World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:18, No:1, 2024

A Semantic Registry to Support Brazilian Aeronautical Web Services Operations

Luís Antonio de Almeida Rodriguez, José Maria Parente de Oliveira, Ednelson Oliveira

Abstract**—**In the last two decades, the world's aviation authorities have made several attempts to create consensus about a global and accepted approach for applying semantics to web services registry descriptions. This problem has led communities to face a fat and disorganized infrastructure to describe aeronautical web services. It is usual for developers to implement ad-hoc connections among consumers and providers and manually create non-standardized service compositions, which need some particular approach to compose and semantically discover a desired web service. Current practices are not precise and tend to focus on lightweight specifications of some parts of the OWL-S and embed them into syntactic descriptions (SOAP artifacts and OWL language). It is necessary to have the ability to manage the use of both technologies. This paper presents an implementation of the ontology OWL-S that describes a *Brazilian Aeronautical Web Service Registry,* which makes it able to publish, advertise, make multi-criteria semantic discovery aligned with the ideas of the System Wide Information Management (SWIM) Program, and invoke web services within the *Air Traffic Management* context. The proposal's best finding is a generic approach to describe semantic web services. The paper also presents a set of functional requirements to guide the ontology development and to compare them to the results to validate the implementation of the OWL-S Ontology.

Keywords**—**Aeronautical Web Services, OWL-S, Semantic Web Services Discovery, Ontologies.

I. Introduction

THE result of the evolution of the internet in the last two decades is a huge amount of stored information that must be understood by humans and information systems to reach ordinary goals like using an online library or buying bus tickets. Because of this excessive growth of data, IT professionals all around the globe must deal with this actual web full of data establishing strategies for organizing, accessing, and interpreting information that is updated every day.

One of the several points of convergence of Semantic Web was defined by the introduction of SOA's paradigms because it addresses the major challenges of becoming a structured Web. Tim Berners Lee had a long-range vision about a Utopian Se-mantic Web [18] and even he had thought about one of the most important initiatives in the evolution of the World Wide Web: - the development of standardized "services for web", or, the popular Web Services (WS). WS can easily provide ordinary users with small portions of information but they can also make available much more complex operations like buying tickets for a Metallica's concert and the needs related, like to rent a car, to book a hotel room or flying tickets, offering a basis [13] for interoperability between providers and consumers using a reliable exchange of standardized XML messages, and it is the main reason for the spread of the use of it.

Nowadays the common standards for implementing WS tend to focus on interoperability among different programming languages but their syntactic descriptions, based on XML, fail to provide a consistent basis to allow the automation of a sort of tasks like to make intelligent discovery of services or service compositions based on the service's functional and non-functional properties descriptions [5]. Current standards for these descriptions offer artifacts like WSDL, from SOAP, where developers can describe a set of characteristics about a service aiming to make a specific service's description different from the others, and allowing these XML specifications to be syntactically queried, offering to an information system or to a web user some answers which should make them able to take the decision to choose (or not) this service among several others.

At this point, the spread of the use of the WS has provoked a growing and disorganized infrastructure where developers are very sure it is a good idea to build software using the WS paradigm but, at the same time, the ways to publish WS, to make use of other publishers' ones and to discover WS among distributed servers and with no standardized descriptions are very serious problems [11]. The sharing of WS is an intelligent facet of SOA since if there is a WS located in another server which can satisfy a company's daily routine, why cannot the company's developers team implement their information systems making use of this shared WS? The need to share and reuse WS has made developers communities to face obstacles to publish, advertise and discover WS in a non-standardized environment.

The Semantic Web Study Groups have tried to fix the lack of meaning in the XML documents by creating machine-readable languages like RDF, RDFSchema and OWL, a set of ontology's representation languages [19]. If we have to XML tags with the same "label", it is impossible to make them different from each other because the XML syntax has no meaning, it is only about "tags" and that is the difference between it and those languages. For trying to fix the lack of meaning in the WSDL artifacts, some authors [2] have proposed an intelligent semantic layer to better describe the WS features, aiming to facilitate automation to discover, advertise, compose, monitor and call the execution of these services.

L. A. A. R., J.M.P.O., and E.O. are with the Aeronautics Institute of Technology, São José dos Campos-SP, 12200-000, Brazil (phone: +551239470000; e-mail: rodriguezlaar@gmail.com, parente@ita.br, oliveirae@fab.mil.br).

Several work groups were made to propose frameworks and machine-readable languages for WS to W3C, which recommends OWL-S (Web Ontology Language for Services) [7], WSMO (Web Service Modeling Ontology) and SAWSDL (Semantic Annotations for WSDL and XML Schema) [4]. All of these standards can compose a semantic artificial intelligence layer and these machine-readable languages are able for the creation of semantic descriptions which can be manipulated by programming languages' API. This way it is possible to build semantic artifacts which could describe an infrastructure able to offer the automation necessary to make the important mentioned tasks.

Some works which try to propose semantic descriptions into WS registries are strongly coupled to existent technologies like the WSDL [11] and the idea is to insert into the XML syntax small portions of a markup language to identify, among a set of services, the ones which get closer to satisfy requirements by using matching techniques and algorithms. These works were focused on mixing OWL-S specifications with the WSDL and UDDI, aiming to support some level of automation based on semantics to identify precisely something about the required WS based on those specifications. It is a huge job since developers must learn and manage both technologies (SOAP, OWL, OWL-S) [13].

Effective WS operations (publish, advertise, discover and invoking) must focus on semantics to be precise and depend directly on the semantic ability [5] to make specifications about the WS. OWL-S [7], [19] is an ontology designed to do that by using OWL language syntax and it is a powerful mechanism to build semantic descriptions, artifacts that could be semantically manipulated and these manipulations could find objects in a precise way, which could define what is the exact WS which is able to accomplish a set of user's requirements. Its high level of expressiveness allows developers to build semantic WS repositories, or WS registries [13], using it as a model to describe conceptual features in a machine-readable language, which makes possible the automation to make intelligent WS operations. The OWL-S also provides [5] a platform to categorize several different types of criteria to make advertisements and semantic searches on its WS descriptions.

This paper presents an implementation model of a WS' Semantic Registry based on the OWL-S reference ontology architecture to support Brazilian Aeronautical WS CRUD Operations (Create, Retrieve, Update and Delete). The paper is organized as follows: Section II presents some characteristics of the ontologies for artificial intelligence, the dorsal spin of this job, and the OWL-S, the Ontology for WS. Section III presents the current approaches to describe WS registries and to automate tasks, Section IV presents the Semantic Web Services Registry implementation and a set of experiments and their results using Protegé [12] graphical interface and a built-in-Python information system to execute CRUD operations in a WS Semantic Registry published on the internet. Section V presents the contributions and further works.

II. Ontologies of Artificial Intelligence

In metaphysics, Ontology is the philosophical study of being, exactly like some concepts such as the existence or the reality [16]. The ontologies lead us to think about how entities are grouped or classified into categories [5], what kind of relationships are essential for each one of them and how does it interfere in the domain they are inserted. Ontologies are being used in computer science and in knowledge engineering [16] for a long time and for different purposes, like conceptual modeling of domains, like standards to share information using syntactic data exchange [16] or as a kernel of domains' descriptions, allowing developers to build huge accessible information sources which can interact with software agents [14] in a new paradigm to build information systems [5]. The Ontologies of Artificial Intelligence are logical artifacts, they are built-in common text files, exactly as any ".txt" but they are written using XML format and a formal ontology representation language [19].

With the essence of the Semantic Web the ontologies, with their meaningful and structured contents, are able to offer intelligent information to software agents, roaming from page to page of the WWW to readily carry out more sophisticated tasks. From an infrastructure's perspective, the Semantic Web has made possible the traditional web has experienced [5] a further extension, represented by the Internet of Things (IoT), feasible today thanks to a huge effort of the communities to advertise the use of ontologies. The main cell of an ontology is the triple, composed of a Subject-Predicate-Object relationship and an ontology is a natural [19] triple repository. Exactly like a text file, or a ".xml" file, both can be read by programming languages' API, like Java or Python and this process can store the content of the files by using these standards and use it as an information source, executing CRUD operations and offering a basis to make syntactic data exchange.

The most important contribution of the Ontologies of Artificial Intelligence for the WWW is the ability to offer structured information [16] which enables a more efficient machine-to-machine cooperation through them. To achieve this, they represent the most suitable tool to enable transfer and comprehension of information among software applications, even those designed and developed by unrelated people in different places [5]. The ontologies make it possible to surf a sea of knowledge [16] available today without human intervention, making the real meaning of the web of things. This paradigm uncovers new horizons for WS and SOA to build really challenging applications [16], like new kind of WS, bridging the virtual and physical dimensions through the real life with humans' ordinary goals being achieved.

A. The OWL-S Ontology for WS Descriptions

This integrated vision of the WWW using ontologies should enable full access not only to content, but also to services on the Web [13] or the WS. The idea is about users and software agents being able to discover, invoke, compose and execute services with a high degree of automation [5]. Developers can use ontologies to model, to formalize data structures, to define domain's vocabulary and the fact that ontologies are

implemented using machine-readable languages makes them not only modeling artifacts, but, an accessible semantic layer [4] to compose software, able to specify business rules, data structure, domain's dialects and Web Services Registries descriptions, becoming a basis to make semantic data exchange. OWL-S is the Ontology for Services from W3C which makes these actions possible. It is called ontology, language, dialect [19] or architecture of ontologies by developer communities.

The OWL-S ontology architecture is composed of several ".owl" files which represent, each one, a different Ontology. These ontologies make use of each other by importing its contents using standardized commands and it proposes a reliable communication and reuse of entities among them. The overall structure of the Upper level of the OWL-S is presented in Fig. 1 and suggests different perspectives [8]:

- A Class diagram represents the content of the Service ontology, represented by the Service.owl file from the original release [7];
- There are four OWL Classes into the *Service.owl ontology*:
  - the Class *Service*, which defines the primitive type - Web Service;
  - the Class *ServiceProfile*, which makes the WS's advertisement to providers and requesters and makes it possible to do semantic discovery and other abilities
  - the Class *ServiceModel*, which gives a detailed description of the WS's operations considering a process view and
  - the Class *ServiceGrounding*, which is responsible for a complete description of how to deal physically with the services via protocols and messages.

OWL-S was built on OWL language stereotypes and it is a standard developed by the Web-Ontology Working Group [19].

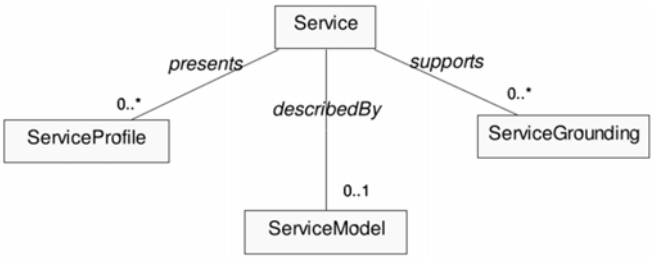

Fig. 1 The upper level of OWL-S ontology [8]

Fig. 1 presents the *Service.owl* ontology which encapsulates four OWL Classes and all the relationships among them. Martin et al. [5] present an approach destined to create a WS description's repository very similar to a WSDL, but composed of semantic descriptions of WS using the OWL-S. The approach starts by populating these Classes with the creation of owl:Individuals of the Service Class, representing each one a specific WS. These owl:Individuals will compose at the end of the process, each one, a huge RDF Graph [19] exactly as in Fig. 6, at almost the end of this paper, full of connections among different ontologies, Classes and Individuals [13] and granting a detailed semantic description of each WS which presents functional and non-functional characteristics. This huge set of RDF Graphs describes business rules, laws and a sort of other visions destined to describe all the features about a WS and to make it possible to have a semantic description's foundation capable of more complex tasks, like service discovery and service composition [13].

Fig. 2 is a UML 2 Class Diagram [18] and it presents the ontology *ServiceProfile.owl*, a specialization's level destined to make the *Advertisement of WS* of the OWL-S using a sort of entities presented at the original implementation, like the ontology *ServiceCategory.owl*, or the Classes *Parameter, Input, Output, Process* and *Product*. These are all a set of criteria to standardize the tasks to publish, advertise, search and to discover WS precisely and using specific features [8]. In order to describe the functionality of a service, the OWL-S architecture proposes several specifications of those Classes from the upper-level ontology.

The specializations of the Classes from the upper level also present other UML2 diagrams which have the same idea of Fig. 2, but applied to present the essence of each other two Classes from the Upper level, the *ServiceModel* and *ServiceGrounding*, through the two ontologies named *Process.owl* and *Grounding.owl* [5]. Fig. 2 also presents the non-functional properties of the WS like the *Provider*, functional elements and additional properties, for example: *preconditions,* which describe particular characteristics of the WS. This semantic description also provides some human-readable information about the WS like its name (*serviceName*), working requirements (*textDescription*) and the mechanism to refer to humans or companies responsible for that service (*contactInformation*).

The *Profile.owl's* specialization of the Upper level presents how the original OWL-S can handle the issues of information transformation [5] by representing a huge set of features about the WS like the *inputs* necessary to invoke and the *output*, or the delivery. The behavior and the change of state caused by the execution of the service, like the preconditions and effects are also mapped to *Profile.owl's* connections. This denotes the connection between Profile and the *ServiceModel* set of instances, by the corresponding *Process*. Each Profile's instance must have an association to an instance of each other connected entity, as presented in Fig. 2. The relationships define specific properties and characteristics, like the *Category* and the *Product*, generated by the WS. The entities connected to Profile are native OWL stereotypes meaning each one presents some type of a specific characteristic which is unique for the description of that instance of Profile. Observing this powerful way to establish semantic descriptions it is possible to realize that syntactic knowledge becomes semantic knowledge.

It is possible to build a complete customized implementation of OWL-S ontology architecture by filling each ontology with a set of *owl:Individuals* [5]. Each owl:Individual, connected to another one from another owl

Class using a specific *ObjectProperty* or, connected to a specific *serviceName* and t*extDescription* which are *DatatypeProperties* [19], provides a 'predicate' or an 'object' of a triple able to be queried.

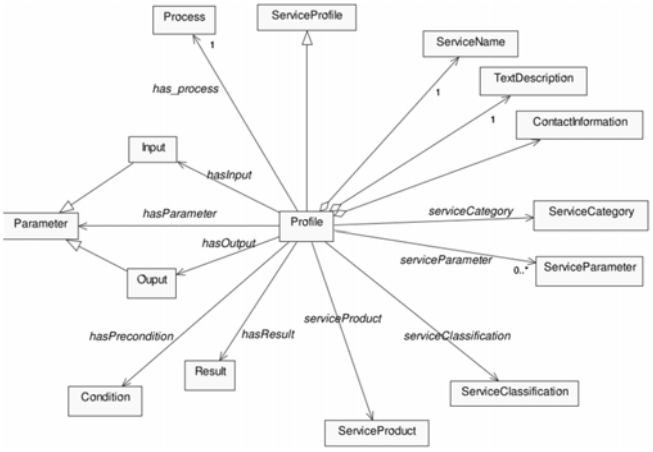

Fig. 2 Profile.owl's specialization level [8]

The OWL-S ontology architecture and implementation allow developers to literally build a Semantic Registry, and it is possible to make queries by using SPARQL [6] and obtain precise answers, which could be used like a communication unique protocol for software agents to execute their actions and to accomplish their goals.

III. Other Approaches to Describe WS Registries

To identify and choose a service is necessary to match words syntactically, such as a service name or a service text description which are written using WS registry's specifications standards like WSDL or UDDI [8]. Different techniques have been proposed to describe registries in a semantic level [15] which can improve the accuracy of WS descriptions: data mining, mapping algorithms, ontology based information systems and service description [20] and all of them have reached the conclusion that the levels of the WS registry's descriptions can reach can offer [15]: Syntactic matching, where the similarity of data is found using syntax driven techniques and the Semantic matching, where the key idea is the mapping of meanings between concepts.

Pranav [10] presents a different registry's description which enables service discovery using a match based on non-functional parameters like a standardized description of WS providers, which advertises their services 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.

Almeida [1] presents in his thesis a reference model to the Brazilian military command and control systems. The idea is a centralized and structured model of specification led by a reference ontology recommended by OTAN, the JC3IEDM, which allows dynamic compositions of WS 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.

Marco et al. [6] describe an approach for the description and discovery of semantic WS using SPARQL and software agents. They propose language to describe the preconditions and post conditions of WS 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.

Rodriguez and Parente [13] have published a paper that proposes an implementation of OWL-S to support semantic discovery for Brazilian Air-Traffic Management Web Services. This work has presented how to fill the original OWL-S ontologies with instances and all those ideas were applied at this work.

IV. An Implementation Model of the OWL-S to Support a Brazilian Aeronautical WS Registry Description

The idea behind this work was to make an experiment which could be coherent with SWIM SDCM [17] specifications and its semantic models based on OWL-S. Brazilian aviation agencies are running to build formal vocabularies, ontologies and semantic descriptions that could serve as a machine-readable knowledge to support intelligent information systems. The goal is for Brazil to reach interoperability with all the nations aligned with the SWIM and to establish Brazilian aviation with its own WS semantic description models. Thus, we can state the following problem: The lack of semantics in the Brazilian Aeronautical WS's registry descriptions destined to support air-traffic management (ATM) information systems. Every day ATM's information systems have a high computational cost to publish, to share, to make use or, a simple task like to discover a WS in a set of syntactic descriptions. There is no complete solution for implementing repositories, or registries, of WS where developers could have a complete semantic description's foundation [13] which could be able to describe the WS with a formalism's level which allows software agents to interact with the registry making automatic tasks like semantic searches or WS compositions.

Following W3C recommendations and also ideas like Martin's [5] implementation, a solution was proposed for the problem: To implement a customized OWL-S ontology to act as a WS' Registry. The goals are twofold:

1. To develop a WS registry by implementing a customized OWL-S, based on a set of Competency Questions (CQ) [9] to define the functional requirements for the *Brazilian OWL-S ontology architecture*. The ontology itself is the registry, or, the WS repository and it can be accessed by any software agents from anywhere in the internet, it is going to be published in the WWW and composed by

several aeronautical WS semantic descriptions coupled to the OWL-S original ontology, but also some new entities created to abide by Brazilian aeronautical domain and laws;

2. The set of CQ will be translated [9] to SPARQL queries and they will be run [6] using a Python Web-based software built for this scientific work and destined to execute the semantic searches for WS to compare the results with the CQ, to make a complete check of requirements accomplishment with answers for the CQ. The SPARQL queries will be executed to find specific WS Individuals (which represents, each one, a unique WS) described in the Brazilian OWL-S ontology architecture, by applying on the semantic search the criteria used to standardize WS operations factors:

criteria = {Category, Name, Result, Input, Output, Process, Condition, Provider, Geographic Region, Expression (SPARQL text), QoS Rank, EndPoint, Protocol, IP_Adress}

To go for the first goal and exactly like the method and data available in Mendeley Data [9], a set of CQ were created to define all the necessary information to explain the connections shown in Figs. 1 and 2. These questions are related to all the functional and non-functional characteristics about all WS and their relationships defined into the Brazilian ontology architecture. For a matter of available space in this paper, a small part of these questions is presented in Table I and the namespace of the Brazilian ontology architecture is published at [22].

TABLE I
A SMALL PORTION OF THE COMPETENCY QUESTIONS

What is the WS which is associated with a specific ServiceCategory_BR?	What is the WS which is associated with a specific ActorDefault_BR which is a Provider?
What is the WS which is associated with a specific Process?	What is the WS which is associated with a specific City?
What is the WS which is associated with a specific Product?	What is the WS which is associated with a specific FinalRank?

Extra entities were created for customizing Brazilian laws and aviation rules, like a class called *owl:Class SparqlConditionExpression*, to be related to the idea to deal with the SPARQL protocol for acting as a message flow to be used to redirect software decisions. The next goal was to begin the registry's descriptions by the creation of each WS which would compose it by creating the WS in a top-down approach [5] through the *Service.owl ontology* and the implementation of named *owl:Individuals* for the owl:Classes: *Service, ServiceModel, ServiceProfile and ServiceGrounding* which would compose the whole set of WS, exactly as presented in Fig. 1. The software Protegé 5.5 [12] was adopted to accomplish this task.

A set of 22 instances (*owl:Individuals*) of Brazilian WS were created to populate the owl:Class *Service* and they represent, each one, a specific Brazilian Aeronautical Web Service exactly as they are defined in real life [21]. After the creation of the instances of the Class *Service*, for each one of them, there is an implementation for the corresponding instance of the Classes *ServiceModel, ServiceProfile and ServiceGrounding*, and also the respective connections among each other exactly as in Fig. 1, using those associations. Fig. 3 presents the owl Class *Service* at the left side, highlighted, and the set of WS already created at the right side of Protegé's graphical interface.

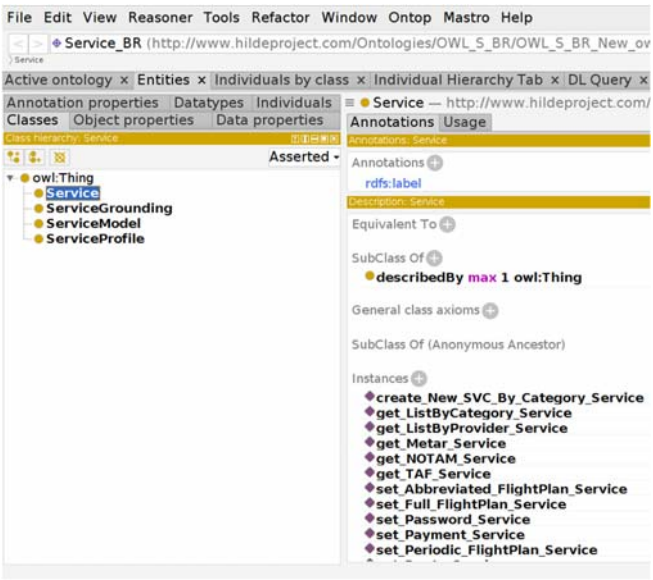

Fig. 3 Instances of the Class Service

Fig. 4 presents, in blue color, the original *ObjectProperties* of OWL-S being used to associate one specific WS, the *owl:Individual get_Metar_Service*, to three other ones: associated to *get_Metar_Profile, get_Metar_Process and get_Metar_Grounding*, as seen at the right side of Protegé. After doing that for all the WS, the upper-level implementation was all set and the next goal was to create the population of the *Profile.owl, Process.owl and Grounding.owl,* the second level's ontologies, to describe the semantic foundation which would allow automatic tasks like semantic discovery and composition of WS. To implement the *Profile.owl* ontology of Brazilian ontology architecture, another set of *owl:Individuals* was inserted exactly as presented in Fig. 2.

A triple repository was created for each instance of Profile which is a multi-criteria way to find any WS by using the SPARQL protocol and by passing one element of criteria as a "search" parameter, exactly like goal number 2. After finding the required Profile's instance, it has another association with an instance of the ontology *Service.owl* in the upper level, so, it is possible to discover its instance of Service (which is a WS) connected to that Profile just by making another SPARQL query. The complete set of a single instance's associations can be seen in Fig. 5.

Fig. 5 presents the whole set of implemented connections between instances of different Ontologies and different Classes necessary to describe *ONE WS* of the Brazilian ontology architecture. To finish the implementation, the *Process.owl and Grounding.owl* specialization's ontologies

were filled with *owl:Individuals* and their relationships are the same way as in Fig. 2. These two 2nd levels of specialized ontologies serve as "*How to access*" the WS and "*how to make physical access*" to the WS, describing the process to access the WS and the physical information, like an Endpoint IP address. After that owl:Classes destined to define the *Atomic and the Composite* WS and the dependency's relationship between them were implemented as triples, allowing to sort and associate the atomic services which will compose a more complex composite WS. These detailed specializations and their diagrams can be seen in [5]

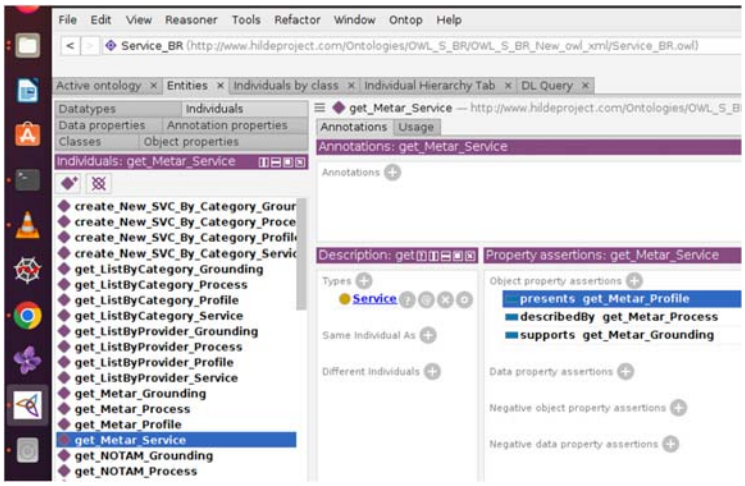

Fig. 4 Associations for the WS get_Metar_Service

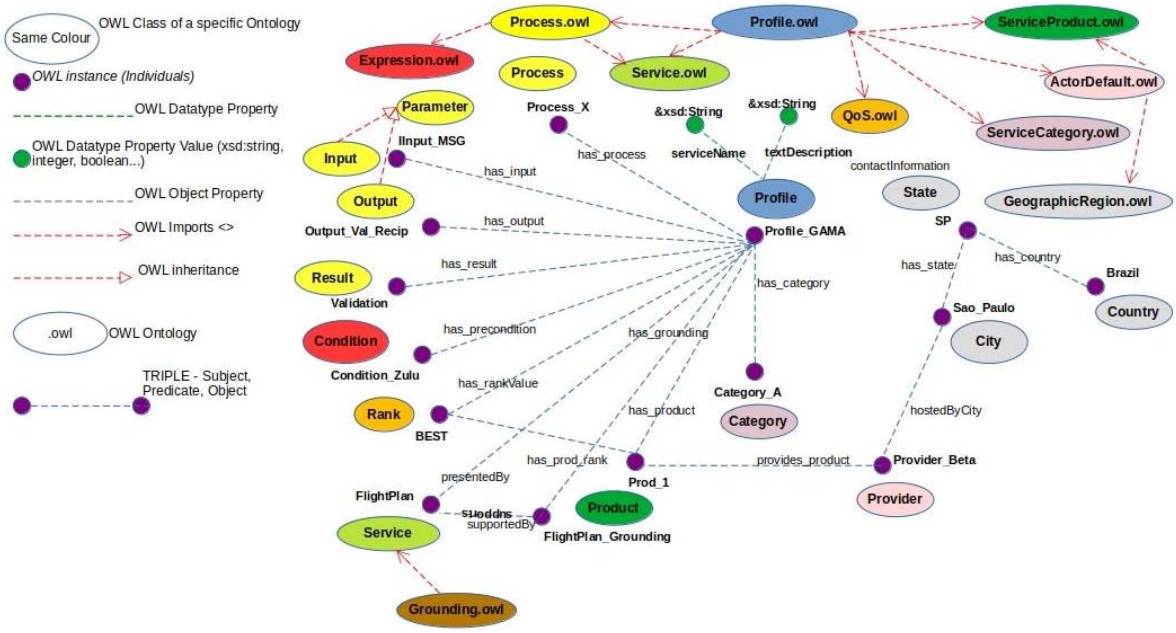

Fig. 5 Implementation of a single instance of WS and its connections

It is very clear to observe in Fig. 5 the advertisement's feature about the instances of Profile Class which can connect it to several other classes' instances to reach a specific WS. It is possible to run one SPARQL query from a criteria entity to find a Profile's instance by choosing one of the peripheral factors cited on goal number 2 and presented in Fig. 2, and it is also possible to run another query to find the instance of Service, which is connected to *THAT* Profile's instance, like in Fig. 5, reaching the goal of describe a semantic foundation composed by several RDF Graphs which, reunited, form the Brazilian ontology architecture, a WS' Registry able to act as an intelligent source of information, or, a kernel of semantic artificial intelligence.

At this point, the whole Brazilian ontology architecture implementation was complete; it was possible to publish it on the internet at [22] and also start the execution of the CQ as described in Table I to check if the ontology is satisfying the functional requirements established for its development. The new Brazilian WS registry offers an accessible information kernel to create, publish, advertise, update, retrieve, delete and

call any kind of distributed standardized WS, but it does not execute any kind of WS.

We have presented the translation of the whole set of CQ to SPARQL code and it was made exactly as Potoniec has proposed [9]. The instances used as examples in the queries' codes were based on Fig. 5, and the idea is start from any instance from the peripheral area of Fig. 5 and navigate from it, using SPARQL queries, to reach the instance of the Class *Service* passing through the *Profile_GAMA* instance (Fig. 5). Table II presents the sequence of CQ being translated and executed on the internet to discover the next instance, and using the instance before as a "search" parameter.

TABLE II
CQ CONVERTED INTO SPARQL QUERIES TO BE ANSWERED AND TO VALIDATE THE FUNCTIONAL REQUIREMENTS

What is the WS which is associated with a specific ServiceCategory_BR?	**Start**: Category_A → **SPARQL_1** → Profile_GAMA → **SPARQL_2** →FlightPlan	**SPARQL_1**: SELECT ? WHERE {? **has_category**Category_A} → Query result = Profile_GAMA **SPARQL_2**: SELECT ? WHERE {? **presentedBy**Profile_GAMA} → Query result = FlightPlan (WS) (This is the WS we want to find)
What is the WS which is associated with a specific Process?	**Start**: Process_X → **SPARQL_1** → Profile_GAMA → **SPARQL_2** →FlightPlan	**SPARQL_1**: SELECT ? WHERE {? **has_process**Process_X} → Query result = Profile_GAMA **SPARQL_2**: SELECT ? WHERE { ?**presentedBy**Profile_GAMA} → Query result = FlightPlan (WS)
What is the WS which is associated with a specific ActorDefault_BR which is a PROVIDER?	**Start**: Provider_Beta → **SPARQL_1** → Profile_GAMA → **SPARQL_2** →FlightPlan	**SPARQL_1**: SELECT ? WHERE {? **contactInformation**Provider_1} → Query result = Profile_GAMA **SPARQL_2**: SELECT ? WHERE { ?**presentedBy**Profile_GAMA} → Query result = FlightPlan (WS)
What is the WS which is associated with a specific Product?	**Start**: Prod_1 → **SPARQL_1** → Profile_GAMA → **SPARQL_2** →FlightPlan	**SPARQL_1**: SELECT? WHERE {? **has_product**Prod_1} → Query result = Profile_GAMA **SPARQL_2**: SELECT ? WHERE { ?**presentedBy**Profile_GAMA} → Query result = FlightPlan (WS)
What is the WS which is associated with a specific FinalRank ?	**Start**: BEST → **SPARQL_1** → Profile_GAMA → **SPARQL_2** →FlightPlan	**SPARQL_1**: SELECT ? WHERE {? **has_rankValue**BEST} → Query result = Profile_GAMA **SPARQL_2**: SELECT ? WHERE { ?**presentedBy**Profile_GAMA} → Query result = FlightPlan (WS)
What is the WS which is associated with a specific City?	**Start**: Sao_Paulo → **SPARQL_1** → Provider_Beta → **SPARQL_2** →Prod_1 →**SPARQL_3** → Profile_GAMA →**SPARQL_4** → FlightPlan	**SPARQL_1**: SELECT ? WHERE {? **hostedByCity**Sao_Paulo} → Query result = Provider_Beta **SPARQL_2**: SELECT ? WHERE { Provider_Beta**provides_product ?**} → Query result = Prod_1 **SPARQL_3**: SELECT ? WHERE {? **has_product**Prod_1} → Query result = Profile_GAMA **SPARQL_4**: SELECT ? WHERE { ?**presentedBy**Profile_GAMA} → Query result = FlightPlan (WS)

It is possible to see in Table II that the SPARQL queries formulated for the experiment answer all the set of questions, meaning the ontology is validated considering Table I which presents the set of the functional requirements to build it. All the partial answers for the queries are *owl:Individuals* which can confirm the correctness for the ATM domain's representation, which can be also considered as a formal vocabulary to make data exchange. The SPARQL's answers validate the functional requirements and the next step was to build a Python Web-based system to present some features of the semantic registry to show it is useful.

V.A PYTHON WEB-BASED INFORMATION SYSTEM

The Brazilian ontology architecture itself is the semantic description which offers a semantic foundation to be accessed and manipulated as an intelligent information's kernel and to acquire the ability to create, edit, publish, advertise WS and call an external execution of the WS previously registered into this semantic registry. The goal of this Python system is to present a particular vision about how to use the Brazilian ontology architecture as a WS' REGISTRY, which it is possible to interact with to execute common WS Providers and Requester's operations, like to publish, advertise, to list, to make semantic discovery and to call a WS from anywhere in the internet.

The *SWIM-Brazil App* is a concept's proof app which offers a software implementation model able to access any semantic registry implemented like the Brazilian ontology architecture. The system offers a set of functionalities which makes a user able to make login, to list all registered WS, to make semantic discovery of WS obtaining precise results (finding a single WS) and using a multi criteria approach coherent with the original OWL-S's implementation. After semantically discovering the WS, the human user or an external app can "call" the execution of this WS no matter where it is published on the internet.

We have chosen the platform Python 3.10.6, the framework Django 4.1 [3] interacting with the owlready 0.38 to act as a persistence layer manipulating the ontologies, the Apache Web Server to store and make the ontologies accessible and the middleware OpenLDAP 2.5.14 for the authentication. Python 3 was chosen to be able to interact with the latest version of owlready2, which allows the manipulation of the OWL-S ontologies in a non-verbose approach. With a view heading to enabling capillarity by accessing via browser to human users and via network to information systems, the SWIM-Brazil's App was designed as a web application.

Usually, Django has its own persistence framework and because of its easy ways to make integration with a sort of middleware, we have changed the persistence's layer framework to use the Python's library owlready2 0.38 and data will be persisted into the Brazilian ontology architecture.

The SWIM-Brazil App web application works according to the scheme described in Fig. 6, which is a little bit different from the original Django architecture [3]: (1) a common user via browser sends an http request to the SWIM-Brazil App; (2) the App server forwards the request to the Apache Web Server, aiming to reach the Brazilian ontology architecture which is stored into this server; (3) the URL setting contained in the urls.py file selects the user's view according to the url specified in the request; (4) the view communicates with the Brazilian ontology architecture) via models.py, renders the html or other format using templates and returns the http response to the App server; (5) finally, the App server returns the requested page to the browser.

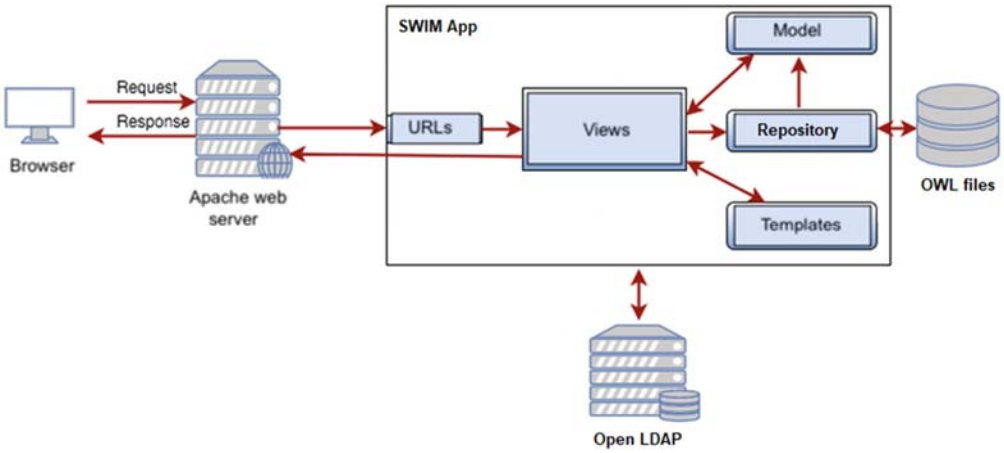

Fig. 6 SWIM-Brazil App architecture (adapted from [3])

An important component of the Django's architecture is the Model, which represents the access of the domain concepts and the encapsulation of the traditional relational database, according to the *Active Record's* design pattern. In the SWIM-Brazil's App this access to a database is not used, the app persists data into the ***Brazilian ontology architecture***, where the tasks of *reading and writing* are made using the owlready2 library.

SWIM-Brazil App is a modular monolith application composed by two modules, exactly as the Django's rules. There are two extension points to connect the next modules of the software. To separate the data manipulation levels, the business rules data are going to be persisted into the Brazilian ontology architecture and the information about authentication is going to be persisted into a LDAP directory service, using OpenLDAP.

Fig. 7 presents two SWIM-Brazil App's web pages destined to allow the user to make semantic discovery according to the search criteria of goal number 2 on this paper and a second web page destined to "search by category criteria". Fig. 8 presents another web page where it is possible to see the semantic search result and the name of the WS to be clicked and call its execution.

Fig. 7 presents the "Criteria of Service", which is the multi criteria options to semantically search for a Web Service. If the User chooses the search by category criteria, the "Service Category" result is presented containing all the specific categories able to be chosen to look for a Web Service. There are different abstraction levels to define this set of categories and they can show a list of services or a specific one, it depends on that abstraction level.

It is possible to execute some efficient automated tasks like semantic search and discovery using several criteria, allowing a common web user to search for web services considering different possible ways to find them. The software developed for this article is a proof of concept and there is a lot of things to still implement.

VI. Contributions and Further Works

In this paper we have described an implementation model of OWL-S which supports Brazilian Aeronautical Web Services. We have proposed a solution for the Air-Traffic WS registries' management based on an existing technology which makes developers able 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 set of Competency Questions destined to establish functional requirements which could be checked for accomplishment as in Table I, and a complete description of twenty aeronautical WS exactly as in Fig. 5.

As contributions, it is possible to cite the Brazilian ontology architecture as an *extension* of the W3C's OWL-S technology, customized to attend the Brazilian ATM's WS domain. Also, a *Brazilian ATM's WS formal description model* was created and it is able to offer some level of interoperability with the nations aligned with the SWIM. Also it is possible to cite the *Brazilian ATM WS' vocabulary* which is going to be an opportunity to standardize the Brazilian ATM's information systems.

Further works could be related to encapsulating the Brazilian ontology architecture and building ATM information systems around it.

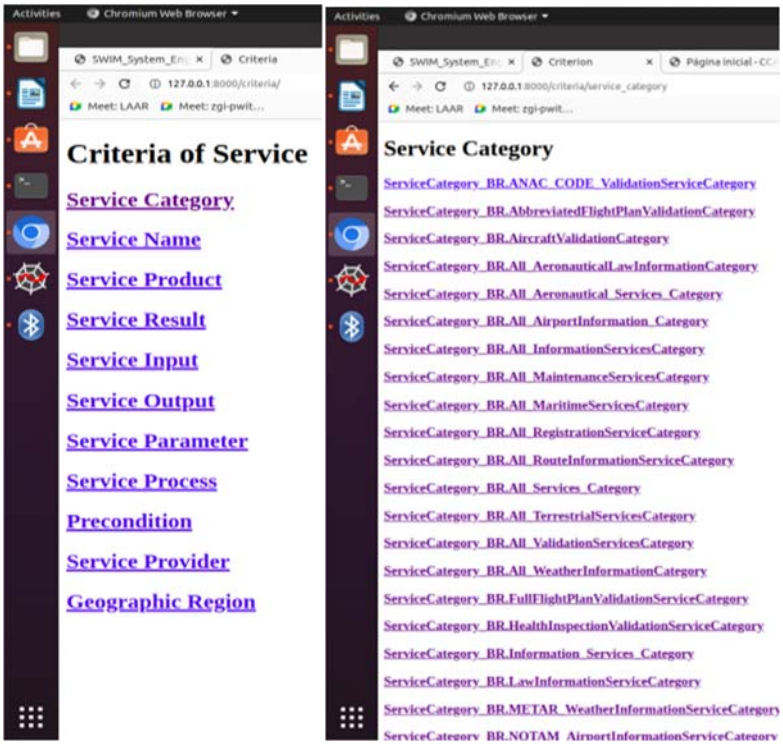

Fig. 7 SWIM-Brazil App semantic search by criteria and search by Category

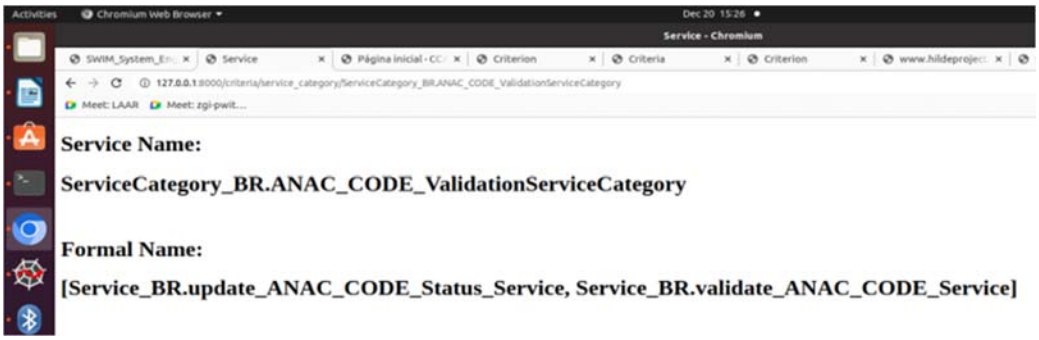

Fig. 8 SWIM-Brazil App semantic search result

REFERENCES

[1] Almeida, J.F. - Um modelo de alinhamento de sistemas de comando e controle. Instituto Tecnológico de Aeronáutica. São José dos Campos, 2009.

[2] Antoniou, G; van Harmelen, F. A Semantic Web Primer. 2. ed. London: The MIT Press, 2008. 287p.

[3] DJANGO. Python framework, http://www.djangoproject.com. Access 03/30/2023.

[4] Kopecky, J. SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Computer Society, 2007.

[5] 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.

[6] 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.

[7] OWL-S - Semantic Markup for Web Services. https://www.w3.org/Submission/OWL-S/. Access 06/01/2023.

[8] 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.

[9] 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.

[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] Protegé. A free, open-source ontology editor and framework for building intelligent systems. https://protege.stanford.edu/. Access 06/1st/2022.

[13] 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.

[14] Rodriguez, L.A.A., Parente, J.M.O. - An Ontology to support Brazilian Traffic Flow Management Based on NASA's ATM Reference Model. SITRAER 2022 - International Air Transportation Symposium, São José dos Campos, Brasil.

[15] Rodriguez, L.A.A., Parente, J.M.O. - An Approach to Support Semantic Discovery Using Ontologies to Describe Aeronautical Web Services Repositories. Proceedings of ONTOBRAS 2023, Brasília - DF, Brasil.
[16] 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.
[17] 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).
[18] Salvatore, G., Giuseppe, L.R.- Advances onto the Internet of Things: How Ontologies Make the Internet of Things Meaningful. Advances in Intelligent Systems and Computing, Volume 260. Polish Academy of Sciences, Warsaw, Poland.
[19] 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/2023.
[20] 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 06/01/2023.
[21] W3C. World Wide Web Consortium. http://www.w3.org. Access 06/1st/2022.
[22] SR - The semantic registry is published at https://www.hildeproject.com.