An Approach to Support Semantic Discovery Using Ontologies to Describe Aeronautical Web Services Repositories

Luís Antonio de A. Rodriguez[1] and José Maria Parente de Oliveira[1]

[1] *Aeronautics Institute of Technology (ITA), Pça Mal Eduardo Gomes S/N, São José dos Campos, Brazil*

Abstract

The lack of an effective and accepted method for implementing a semantic web services infrastructure to make it possible to execute automatic tasks, like services discovery or orchestration, leads developers communities to face a non-standardized and disorganized backlog of published Aeronautical Web Services. Developers must to hard code the connections between publishers and consumers and this kind of ad-hoc network makes the simple need to compose or to find a specific Web Service a nightmare for programmers because of the lack of semantics. Current practices tend to mix technologies and use small portions of the OWL-S Ontology for trying to specify semantic "anchors" that serve as single identifiers, making it possible to find them when they are embedded into SOAP artifacts as WSDL or UDDI. To manage these mixed descriptions, it is necessary to build software tools that match them by using known algorithms and filtering result lists, mixing technologies, increasing complexity to access those artifacts and coupling developers to specific protocols, tools and methodologies. This paper presents a generic implementation model of the OWL-S ontology architecture which describes semantically, and using a machine-readable language, a web services' registry destined to publish, advertise and make a multi criteria semantic discovery of Brazilian Aeronautical Web Services aligned with the Service Description Conceptual Model (SDCM) of the System Wide Information Management (SWIM), within the Air-Traffic context. Furthermore, a set of Competency Questions was formulated, answered after a translation to SPARQL queries to make it possible to present a comparison between the obtained results and the previously described ontology's functional requirements.

Keywords

Aeronautical Web Services, OWL-S, Semantic Web Services Discovery, Ontologies

Proceedings of the XVI Seminar on Ontology Research in Brazil (ONTOBRAS 2023) and VII Doctoral and Masters Consortium on Ontologies (WTDO 2023), Brasilia, Brazil, August 28 - September 01, 2023.

rodriguezlaar@gmail.com (L. A. A. Rodriguez); parente@ita.br (J. M. P. Oliveira)

1. Introduction

Web Services (WS) were conceived to provide users with information but the evolution process of use of the World Wide Web has proved they can do much more complex operations, all related to reaching human ordinary goals. The use of this technology has spread [3] because of some facets and the multi-platform is one of them, which allows developers to be not coupled with any programming language; it also is a fact that the realization of interoperability between providers and consumers using a reliable exchange of XML messages, which is the major reason for those spread.

The common current standards for describing WS focus on interoperability across development platforms, but there is a lack of mechanisms to provide some automation [19] to allow developers to program intelligent searches or service compositions based on service's functional and non-functional properties described in the repositories of WS. As the technology has spread, the number of providers and consumers has increased and so has the variety and number of WS, making it difficult the developers task to program software to make service discovery using the least possible amount of time and considering a set of consumer's requirements. The artifacts used to describe the WS metadata are based on XML descriptions, forcing developers to deal with syntactic search which is not unambiguous.

The need to share and the reuse's facet of Service-Oriented Architecture (SOA) has made developer communities to face serious programming costs to publish, advertise and mainly locate other WS which could provide solutions for their problems or create compositions [7] destined to "more complex composite WS" in a non-standardized environment. The limitations are related to the absence of meaning in an XML description [20].

Trying to satisfy the absence of entities explicit meaning in a XML's tag description, developers have tried to bring meaning for the domain's objects, coupling to the syntactic descriptions different classification schemes, which were able to describe WS functionalities using declared taxonomies like NAISC or UNSPSC, from USA manufacturer industry. For instance, a fictional company called Aerial Cargo Transport Service could use an UNSPSC code 78.10.15.01.00 to describe a functionality to fit volumes into aeronautical pallets described in a WSDL or YAML description. These classification schemes were used to link [20] known reference terms into those standardized XML descriptions to be serialized and tracked to match a feature about a WS which is classified and described by the term embedded into the classification schemes dictionary. This way it is possible to find the desired WS by matching known syntactically described names into the XML with a meaning presented into a dictionary; and that was one of the first steps to bring semantics to WS descriptions. Although it is possible to discover WS using taxonomies described in classification schemes, the search yields coarse results and several errors in many cases [20].

An effective and precise WS discovery must focus on semantics for identification of entities and depends [11] directly on the semantic ability to make machine-readable specifications about WS which can be interpreted by programming languages' API. These specifications can be implemented by using ontologies, in which OWL-S [3] is an ontology architecture designed to allow users to create Semantic WS descriptions, artifacts which are able to be queried by a single protocol, the SPARQL language [17]. The results of these queries are semantic stereotypes from the OWL language which define precisely what is the exact service which matches a set of consumer requirements. The initial idea about this work was to identify, among a set of services, the single one which matches to satisfy a set of requirements by using a single method.

Some works in this area were focused on mixing technologies to try to bring semantics to the WS repositories, using OWL language coupled to existing artifacts like WSDL and UDDI [11], aiming to support some level of automation by searching onto this portions of semantic descriptions which are embedded into syntactic XML artifacts and identifying some metadata about a specific WS written on these descriptions. This mix of technologies appears to be a huge job since developers must understand, must build software to serialize both technologies in a single XML file and must identify, applying known algorithms on syntactic text, the semantic portions which are essential to discover and invoke a specific WS. The complexity to manage such mix of technologies and the bad results [20] obtained have led developers to deal with flaws in semantic discovery of WS, and the syntactic search has become the nightmare of programmers that have to build different WS "finders".

This paper presents an approach to support semantic discovery by using a single protocol and based on an implementation of OWL-S to describe a Brazilian Aeronautical Web Services repository. The paper is organized as follows: Section 2 presents the SWIM model, for the understanding of the

use case, Section 3 presents a Background with a quick overview about the Methontology, the methodology the authors have used to develop the ontologies and the OWL-S, the "ontology for services", Section 4 presents current approaches to make Service Discovery and its uses. Section 5 presents the use case and a set of experiments, and its results, using Protegé, which makes Service Discovery using SPARQL queries. Section 6 presents the conclusions and further works.

2. The SWIM

The System Wide Information Management Program (SWIM) is a standardization of international civil aviation communication and interoperability [1]. It consists of data exchange models and standards, infrastructure and governance which enables management of Air-Traffic information and its exchange between qualified parties via interoperable patterns. It is a primary ICAO (International Civilian Aviation Organization) [1] technical work program on information standardization to complement human-to-human communication with machine-readable communication [2], aiming to increase air navigation data distribution and accessibility with quality of data exchange.

The implementation of the SWIM concept must create an interoperability environment which allows all the SWIM IT systems to cope with the full complexity of semantic information descriptions and to be able to make semantic data exchange. The SWIM concept introduces a significant change to the practices to manage information during the whole life cycle of the Traffic Flow Management process. Infrastructure, standards and governance recommend standardization by promoting semantic and structural interoperability among stakeholders by developing a common set of semantic and structural artifacts (taxonomies, ontologies and controlled vocabularies).

The SWIM relationship with W3C standards means the use of OWL-S architecture, which is formalized through an extensive documentation which considers it for all the modeling artifacts and systems descriptions. The semantic power of that ontology is reinforced by SWIM requirements specifications to provide precision and quality for the exchanged information. This paper follows the OWL-S requirements specifications of SWIM [2] to implement the idea.

3. Background

3.1. Methontology

The authors have chosen a formal methodology to develop the customized OWL-S: the METHONTOLOGY, also known as Methodology for Ontology Engineering. It is a methodology to develop ontologies which provides a systematic and structured approach [24] to guide practitioners through the ontology engineering process. The main focus of the METHONTOLOGY is to ensure the quality, reusability, and maintainability of ontologies.

The methodology consists of several activities that are performed in a sequence:

1. **Specification**: The goals and scope of the ontology are defined. This activity helps in understanding the domain and the specific objectives of the ontology.
2. **Conceptualization**: The creation of a conceptual model that represents the knowledge domain. The conceptualization phase helps in capturing the domain knowledge and structuring it into a coherent ontology.
3. **Integration**: Focuses on incorporating external resources and existing ontologies into the development process, promoting interoperability and re-usability by leveraging existing resources.
4. **Implementation**: The ontology is translated into a formal language such as OWL (Web Ontology Language) or OWL-S (the Ontology for Services). This activity involves defining classes, properties, and axioms, and ensuring the ontology adheres to the formal syntax and semantics.
5. **Evaluation**: The evaluation activity assesses the quality and effectiveness of the ontology. It involves performing tests, reviewing the ontology against predefined criteria, and validating its correctness and coherence through the competency questions.
6. **Deployment:** Once the ontology has undergone evaluation and refinement, it is ready for deployment and to be available to users and integrating it into systems or applications.

Throughout these activities the METHONTOLOGY emphasizes the use of best practices, guidelines, and documentation to ensure the ontology is developed in a systematic and well-structured manner. The methodology promotes the collaboration of domain experts and ontology engineers to capture and represent domain knowledge accurately and effectively.

3.2. OWL-S: The Ontology for Web Services Semantic Descriptions

Ontologies are seen as structured knowledge collections [5]. An ontology is a logic theory which describes a specific meaning of an entire formal vocabulary, which also represents a commitment to conceptualize a specific domain at the Semantic Web. This integrated vision should enable full access not only to content, but also to services on the Web, or the Web Services [21]. The idea is about users and software agents being able to discover, invoke, compose and execute services with a high degree of automation. OWL-S [3] is an ontology for describing Web Services which makes these actions possible.

The overall structure of the OWL-S and its architecture of ontologies suggests different goals [6], as presented in Figure 1. As can be seen in the figure, the Service ontology (*Service.owl*) is composed of three OWL Classes which have specific relationships among each other: the *ServiceProfile* that makes the advertisement of the WS and makes it possible to implement semantic discovery by declaring functional and non-functional features, the *ServiceModel* which gives a detailed description of the WS' operations, describing the whole process executed by it, and the *ServiceGrounding* that is responsible for a complete description of how to deal with the services via messages, protocols and other physical connections and ports necessary to navigate in a network.

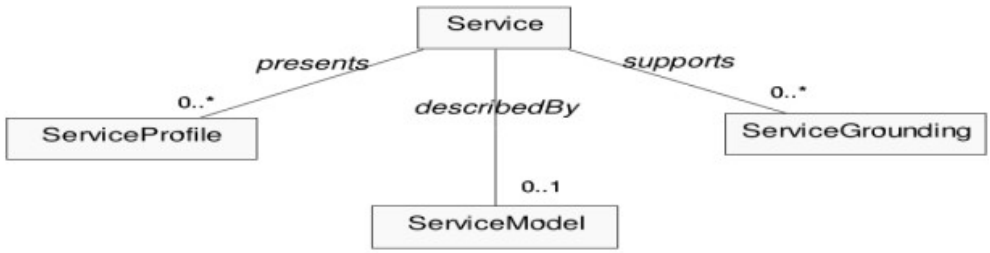

Figure 1: The Upper Level of the OWL-S Ontology Architecture [6]

OWL-S was built on OWL stereotypes and it is a standard [3] developed by the Web-Ontology Working Group. This paper is focused only on the *ServiceProfile* part of the OWL-S architecture, or the advertisement features. A high level of expressiveness present in the OWL-S architecture allows developers to implement [21] semantic descriptions using it as a model and using its ability to define conceptual features embedded into a machine-readable language which makes it possible to have enough automation to make intelligent service discovery. The OWL-S' Profile Ontology also provides a platform to categorize several different types of criteria to make advertisements about WS and to present features about them, allowing human consumers or applications to discover one of them precisely and automatically.

Current approaches in service discovery support the idea of making use of functional and non-functional criteria associated with the OWL-S *Profile* Class [21] and these are powerful semantic descriptions which could allow developers to publish, advertise and make semantic discovery on a semantic registry, exactly like the UDDI and WSDL artifacts from SOAP. Figure 2 is a UML 2 Class Diagram of an ontology which is a second level of the OWL-S' original architecture and a set of criteria to advertise WS using the original *Profile* multi criteria features [6]. In order to describe the functionality of a service, the OWL-S architecture encompasses an ontology called Profile, which is the specification of OWL Class *ServiceProfile*.

The Profile ontology also provides some human-readable information about its name (*serviceName*), working requirements (*textDescription*) and the mechanism to refer to humans or companies responsible for that WS (*contactInformation*) [3]. The Profile ontology description can handle the issues of information transformation by representing a huge set of features about the

services like the input necessary to invoke and the output, also known as the delivery, which is a set of behavioral descriptions like the change of state caused by the execution of the WS or the preconditions and effects. All of these features are defind by means of OWL properties.

The relationships define specific properties and characteristics of an instance of Profile, like the *ServiceCategory* and *ServiceProduct*, unique information about a unique instance. Each Profile's instance must have an association to one or more instances of each other connected entity as in Figure 2. The entities connected to the Profile Class at the center are native OWL Classes, Object Properties and Datatype Properties and each one presents some type of a specific characteristic which is unique for the description of one triple that is composed by an instance of Profile, a connection property, and another instance of a different Class.

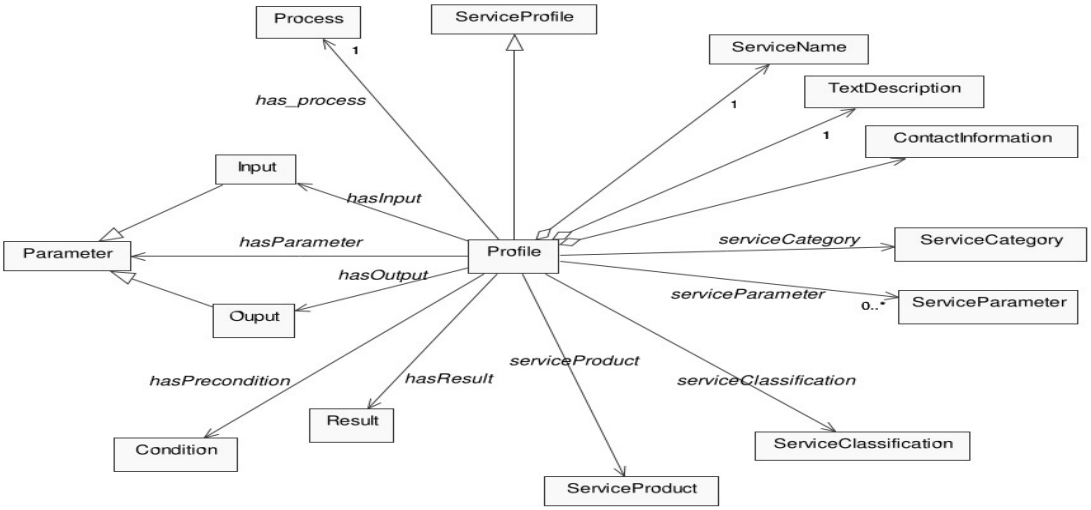

Figure 2: All Profile Associations [6]

It is possible to build a complete implementation of OWL-S ontology advertisement foundation by filling each OWL Class in Figure 1 and the Profile ontology in Figure 2 with a set of instances [6]. Each instance, connected to another one from another owl Class using a specific *ObjectProperty* or, connected to a specific s*erviceName* and *textDescription, DatatypeProperties*, provides a 'predicate' or an 'object' of a triple which describes a feature about the WS which can be queried. The OWL-S architecture and specification allow developers to literally build a Semantic Registry, making possible to pose queries in SPARQL [17] and obtaining the corresponding answers, which could be used like messages' flow to software agents to execute their actions and to accomplish their goals.

4. Current Approaches to make Web Services Discovery

Service Discovery can be defined as the process of matching user requests with some available web services [11]. In general, to identify and choose a WS is necessary to match words syntactically, such as a service name or a service text description which are written using standards like WSDL / UDDI [3] or YAML & JASON [22]. Different techniques have been proposed to improve the accuracy of the service discovery's results [23]: data mining [14], Mapping algorithms [13], ontology based and service description [8] and all of them have reached the conclusion that information matching can be done at two levels:

- Syntactic matching - similarity of data is found using syntax driven techniques.
- Semantic matching – the key idea is the mapping of meanings between concepts.

Pranav Kadam [10] presents a type of service discovery which uses a matching process based on non-functional parameters. The providers advertise about their WS presenting their capabilities using non-functional specifications and criteria along with evident functional matching. The QoS is used to specify those parameters and some other matching schemes are used, like Domain, Category and Business offer to build a matching algorithm oriented to a specific domain of services. The QoS specification is an extension of the original Profile and has its customized taxonomy. It is a difficult

work to implement the matching schemes, apply an algorithm and filter results to get the expected results.

Almeida [9] presents a reference model to the Brazilian military command and control (C2) systems. The idea is a centralized and structured model of specification led by a reference data model called JC3IEDM, which allows dynamic compositions of Web Services semantically described like this model. This work presents a tool to convert a regular WSDL into a customized one, which has small portions of OWL-S specifications to bring some semantic description for this artifact. It also presents a modified UDDI registry to publish the advertisement about the web services. The work is strongly coupled to SOAP technology and tries to bring some semantics to discover the services. It is possible to say it is a huge effort to understand the JC3IEDM model because of its size.

Marco et al. [17] describe an approach for the description and discovery of semantic web services using SPARQL language and software agents. They use SPARQL to describe the preconditions and post conditions of web services as well as the goals of each agent to make discoveries. Also, they show that the SPARQL query evaluation can be used to check the preconditions in a given context, to build post conditions which will result from the execution of the service and determine if it satisfies or not the agents. Consumers are interested in some other criteria besides conditions, like the user's evaluation about the WS, QoS factors and some others.

A major part of the actual research for service discovery is strongly coupled to existing technologies [20] and tries to use small portions of semantic descriptions integrated with existent artifacts pertaining to the actual standards for Web Services to discover them. All of them offer a complex architecture of implementation which mixes semantic artifacts with syntactic standards of actual web services description and none of them have implemented a complete semantically described solution. All of them also offer a customized tool to convert the original artifacts to describe web services into "semantically enriched" new customized artifacts and it is a limitation to adopt these ideas.

The solution proposed in this work takes into account to access only a single information kernel and by using a single protocol. The kernel is composed by a set of published formal ontologies which were developed based on the original OWL-S architecture but refactored to abide Brazilian Aeronautical Laws and aiming to support all operations over common standard WS. The solution proposal aims to guarantee the complete semantic description of the Web Services, which is the foundation necessary to create automation also to make semantic discovery. The access made by using a single artifact and the unique SPARQL protocol is the crucial difference among those works and this one proposed here by the authors.

5. An Implementation of OWL-S to Support Semantic Discovery of Web Services

World's aviation agencies are rushing to build formal vocabularies, ontologies and semantic descriptions that could serve as a machine-readable knowledge to support intelligent information systems [25]. The goal is to reach interoperability with all the nations aligned with the SWIM model and it is necessary for Brazilian aviation to establish its own WS semantic description models to interact with other countries.

Thus, it is possible to state the following problem: *"The excess of complexity of the syntactic throughput of web services actual architecture's standard artifacts to execute operations over them like service discovery. It is caused by the lack of a semantic foundation on the Web Service's descriptions, in particular that ones used to support air-traffic management (ATM) information systems"*. This is really a problem since the descriptions are built syntactically and there is no inference, there is no necessary automation to execute more complex tasks like the discovery of services. The works presented in Section 4 try to bring some kind of semantics by mixing W3C's syntactic standards with different taxonomies representation, including W3C's own representing ontologies languages, which are semantic artifacts. In using that technologies, developers must:

- Serialize syntactic artifacts (UDDI, WSDL) which contain more than one standardized language to write knowledge (XML, TBox, XSD, OWL, OWL-S);

- Separate the "worlds of perceptions" about those different languages to become possible to execute CRUD operations over them;
- Search the reference parts of the artifacts and to compare their original parameters with the semantic descriptions (they are there for that);
- Apply some algorithm to compare parameters;
- Implement a tool to execute the filter to get more precise results.

The authors have not found a complete solution for implementing semantic repositories for WS, where developers could have semantic descriptions [6] able to offer an infrastructure to describe WS containing a high formalism's level which could allow software agents to interact with the repository making complex interactions like semantic searches or WS compositions. Following W3C's recommendations to formalize descriptions using machine-readable languages, the authors proposed the following solution for the problem: *To implement a customized OWL-S ontology aiming to make it act as a Brazilian Aeronautical Web Services' Registry and to be able to offer support to make semantic discovery of Web Services.*

The authors chose the Methontology [24] to formally develop the customized OWL-S ontology and for applying the defined approach they have described the first phase, **Specification**. Thus, the goals are the following:

- To develop a customized OWL-S ontology based on a set of Competency Questions (CQ) [15] to define the functional requirements . The proposal of the ontology is to act as a WS registry which is able to store all functional and non-functional features on aeronautical WS semantic descriptions coupled to the OWL-S original ontology, but supporting the creation of some entities to abide Brazilian aeronautical laws;
- The **scope** of the ontology is the set of web services regularly offered by the Brazilian authorities to the Air-Traffic community;
- The set of CQ must be translated to SPARQL queries [3] to allow an evaluation that compares the results of these queries with the CQ to make a complete check of requirements accomplishment. To execute the semantic discovery by running SPARQL queries to find specific WS' instances described in the developed OWL-S ontology (which represents, each one, a unique WS) as a result, by applying one or more search criteria where: ***search_criteria*** = {Category, Name, Result, Input, Output, Process, Condition, Provider, Geographic Region, Expression, QoS Rank}

The ***search_criteria*** presents the whole set of search's criteria which is going to be adopted during the experiment execution. The criteria is described exactly as in the original *Profile*.owl ontology, in the OWL-S architecture. The idea of the experiment was to find the same instance of WS by searching it using any of those different criteria, or, make different queries and find the same instance associated to each criterion to all WS described in the OWL-S_BR.

Next step according to the approach [24] was to build the **Conceptualization** of the ontology and it was made by generating an architecture's project as defined in Figure 3. The customized OWL-S was named **OWL-S_BR**, for Brazilian purposes and the task was to define its final architecture. The original OWL-S' latest release offers several ontologies which can be downloaded from official channels [3] in the ".owl" format. Each of them were used to define a specific small domain related to characteristics about the WS global descriptions, like the execution process, the advertisement, the access and security policies or the QoS (Quality of Service).

Thus, the conceived architecture of the OWL-S_BR presents the Brazilian implemented ontology architecture in accordance with all the recommendations of the Web-Ontology Working Group [3] and the suggestions mentioned at the OWL-S Release 1.2 webpage. Each component of Figure 3 represents a different ontology or class and the whole set of owl files from the original release was used and some entities were created (*SparqlInput*, *SparqlOutput*) for customizing Brazilian laws and rules and also the entities are related to SPARQL for acting as a message flow protocol when accessing the ontology as an information kernel.

For the next step, the **Integration** phase of the approach, the authors have set all **namespaces** of all ontologies described in Figure 3 aiming to publish it on the internet at:<http://www.hildeproject.com>. The namespaces of the original OWL-S ontologies were modified and the authors have integrated the whole set of ontologies to the original *Profile*.owl aiming to make

part of the original architecture to define this domain. After this phase was finished, the next step was to implement the ontology.

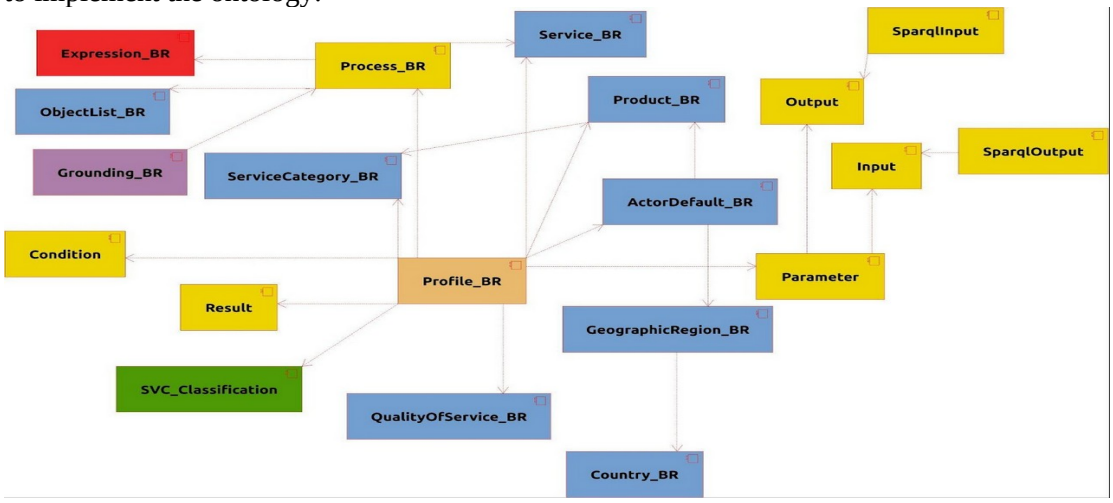

Figure 3: OWL-S_BR ontology architecture (from the authors)

For the **Implementation** a set of CQ to define the questions which must be answered by the ontology was created. Exactly like the method and data available in *Mendeley Data* [15], a set of several CQ was created to define all the necessary functional requirements to implement the ontologies shown in Figure 3. For a matter of space, a small part of these questions is presented in Table 1.

Table 1 - Competency Questions

CQ (Functional Requirements)	
What is the Web Service which is associated with a specific instance of ServiceCategory_BR?	*What is the Web Service which is associated with a specific instance of ActorDefault_BR which is a PROVIDER?*
What is the Web Service which is associated with a specific instance of Product_BR?	*What is the geographic region of a specific Provider?*
What is the Web Service which is associated with a specific instance of Result_BR?	*What is the Web Service which is associated with a specific instance of FinalRank (QualityOfService_BR)?*

The next step was to create the WS specifications by the creation of each WS which would compose the registry. To represent each WS, an ***Individual*** of the OWL **Class *Service***, pertaining to the *Service.owl* ontology was created . The population of the OWL-S ontology was made aiming a *top-down approach* taking into account the OWL-S exactly as presented in Figure 1. The *Service.owl* ontology was populated with a set of different kinds of *OWL Individuals* which represent the real Brazilian Aeronautical Web Services and a set of related Individuals were created for the owl:Classes: ***Service, ServiceModel, ServiceProfile*** and ***ServiceGrounding*** which would compose the whole set of WS' original upper level descriptions.

After the creation of the instances for the *Service.owl* ontology, the corresponding connections among each class were created exactly as presented in Figure 1, using *OWL Object Properties* to define the associations presented in that UML diagram. Figure 4 shows the Individuals of the owl Class ***Service*** and the set of WS already named. Observing the Classes presented at the left side of Figure 4, all the connections between two specific instances of those four Classes were set using specific Object Properties already existing at the original OWL-S ontology, exactly as presented in Figure 5.

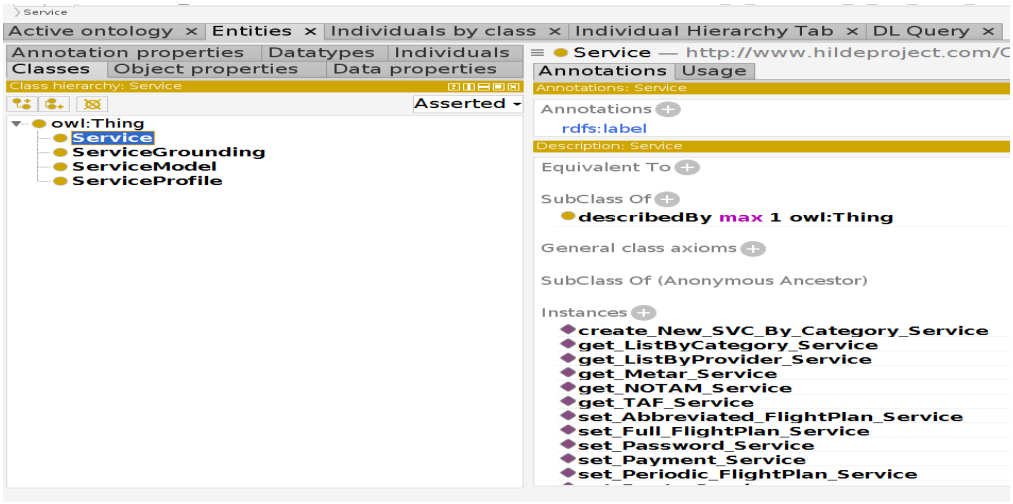

Figure 4: Set of implemented Brazilian Aeronautical Web Services

Figure 5 presents, in blue colour, the original *Object Properties* of OWL-S being used to associate the instance of Web Service **"get_Metar_Service"** to three other **Individuals** coming from the other three OWL Classes of the Upper level: **get_Metar_Profile, get_Metar_Process** and **get_Metar_Grounding**, as seen at the right side. Twenty instances of owl:Class Service have been implemented and the consequence was the creation of the same amount of instances for the other three Classes of the Upper level. After this level implementation was all set, the authors focused on preparing the ***advertisement's implementation***, the focus of this work and which would be the semantic foundation to allow the necessary description's formalism to have automation to execute complex tasks like semantic discovery [21].

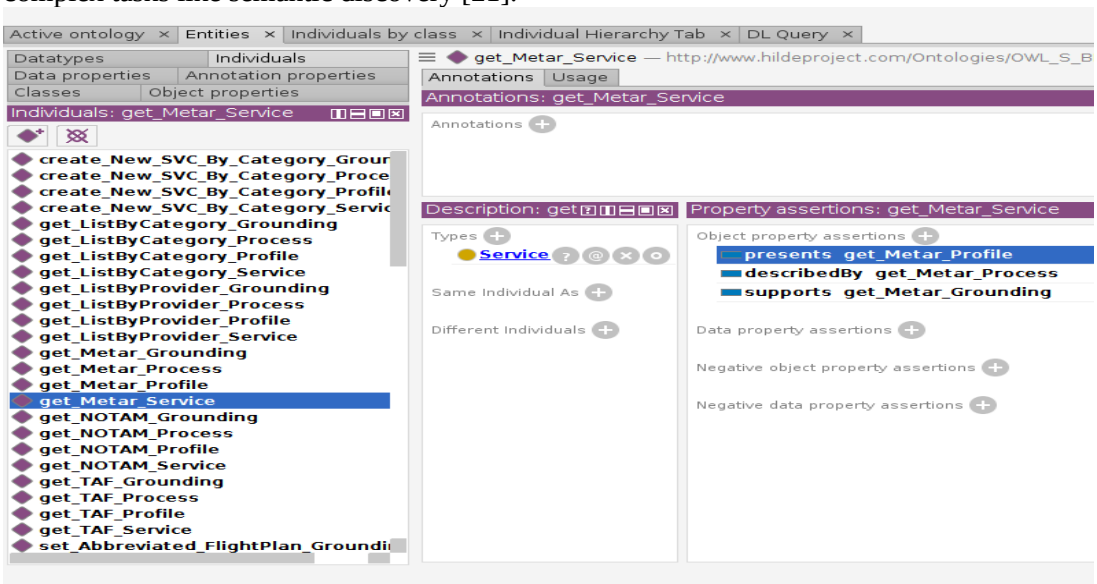

Figure 5: Figure 1's associations represented by the example of WS: *get_Metar_Service*

To implement the advertisement feature of OWL-S, the authors created another set of instances, this time into the **ontology *Profile.owl*** exactly into the Class *Profile*, at the center of Figure 2, and they also created all the associations for each connection of this Figure. Then the authors implemented a triple repository for each instance of Profile which would serve as a multi criteria way to find any of them by making SPARQL queries passing one element of ***search_criteria*** as a parameter.

Figure 6 presents the whole set of implemented connections for an **Single Web Service**, a unique Individual of the *Service* Class at the bottom left. For the advertisement part related to this specific WS there are several instances of different Classes connected to a single instance of Profile, representing, each one, one element of the ***search_criteria*** set mentioned before and trying to build a semantic foundation for all WS descriptions and creating a RDF Graph which makes possible to

execute the semantic search approach proposed in this work. Figure 6 presents the complete set of relationships of a single instance of the web services registered.

Observing Figure 6, it is possible to realize that there are several "ways" to navigate trough this RDF Graph (or, a triple repository) to reach precisely an instance of *the Service Class* just by using the SPARQL protocol. It only depends on the distance of the node from the Profile's instance. If a navigation starts at a "category", there is one query necessary to reach the Profile instance, at the center, and, after that, another query to reach precisely the desired WS at the *Service* instance, at the bottom left side of Figure 6.

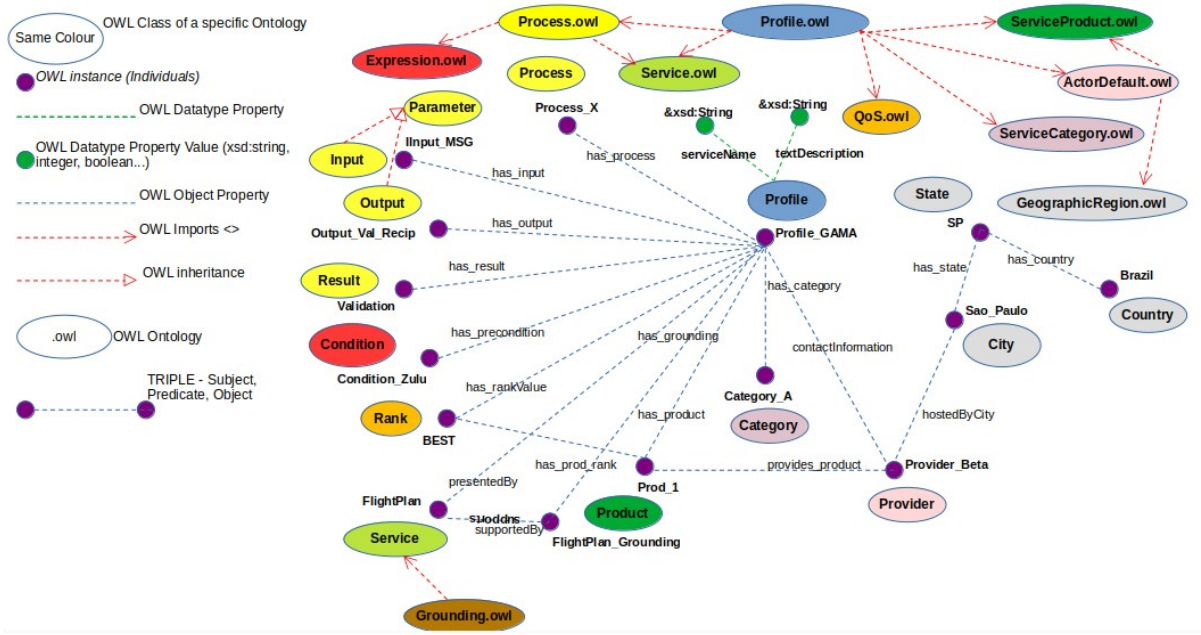

Figure 6: Conceptual model of the implementation of a single instance of WS and its connections [19]

Table 2 presents the translation of the CQ in Table 1 to SPARQL. Some CQ were translated to two queries and there are others which need three queries to reach the exact *Service* instance. It depends only on where (what node of the Graph) you start the navigation to reach the instance of *Service* at the bottom left of Figure 6. For example, if you start a way to the *Profile* instance at the center, starting from a "category" instance, it is near than starting from the City of Sao_Paulo_City, the distance determines directly how many queries are going to be executed to reach the desired WS at the end.

Table 2 - Competency Questions translated to SPARQL

*SELECT * WHERE {?subject prof:hasCategory* cat:***Category_A***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy* svc:***Profile_GAMA***} *Result:* ***[FlightPlan]***	*SELECT * WHERE {?subject prof:contact_Information* actor:***Provider_Beta***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy svc:**Profile_GAMA**} Result:* ***[FlightPlan]***
*SELECT * WHERE {?subject prof:hasProduct_BR* prod:***Prod_1***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy svc:**Profile_GAMA**} Result:* ***[FlightPlan]***	*SELECT * WHERE {?subject actor:hostedByCity:**SaoPaulo_City**} Result:* ***[Provider_Beta]*** *SELECT * WHERE {?subject prof:hasProvider* prof:***Provider_Beta***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy svc:**Profile_GAMA**} Result:* ***[FlightPlan]***
*SELECT * WHERE {?subject prof:hasCategory* resul:***Validation***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy svc:**Profile_GAMA**} Result:* ***[FlightPlan]***	*SELECT * WHERE {?subject prod:hasProd_rank* qos:***BEST***} *Result:* ***[Prod_1]*** *SELECT * WHERE {?subject prof:hasProduct_BR* prod:***Prod_1***} *Result:* ***[Profile_GAMA]*** *SELECT * WHERE {?subject svc:presentedBy svc:**Profile_GAMA**} Result:* ***[FlightPlan]***

It is possible to notice that the SPARQL codification shows how to do that navigation through Figure 6 instances passing parameters and reaching precisely an instance of the Class *Service*. The queries were executed using Protegé 5.5 and have shown precise results using this protocol. The same

instance of WS was found when querying the ontology to search for a unique WS. The ontology was validated considering the set of Competency Questions after the translation [15] and execution of the whole set of SPARQL queries and the execution of the semantic search has been proved shown precise results reaching a unique WS considering multi criteria queries. Figure 7 shows an example of a SPARQL query which was run in Protegé.

At this point the customized Brazilian OWL-S was totally implemented and ready to be validated. The next step of the ontology development methodology was to create an **Evaluation** mechanism to verify if the ontology was valid or not. An interesting point of this work is: the mechanism used to evaluate the ontology was the same to execute the semantic discovery: to translate the CQ of Table 1 into SPARQL queries and run them using the software Protegé to verify if the results were valid or not. At this point the ontology was completely implemented and it was possible to start the execution of the SPARQL queries as described in Table 2. Table 2 presents the CQ translated to SPARQL queries considering the PREFIXES of the customized OWL-S_BR as: PREFIX svc, prof, cat, prod, resul, actor, geog, qos: <*www.hildeproject.com#*>.

5.1. Results

The results of the experiments are related to the need to answer those questions mentioned in Table1. The questions represent the functional requirements which guided the whole ontology development process. The authors have translated those questions to SPARQL queries code, exactly as in [15]. They have taken the suggestions of this reference and it has made the task to translate the questions a guided work. After the translation was finished, they have run the whole set of queries using Protegé aiming to verify if the queries could bring the expected results needed to answer those CQ in Table 1.

To make service discovery, after finding the required Profile's instance it is necessary to find the instance of the OWL Class *Service*, which is the real Web Service and there is another association between Profile and Service Classes (in the *Service.owl* ontology), so, it is possible to discover the instance of Service connected to that Profile just by making another SPARQL query. Figure 7 presents how the SPARQL queries of Table 2 navigate through the RDF Graph reaching the WS.

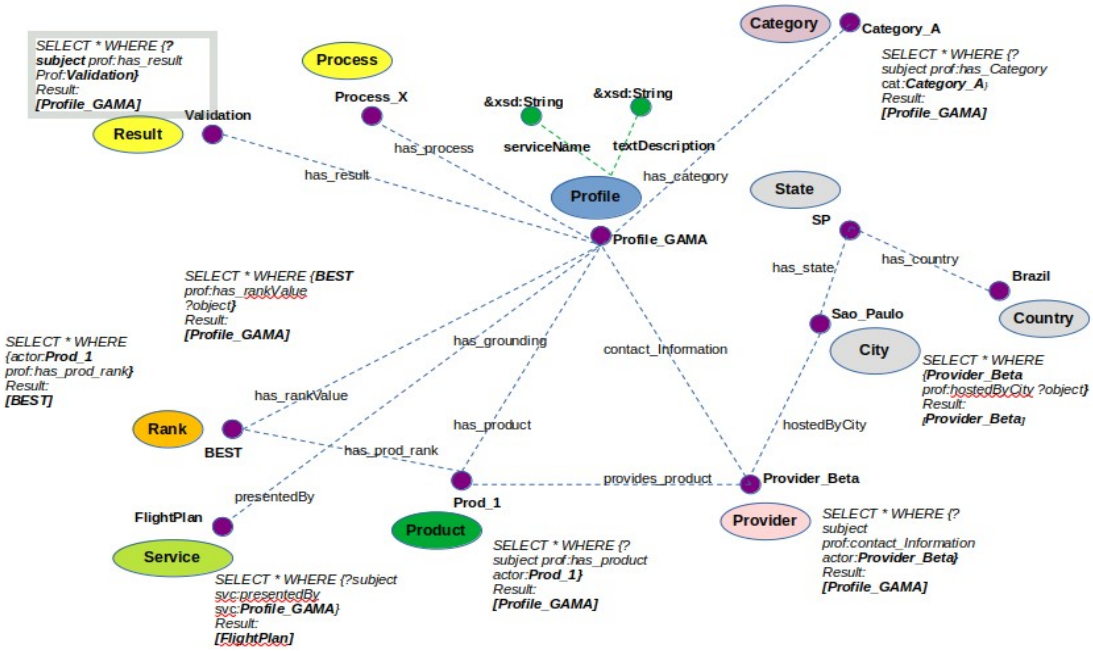

Figure 7: Execution of the SPARQL queries and navigation through the RDF

The **Evaluation** mechanism presented in Figure 7 shows how all CQ were translated to SPARQL queries and also how all of them were executed to navigate through the RDF Graph and check if the ontology is validated considering the CQ as a set of functional requirements to an ontology engineering process. The process of Evaluation has concluded the ontology is validated considering a set of functional requirements presented in Table 1 and translated to Sparql queries in Table 2.The last

phase of Methontology was the **Deployment** of the Brazilian ontology and the authors have published it at the address: http://www.hildeproject.com and, after that they consider that the goals of this work were reached.

6. Contributions and Future Works

In this paper we described an implementation of OWL-S which supports semantic discovery. We proposed a new solution for the Air-Traffic management knowledge based on an existing technology which allows developers to implement semantic descriptions and use them as machine-readable artifacts by programming languages API. The implementation considered a top-down approach to fill all levels of the architecture with instances and the twofold goals were reached with a customized OWL-S ontology and a set of Competency Questions used to establish functional requirements which could be checked for accomplishment as in Table 2, and a description of twenty aeronautical web services exactly as in Figure 3.

It is possible to mention some contributions:

- A new **extension** of the OWL-S architecture from W3C, which describes a Web Services Registry related to the global ATM's domain, including the Brazilian customized domain, that can be useful to FAA (SWIM, 2023), Eurocontrol (ICAO, 2023) and to the rest of the world's aviation information systems;
- A new Brazilian Aeronautical Web Services Description **Ontology**, able to describe semantically all the WS destined to Brazilian ATM information systems;
- A new **domain's ontology** to standardize the Brazilian aeronautical web services actual formal descriptions model and the ATM domain's vocabulary;
- A profound **reduction of the complexity** used by developers to describe and execute complex operations like *semantic discovery* over generic web services descriptions;
- A new **original method** to make ***semantic discovery*** considering multi criteria search's parameters.

Future works should support software engineering to build information systems using the artifacts generated by this work. The answers of the CQ could serve as messages to allow software actions based on semantic knowledge. This approach could bring intelligent software decisions to publish, advertise, discover and invoke any web service described by a standardized implementation of OWL-S like that one. This semantic intelligence could bring a less complex way to develop web services registry descriptions.

7. References

[1] ICAO - System Wide Information Management (SWIM), the International Civil Aviation Organization. www.icao.int. Access 06/1st/2022.

[2] SWIM - MANUAL ON SYSTEM WIDE INFORMATION MANAGEMENT (SWIM) CONCEPT. International Civil Aviation Organization. 999 Robert Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7. Website https://www.icao.int/APAC/Pages/swim.aspx. Access 06/1st/2022.

[3] W3C, World Wide Web Consortium, *WWW.W3C.COM. Access may, 30th, 2023.*

[4] UML2, The Unified Modeling Language Specification Version 2.5.1.UML®. Unified Modeling Language: https://www.omg.org/spec/UML/2.5.1/About-UML/. Access 06/01/2022.

[5] Guarino, N., et al. What Is an Ontology? 2009. ITSC-CNR, Laboratory for Applied Ontology, 38100 Trento, Italy, nicola.guarino@cnr.it.

[6] Panagiotis Bouros - Semantic Web Services: A conceptual comparison of OWL-S, WSMO and METEOR-S approaches. Technical Report. Department of Informatics and Telecommunications. National and Kapodistrian University of Athens (NKUA). Panepistimiopolis, T.Y.P.A. Buildings, GR-157 84 Ilisia, Athens, Greece. 2006. pbour@di.uoa.gr.

[7] Paolucci, M.; Kawamura, T.; Payne, T.R.;Sycara, K.. Semantic Matching of Web Services Capabilities - Carnegie Mellon University. Research & development Center, Toshiba Corp. 2006.

[8] Ziembicki, J.I.; Distributed Search in Semantic Web Service Discovery. A thesis presented to the University of Waterloo in fulfillment of the requirement for the degree of Master of Mathematics in Computer Science. Waterloo, Ontario, Canada, 2006.
[9] Almeida, J.F. - Um modelo de alinhamento de sistemas de comando e controle. Instituto Tecnológico de Aeronáutica. São José dos Campos, 2009.
[10] Pranav, K. - Service Matching based on OWL-S. Seminar Thesis Submitted to the Software Engineering Research Group in Partial Fulfillment of the Requirements for the Seminar Cloud Computing and Services by Pranav Kadam, Vogeliusweg 17, 33100 Paderborn. Paderborn, March 2013.
[11] Priyadharshini, G.; Gunasri R.; Saravana B. - A Survey on Semantic Web Service Discovery Methods. International Journal of Computer Applications (0975 – 8887) Volume 82 – No 11, November 2013. Tamil Nadu, India.
[12] Rohallah B., Ramdane M., Zaïdi S. - Semantic Web Service Discovery Based on Agents and Ontologies. International Journal of Innovation, Management and Technology, Vol. 3, No. 4, August 2012.
[13] Ashraf B. El-Sisi. - Fast Mapping Algorithm from WSDL to OWL-S. I.J. Information Technology and Computer Science, 2014, 09, 24-31 Published Online August 2014 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2014.09.03. Computer Science Dept., Faculty of Computers and Information, Menoufia University, Egypt.
[14] Nayak R., Lee B.: Web Service Discovery with additional Semantics and Clustering. In: IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), Silicon Valley, USA (2007).
[15] Potoniec J.; Wi D.; Ławrynowicz A.; Keet M. - Dataset of ontology competency questions to SPARQL-OWL queries translations. Data Article. Elsevier. Data in brief 29 (2020) 105098. Contents lists available at ScienceDirect - Data in brief: journal homepage: www.elsevier.com/locate/dib.
[16] Nawaz, F., Qadir, K., Farooq Ahmad, H.: SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish Subscribe Model. In: Fourth International Conference on Semantics, Knowledge and Grid. IEEE Xplore (2008).
[17] Marco L.S., David M., Claude M. - Discovering Semantic Web Services using SPARQL and Intelligent Agents - Hewlett-Packard Italy Innovation Center, Corso Trapani 16, 10139 Torino, Italy. 2012.
[18] Protegé. A free, open-source ontology editor and framework for building intelligent systems.https://protege.stanford.edu/. Access 06/1st/2022.
[19] Rodriguez, L.A.A.,Parente, J.M.O. - An Implementation of OWL-S to Support Semantic Web Services Discovery. Proceedings of FOMI2022: 12th International Workshop on Formal Ontologies meet Industry, September 12-15, 2022, Tarbes, France.
[20] Srinivasan, N.; Paolucci, M.; Sycara, K. Adding OWL-S to UDDI, implementation and throughput. Rasearch & Development Center, Robotics Institute, Carnegie Mellon University, USA. {{naveen,paolucci,katia}@cs.cmu.edu}.
[21] Martin, D. et al. Bringing Semantics to Web Services with OWL-S. Conference Paper at World Wide Web (2007) 10:243–277. DOI 10.1007/s11280-007-0033-x.
[22] Fielding, Roy T. Chapter 5: Rpresentational State Transfer (REST). Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.
[23] Rong, W., Liu, K.: A Survey of Context Aware Web Service Discovery: From User's Perspective. In: Fifth IEEE International Symposium on Service Oriented System Engineering (2010).
[24] Gomez-Péres, A., Fernández-López, M., Corcho, O. METHONTOLOGY: From Ontological Art towards Ontological Engineering. Publication Year: 2004 Conference/Journal: Springer Lecture Notes in Computer Science (LNCS), Vol. 3200 Pages: 17-31.
[25] Air Space Control Department, Brazilian Air Force, http://www.decea.gov.br.