.1080/17517575.2019.1590859

Shvaiko, P., & Euzenat, J. (2008). Ten Challenges for Ontology Matching. In R. Meersman & Z. Tari (Eds.), *On the Move to Meaningful Internet Systems: OTM 2008* (pp. 1164–1182). Springer. https://doi.org/10.1007/978-3-540-88873-4_18

Silva, N., Mira da Silva, M., & Sousa, P. M. M. V. A. de. (2017). Modelling the Evolution of Enterprise Architectures Using Ontologies. *2017 IEEE 19th Conference on Business Informatics (CBI)*, *01*, 79–88. https://doi.org/10.1109/CBI.2017.17

Simović, A. P., Babarogić, S., & Pantelić, O. (2018). A domain-specific language for supporting event log extraction from ERP systems. *2018 7th International Conference on Computers Communications and

Control (ICCCC), 12–16. https://doi.org/10.1109/ICCCC.2018.8390430

Sunkle, S., Kulkarni, V., & Roychoudhury, S. (2013). Analyzing Enterprise Models Using Enterprise Architecture-Based Ontology. In A. Moreira, B. Schätz, J. Gray, A. Vallecillo, & P. Clarke (Eds.), *Model-Driven Engineering Languages and Systems* (pp. 622–638). Springer. https://doi.org/10.1007/978-3-642-41533-3_38

The Open Group. (2018). *The TOGAF® Standard, Version 9.2*. https://publications.opengroup.org/standards/togaf/c182. https://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html

The Open Group. (2019, November). *ArchiMate® 3.1 Specification*. https://pubs.opengroup.org/architecture/archimate3-doc/

Trojahn, C., Vieira, R., Schmidt, D., Pease, A., & Guizzardi, G. (2022). Foundational ontologies meet ontology matching: A survey. *Semantic Web*, *13*(4), 685–704. https://doi.org/10.3233/SW-210447