Design Approach to Platform Agnostic Service API Modeling for Interoperability of Cross-enterprise Vehicle Applications

Sangita De
Dept. of Computer Science and Engineering, Faculty of Applied Sciences
University of West Bohemia
Regensburg, Germany
sangita@students.zcu.cz
https://orcid.org/0000-0002-0741-878X

Juergen Mottok
Dept. of Electrical Engineering and Information Technology
OstBayerische Technical University(OTH)
Regensburg, Germany
juergen.mottok@oth-regensburg.de

Premek Brada
Dept. of Computer Science and Engineering, Faculty of Applied Sciences
University of West Bohemia
Pilsen, Czech Republic
brada@kiv.zcu.cz

***Abstract*—In the last few years, the collaboration of services between the service-oriented, cross-enterprise vehicle application frameworks has gradually increased to generate novel and more complicated vehicle services for the automotive industry. In these service collaboration scenarios, where heterogeneous application frameworks participate to realize complex vehicle services, a source of discord that emerges is that the service providers must always check, before the service deployment, whether the clients or service consumers on the other side of the communication link are compatible with a given service's API (Application Program Interface). While using standardized templates like ontologies for API's semantic specifications are crucial for a service discovery and semantic interoperability, nevertheless, accessing these service APIs' semantic data using a standardized and understandable syntactical specification template is also equally substantial to ease services interoperability. In fact, such complex service collaboration scenarios motivate this research work which proposes a design approach towards standardized, domain-specific, platform-agnostic semantic and syntactic specification of vehicle services API models. This paper also uses a typical vehicle domain case study to illustrate the design approach and a reference mapping between the platform-agnostic semantics specifications of a vehicle service API ontological model and its corresponding language-neutral, syntactic representation using the OpenAPI standard.**

Keywords—Semantic, service, API, synergy, RESTful, interoperability, syntax, domain, ontology, Grounding, RPC

I. Introduction

In today's era, applications in vehicle domain are implemented as multiple distributed components, and those components call each other's Application Program Interfaces (APIs) for the complete application to function. With the increase in demand of novel services in the automotive industry, automotive enterprises prefer to collaborate with other qualified third-party, cross knowledge domain partners such as Robotics, Telematics, Infotainment to provide complex automotive functions (or services), such as autonomous driving, feature upgradability, IoT (Internet of Things), etc. The services are black boxes to the requesters, which means that their source codes are not publicly available. Therefore, to make these services flexibly accessible, their APIs must be specified using standardized semantic and syntactic specifications. The incompatibility between the heterogeneous platforms specific artifacts that are part of semantic specification of various vehicle service models causes impediment in semantic interoperability between the various API models of the services. From a modeling perspective, to facilitate a holistic and meaningful data exchange between several heterogeneous vehicle services' API models in vehicle domain, it is essential to link the platform-agnostic API data at semantic level using a shared vocabulary of the domain. In fact, modeling issues like formal semantics or interoperability between heterogeneous applications within a domain motivates the development of ontology languages. Nevertheless, an emerging source of discord in this direction is accessing the ontology-based semantic data in knowledge graphs through SPARQL queries, etc. which requires prerequisite knowledge in semantic web technologies. Therefore, a possible solution to this discord, could be to ease the access of ontology-based graphs containing vehicle services' API semantic data by using a standardized, language-neutral syntax with a well-known, consistent documentation that is, understandable to most of the software developers in the vehicle application domain[6].

Most of the novel and complex automotive software systems are being built using the service-oriented architecture (SOA) and microservices paradigms. The vehicle services, accessed by client applications through their APIs, mostly follow the remote operation call (RPC) oriented style or the representational state transfer (REST) style. In fact, this is the major reason for the growing popularity of OpenAPI Specification (OAS) among automotive software developers, as its generators helps to keep the service implementation and language-neutral syntax, API documentation consistent [2]. OAS also provides an easy way to access vehicle services' API semantic data in ontology-based knowledge graphs through REST or RPC APIs. This further helps to achieve better results in terms of vehicle service invocation and discovery, thereby ease in semantic interoperability.

A. Background and Related Work

In today's world, usually, any standalone framework in the vehicle application domain cannot provide the complete spectrum of vehicle services, in such cases, to realize complex and novel vehicle services, automotive application software developers often focus on utilizing services from other qualified knowledge domain partners. With an increase in vast variety of services provided to the passengers, a lot of functionalities are now actively running on the vehicle's on-board systems. Data is what all these functions are based on, be it sensor data, user profiles, traffic broadcasts or car-to-car

2021 International Conference on Applied Electronics (AE) | 978-80-261-0973-0/21/$31.00 ©2021 10.23919/AE51540.2021.9542890

ISBN 978-80-261-0973-0,

messages from peer vehicles. The key to success in making such a complex ecosystem work however, lies in the organized and efficient access to the data [4]. In such complex service composition scenarios, for service compatibility, it has become quite common for providers to check, before each service deployment, whether the (pre-existing) clients are compatible with a given service's API, including its evolved version(s), as they may have not originally been designed and tested against it [2]. For the services in semantic web domain, authors et al. also present an approach to standardize semantic request and offer specifications that use heterogeneous domain ontologies. An ontology matchmaker aligns these ontologies and produces a relational Query View Transformation (QVTr) script which resolves the heterogeneity and eases service matchmaking [3]. The METEOR-S Web Service Annotation Framework (MWSAF) [3] is an approach to annotate WSDL (Web Service Description Language) specifications with semantics. XML schemas contained in a WSDL specification are matched with several domain ontologies. The best matching ontology is used to classify the web service.

Authors et al. [4] propose an *Automotive Ontology* at the core of a car's information system and mainly contributes towards a reference ontological model design, highlighting vital areas of automotive application domain knowledge in conjunction with reasoning. The work specifies meta information such as time, confidence, and privacy, functions can employ their own reasoning and knowledge can be shared across the boundaries of a vehicle without being dependent on a strict low-level protocol or a particular manufacturer [4]. In context of standardization of service API models, ASSAM [9] includes a WSDL annotator that automatically suggest ontology concepts to annotate each WSDL element. The annotations can be exported as OWL-S which also includes XSLT lifting and lowering transformations. Authors et al. [8] proposes adapter-based approach to convert raw data like XML (Extended Markup Language) to WSML (Web Service Modeling Language) to use WSMO (Web Service Modeling Ontology) for modeling software services in a standardized and generic manner for services interoperability. Authors et al. [2] proposes a method for creating platform-agnostic service API semantic and syntactic representations like OpenAPI, RAML for several technologies, and a method for comparing these representations to evaluate API compatibility from client's point of view.

However, design of most of the vehicle service API models directly using the benefits of W3C standardized modeling templates like WSMO, WSML and OWL-S tool remains challenging due to several reasons like frequently changing vehicle application software components requirements, changing vehicle standards, library dependencies, dependency on strict low-level protocols, etc.

B. Contribution

To design an approach for checking the syntactic and semantic level of service APIs' compatibility, a key consideration is that such interfaces are conceptually similar to a standard meta library of the domain and thus similar methods can be used to model the service APIs and reason about them based on this standard meta library [2][1]. The semantics traits of cross-enterprise vehicle service frameworks' APIs can be effectively compared and correlated to explore synergies, thereby enhancing interoperability between them. Platform-agnostic and technology-agnostic service APIs' semantic specification model for vehicle applications of heterogeneous SOA based frameworks is substantial for semantic interoperability and compatibility. Such generic service API semantic specification template for heterogeneous vehicle service component frameworks, must be expressed in an abstract syntax tree purely based on semantic traits independent of platform details. Most important, a generic vehicle service API semantic specification model would remain transparent and consistent in its representation format along with its other communication peer partners for service invocation, service discovery and deployment of the services to target platforms.

With this perspective, this research work proposes a modeling approach to standardize the semantic specifications of cross-enterprise vehicle services API models by using a platform-independent, ontological modeling template, based on certain fundamental semantic traits. Also, to ensure semantic interoperability between the semantically equivalent but syntactically different interface concepts of various vehicle services API ontological models, we defined an *ontology mediator*, *DM*, which is a platform-agnostic and domain-specific ontology. *DM* can be used as a common semantic background to glue the semantic bridge between the various concepts of heterogeneous frameworks' service API ontologies, based on identified synergies in semantic traits[1].

Nevertheless, for the easy accessibility of the semantic data of service APIs by clients within a domain, it is also equally important to connect the semantic service specifications of heterogeneous vehicle frameworks' API models to a corresponding standardized, platform-independent, human understandable, language-neutral syntax, for example OpenAPI [3][2]. OpenAPI uses schema language definition to describe service API models in a document format. As the JSON schema or YAML are easily understandable and accepted by most of the vehicle application frameworks from heterogeneous knowledge domains, therefore, we propose to implement a reference map between semantic specifications of vehicle service API ontological models and corresponding language neutral OpenAPI syntax, using a methodology known as *Service Grounding* [3][5]. To illustrate the proposed design approach and *Service Grounding*, a vehicle case study has been used.

II. Design Approach

With the given wide plethora of heterogeneous software platforms specific service API specification languages, it can be said that designing a service API that is long lived, so that the end users can understand and use it easily as a mutual API contract without many impediments and without time-consuming transformations is a bit challenging. With our proposed design approach towards platform-agnostic, generic vehicle service APIs' semantic and syntactic specification, let us consider a complex and novel vehicle domain *case study*, namely, *Keyless Vehicle Entry*, as illustrated in Fig. 1. In this *case study*, the owner of a car wants to give the vehicle access to someone just by using his mobile phone, and the owner of the car is geographically located far away from his car. This *case study* involves service collaborations from third-party, cross knowledge domain platforms such as Robotics, Telematics, Infotainment, Cloud, etc. To simplify the illustration, we consider the four most used cross-enterprise vehicle application component frameworks that are used as service collaborators to realize this complex *case study*. They are namely, AUTOSAR Adaptive, Franca (for Genivi),

Android and MuleSoft (for Amazon Web Services). These cross-enterprises vehicle application frameworks are from different knowledge domains [7]. To ensure semantic interoperability, it is substantial to semantically map between these heterogeneous frameworks' service API models based on identified semantic synergies between their API models' interface concepts. The semantic synergies between the service API models are manually identified based on vehicle domain specific, platform-agnostic, generic functional semantic traits of the APIs, also explained later in section III.

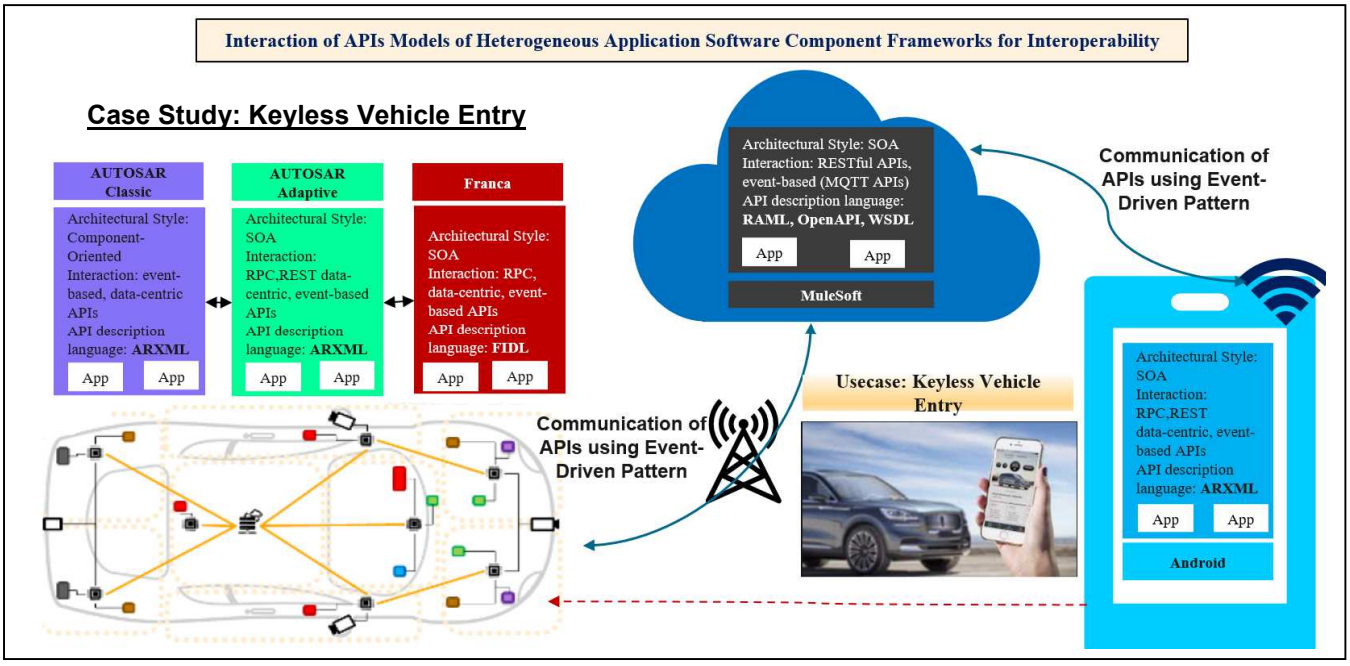

Fig. 1. Communication between heterogeneous platforms' service API models for novel and complex vehicle case study.

Exploring of semantic synergies manually between the heterogeneous vehicle service providing frameworks' API models ensures correlations between the API models' semantic concepts and opens scope for code reuse. As a next step, based on the semantic synergies identified between the heterogeneous service API models' concepts, a repository is virtually defined. This domain-specific, platform-independent, generic service API semantic traits virtual repository is further realized in form of an ontology, namely, *ontology mediator*, *DM* [1].

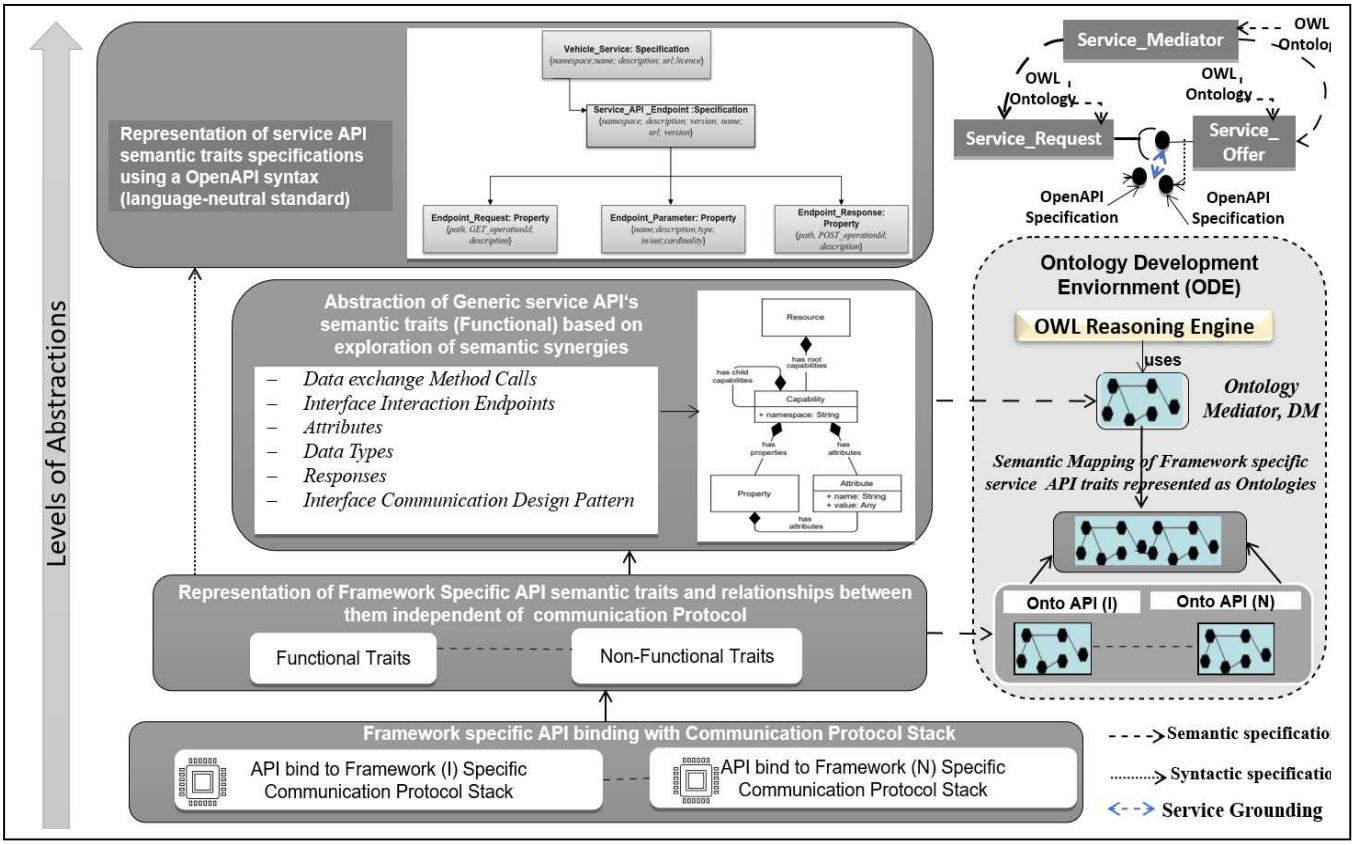

Fig. 2. Layered Design Approach for platform-Independent vehicle service API modeling using semantic and syntactic specifications.

Within the scope of the given *case study,* the *ontology mediator*, (illustrated as *DM* in Fig. 2 and Fig. 3), can glue the semantic gap between the heterogeneous knowledge domain vehicle service API models' concepts, when represented using a predefined ontology template, as illustrated in Fig. 3. The *ontology mediator* along with the reasoner of an ontology framework can be also used for service matching between a service consumer and provider [1]. The *inferred axioms* generated by the reasoner to glue the semantic gap between the API models can be further verified using SPARQL engine.

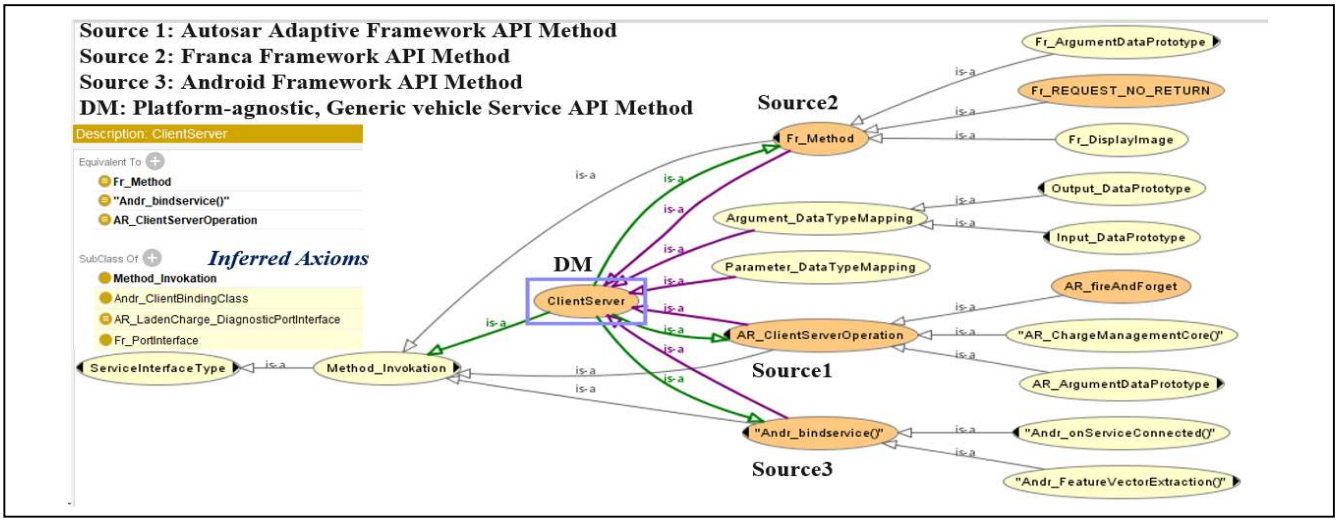

Fig. 3. Overview of use of an ontology mediator along with the inference from reasoner to semantically map the service frameworks' API models.

As a last step, to resolve the incompatibilities that emerges in the absence of *Semantic Web* knowledge between the SOA frameworks' API models during service invocation and discovery in accessing the APIs' semantic data given in the ontologies, we propose to map the semantic data in the various API ontological models to a corresponding platform-independent, language-neutral, OAS standard syntax using a methodology, named as, *Service Grounding* [3][6]. The *Service Grounding* methodology is based on OWL to OAS mapping and is illustrated in detail later in the next section [6].

III. Methodology for Implementation

To illustrate and realize the proposed layered design approach towards platform and technology agnostic service API modeling in vehicle domain, we use different methodologies in the following subsections to implement the different layers of the design approach (in Fig. 2).

A. Static Semantic Analysis of Heterogeneous Vehicle Application SOA Frameworks' API Models

In context to better understand the interface semantic traits of different vehicle service component frameworks and to bridge the semantic gap between them, we consider the vehicle service component models as *Resources* that describe the service process model including the name of the service and names of the service APIs. In general, for the semantic interoperability, the functional semantic traits of the APIs of the various *Resources* in vehicle application domain, fundamentally includes following[7]:

- *Data exchange Method Calls*: *Resources* might have method signatures containing information with valid parameter types, for example, *ClientServer*, *Sender-Receiver*, *Publish-Subscribe*, *Broadcast*, *GET*, *PUT*, *POST*, etc.
- *Attributes*: *Resources* may have specification of properties or fields to describe the service API, for example, for a book, the title of the book is an attribute, etc.
- *Interface interaction points*: Service API interaction end points for *Resources*, for example, subscription to *topics*, *message handlers*, etc.
- *Responses*: The specification of API responses for the *Resources*, for example, *Callbacks*, *Notifications*, etc.
- *Data Types*: Modeling of the API data for *Resources* through a streamlined type of system that encompasses JSON and XML Schema.
- *Interface Communication Design Pattern*: specification of communication design pattern between API models of service provider and service consumer, for example RPC, REST, Data Distribution Services (DDS), etc.

Based on the above mentioned fundamental semantic traits (functional) of service API models, the various vehicle application frameworks within the considered *case study* are manually semantically compared, as illustrated in TABLE I. [7][2]. In the considered *case study*, where vehicle applications request for services using RPC communication paradigm from cloud service providers which in turn deploys the requested service API using REST communication paradigm, in such scenarios, it becomes important for the vehicle application developers from the perspective of semantic interoperability to identify the semantic synergies in the different frameworks' service API models independent of any middleware communication protocol.

B. Ontological Semantic Specification for Platform-agnostic Vehicle Service API Models

Based on the semantic analysis of vehicle application SOA frameworks' API semantic traits (functional) and subsequent exploration of semantic synergies between the API semantic concepts, the layered design approach (as illustrated in Fig. 2), abstracts these traits and defines a platform-agnostic, vehicle domain specific *Generic API semantic traits repository*. The *Generic API semantic traits repository* is a virtual repository that contains fundamental API functional semantic traits (explained in subsection *A*) that are functionally common or

overlapping in most of the vehicle application SOA frameworks independent of platform and technology details[4]. The virtual *Generic API semantic traits repository* is implemented in real world as a platform-independent, domain-specific ontology. This domain-specific modelled ontology is named as an *ontology mediator*. Due to its domain-specific nature, the *ontology mediator*, namely, *DM*, can be used as a common semantic background to glue the semantic bridge and ensure interoperability between the various APIs' semantic traits of heterogeneous frameworks. TABLE I. [7][2] illustrates the semantic comparison between API traits of frameworks considered in the given *case study*[1].

TABLE I. SEMANTIC COMPARISON OF API FUNCTIONAL TRAITS OF VEHICLE APPLICATION FRAMEWORKS IN CASE STUDY

SOA Frameworks for Vehicle Services	Data Types Supported	Interface Description Language	Interface Interaction End points	Communication Design Pattern	Communication Paradigm	Request/Response Type	Method Behaviours
AUTOSAR Adaptive (*From: Automotive domain*)	Int,float,double, String,Boolean, Associate Map,enum,Vector	ARXML	Events, Method_Calls()	Client-Server,	RPC, REST(using SOME/IP with underlying TCP/IP))	Messages for: Event Subscription, Notifications	Synchronous, Asynchronous (limited scope)
Franca+ (*From : Car Infotaintment Domain*)	Int, float, double, String, Array, Bytebuffer, enumeration, Boolean, struct, union, Map, Constants	FIDL, FCDL	Events, Method_Calls()	Client-Server	RPC (using SOME/IP with underlying TCP/IP))	Messages for: Event Subscription, Notification, Broadcast	Synchronous, Asynchronous (limited scope)
Android (*From: Telematics Domain*)	Int, long, float, char, boolean, double,String, List, Charsequence	AIDL	Events, Method_Calls()	Client-Server	gRPC(using TCP/IP) REST (using HTTP)	Messages	Synchronous, Asynchronous
MuleSoft (*From: Semantic Web Domain*)	As JSON Objects	RAML (using YAML)	Events, Method_Calls()	Request-Response (Client-Server)	REST(using HTTP)	Schemas	Synchronous, Asynchronous (limited scope)

However, as *DM* is a platform-independent ontology, the semantic traits specified in *DM,* can also be flexibly customized in future and reused within the domain based on scopes of different vehicle functional clusters or complex case studies[4]. Apart from modeling the *ontology mediator* for interoperability, the vehicle services frameworks' API models' must be also abstracted and semantically specified using a generic, standardized ontology template independent of platform and technology specific details and focusing on the fundamental semantic functional traits that are described in the earlier subsection *A*. In general, for using such an ontology template for modeling vehicle service frameworks' APIs semantics, we propose mapping of various (1) *Resources* to *concepts*. (2) *Data exchange Method Calls* to also *concepts* and represented as *owl:Class* and *rdfs:subClassOf* (3) *Attributes* to *ObjectProperty* or *DatatypeProperty* (4) *Responses* to Outputs of operation, that is, the data an operation produces, (5) *DataTypes* to *rdfs:range*.

C. Language-neutral Syntactic Specification for Vehicle Application SOA Frameworks' API Models

In scenarios, where a service requester who has discovered a suitable service with the help of a service matchmaker or *ontology mediator* and a reasoner and he want to further interact with this service API to implement a complex vehicle IT-solution. In such scenarios, where the services APIs are specified by their abstract semantic data, the corresponding syntactic specifications of the service APIs are still substantial for precise invocation of the services and for ease in access to services APIs' semantic data. *Service Grounding* methodology allows automotive software developers easy accessibility of service API's semantic data specified in ontology-based knowledge graphs by mapping the ontology to a corresponding interface that developers are used to work with. This is especially important, in the case of developers who have an interest in the services APIs' semantic data available in ontology knowledge graphs but are not used to *Semantic Web technologies*.

OAS as a part of *Swagger Hub* application framework, is broadly adopted as a de facto standard for describing REST APIs in a programming language-agnostic interface. In fact, developers are now more familiar with REST APIs as a resource-access way as it hides details about the implementation of operations for resource management [6]. It is also possible to integrate RPC semantics in API documentation using OAS standard. In general, the major building blocks used in specification of APIs semantic data in a language-neutral representation using OAS, includes namely, *Vehicle_Service API type specification*, *End_point specifications for Request and Response and Vehicle Service API property*, as also illustrated in Fig.2. *Service Grounding* methodology is implemented based on mapping of API entities from OWL2 to OAS 3.0.0. Some of the major mappings include mapping between (1) API's ontology *Class* or *concepts* to OAS *Schema objects*. (2) *Data and ObjectProperty* to attributes or *properties*, primitives, expressions of OAS *Schema objects*. In an OAS schema, a *Path Object* holds the *Resources* exposed in an API. A *Path Item Object* describes the available operations to manage the *resource* on a single path [6]. To further illustrate the *Service Grounding* methodology, we consider a reference implementation of mapping of semantic entities from a vehicle service API ontology knowledge graph (specified in OWL2) to a corresponding language-neutral, syntactic specification document using OAS3.0. For this, we consider a functional part of the given *case study* (in Fig. 1)[3][6].

The functional part of the *case study* considered is, namely, *Face Recognition*, as illustrated in Fig. 4. As seen, firstly, an *Android* platform's service software component to describe the service process model on *Face Recognition*, is

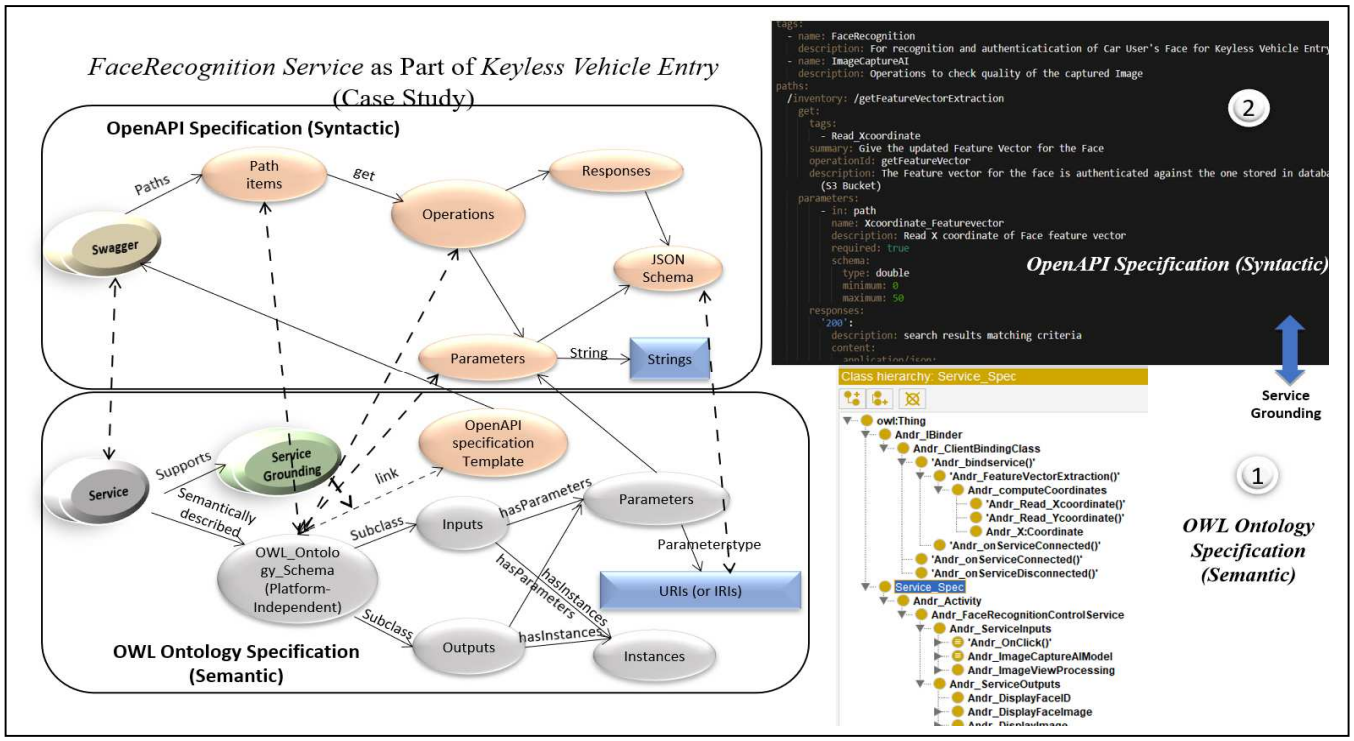

Fig. 4. Reference implementation of OWL to OAS mapping for the Face Recognition service as part of the case study.

semantically specified using a standardized ontology template. As a next step, to ease the service API's semantic data accessibility for subsequent exchange of services, the API ontological model' entities are mapped to their corresponding syntactic specifications in OAS schema, using a *Service Grounding* migrator tool [6] based on OWL to OAS mappings [6]. When compared with other existing language-neutral syntaxes for service API specifications like RAML, GraphQL, API BluePrint, etc., OAS provides several advantages. Due to its wide adoption, OAS has a big community behind, which has provided various tools to allow developers to generate API documentation, API versions, API codes generation in preferred programming languages using a *codegenerator* [2].

IV. Conclusion

This paper describes a design approach towards modeling of platform-agnostic, technology-agnostic, standardized, vehicle domain service API models. To ensure services semantic interoperability, the proposed design approach uses an abstract OWL2 based standardized ontology template for describing platform-independent semantics of various vehicle service API models key concepts. With the given approach, to improve semantic mapping and interoperability between heterogeneous vehicle SOA based frameworks APIs' ontologies, design of an *Ontology Mediator* as a platform and technology independent ontology has been proposed. The *Ontology Mediator* is used along with a reasoner support to glue the semantic bridge between the various concepts of heterogeneous vehicle services APIs ontologies. For easy accessibility of service APIs' semantic data during service discovery in absence of *Semantic Web* knowledge, a *Service Grounding* methodology is proposed. *Service Grounding* connect the semantically specified API data in ontologies to corresponding syntactically specified, language-neutral, platform-agnostic OpenAPI standard schema. To illustrate the proposed design approach and *Service Grounding,* we considered a typical vehicle domain *case study* and implemented a part of the *case study* to demonstrate OWL2 to OAS mapping. However, not all the interface entities between the various service API ontologies could be successfully semantically mapped to one another, therefore, we would like to further extend this contribution, in future, to address this limitation and also in the direction of *Service Grounding.*

References

[1] E. D. Valle and D. Cerizza and I. Celino, "The mediators centric approach to Automatic Web Service Discovery of Glue", MEDIATE2005. 168, 2008, pp. 35-50.

[2] Z. Vales and P. Brada, "Service API Modeling and Comparison: A Technology-IndependentApproach", 46th Euromicro Conference on Software Engineering and Advanced Applications, 2020, pp. 158-161.

[3] S. Schwichtenberg, C. Gerth and G. Engels, "From Open API to Semantic Specifications and Code Adapters," 2017 IEEE International Conference on Web Services (ICWS), USA, 2017, pp. 484-491.

[4] M. Feld and C. Müller, "The automotive ontology: managing knowledge inside the vehicle and sharing it between cars", In Proceedings of AutomotiveUI, New York, NY, USA, 2011, pp.79–86.

[5] D. Roman, J. Kopecky, T. Vitvar, J. Domingue, and D. Fensel, "WSMO-Lite and hRESTS: Lightweight semantic annotations for web services and RESTful APIs," Elsevier Journal, vol. 31, 2014.

[6] P. Espinoza-Arias, Garijo D. and O. Corcho , "Mapping the Web Ontology Language to the OpenAPI Specification", In: Advances in Conceptual Modeling. ER 2020, vol 12584. Springer, Cham, 2020.

[7] S. De, M. Niklas, B. Rooney, J. Mottok and P. Brada, "Towards semantic model-to-model mapping of cross-domain component interfaces for interoperability of vehicle applications: An approach towards synergy exploration". In: Joint Proceedings of the Workshop MDE4IoT & ModComp, 2019, pp. 57-64. ISSN: 1613-0073.

[8] E. Kilgarriff, B. Sapkota, L. Vasiliu and D. Aiken, "XML to WSML adapter Implementation", in Proceedings of the 2nd WSMO Implementation Workshop, 2005.

[9] A. Heß, E. Johnston, and N. Kushmerick, "ASSAM: A tool for semi-automatically annotating semantic web services,"in Proceedings of the Third International Semantic Web Conference (ISWC), 2004, pp. 320–334.