AN ONTOLOGY TO SUPPORT BRAZILIAN TRAFFIC FLOW MANAGEMENT BASED ON NASA's ATM REFERENCE MODEL

Luís Antonio de Almeida Rodriguez, José Maria Parente de Oliveira
Aeronautics Institute of Technology

rodriguezlaar@gmail.com, parente@ita.br

PAPER ID: SIT178

ABSTRACT

Time over the past two decades has shown different data models being introduced to try to standardize information for interoperability in the aviation domain. World's aviation organizations are rushing to maximize the efficiency of data exchange and actual initiatives like SWIM recommend the use of semantic knowledge descriptions to support air-traffic management information systems. The use of ontologies is increasing as a next step in aviation's data structures evolution, describing semantics (concepts, properties and relationships) and being conceived in machine-readable language, able to be accessed via programming languages. This paper presents an OWL-DL Ontology to support Air Traffic Flow Management based on NASA's ATM (Air Traffic Management) reference model. The original RDF files from NASA's release are the core of the implemented ontology, which was built to adapt NASA's taxonomy to Brazilian aeronautical laws and rules. Furthermore, the paper presents a set of experimental results which were carried out to manipulate the Brazilian customized ontology by using Python language and making CRUD (Create, Read, Update, and Delete) operations. In addition, the experiments show how to fill out and persist a flight plan form to simulate a pilot and an aeronautical information system interacting for a flight authorization.

Keywords: Ontology, NASA's atmontocore, machine-readable language, OWL.

ACKNOWLEDGMENTS

We would like to thank all the institutions involved in this work for their support: Aeronautics Institute of Technology - ITA, Air Navigation Institute (ICEA), Airspace Control Department (DECEA), and the *National Council for Scientific and Technological Development* – CNPq.

1 INTRODUCTION

The next generation of the worldwide air transportation system is an important subject of a lot of research by national airspace teams in several countries (SWIM, 2022). Civil aviation authorities all around the globe are rushing to integrate systems with the FAA (Federal Aviation Association) and Eurocontrol, two of the biggest ATM organizations. It is worthy saying that the use of reference models adopted by these two players is spreading among ATM software development teams.

The SWIM (System Wide Information Management) (SWIM, 2022) Program recommends the use of conceptual and logical models to be adopted by any system which must be integrated with this worldwide new reference model. One facet of this collaborative work among countries is to increase the information flow efficiency of all airlines operations to facilitate the whole civil air transportation network. Brazilian aviation authorities are also running to integrate their ATM systems to this new age (DECEA, 2022).

The lack of the adoption of a single formal data exchange model by world's aviation is holding back its ATM systems integration (ICAO, 2022). Different data models must be translated by software to share information and the Ontologies are the artifacts which can enable a shared understanding, using a formal way (OWL, 2022) to build semantic descriptions which can be used as machine-readable knowledge cores by ATM systems.

The lack of a formal data exchange model to describe the Brazilian Traffic Flow Management domain is a problem since it is an obstacle to integrate air-traffic operations systems with those two important ATM players. Thus, it is too expensive for Brazil and FAA or Eurocontrol to exchange data by software. The goal of this work is to present an implementation of OWL-DL (OWL, 2022) ontology based on NASA's ATM reference model (NASA, 2022) to serve as a formal data exchange model to Brazilian ATM systems.

This paper is organized as follows: Section 2 presents NASA's ATM ontology and a related paper which presents a direct comparison between NASA's and the Eurocontrol ontology. Section 3 presents the ontology development and Python experiments of querying and executing CRUD operations in the ontology, aiming to validate functional requirements and the purpose of the ontology. Section 4 presents contributions, a brief conclusion and further works.

2 NASA's ATM ONTOLOGY AND A RELATED WORK

NASA's ATM Ontology is a conceptual model which is implemented and released by a work group representing FAA, NASA and some Industry organizations using the machine-readable languages OWL, RDF and RDFS (W3C, 2022). It defines classes of entities and their respective relationships considering (NASA, 2022) the United States National Airspace System (NAS) and the management of the air-traffic domain.

A huge number and variety of entities which are directly related to that domain is represented, including classes corresponding to *flights, flight plans, aircraft, airports, weather conditions* and many others. The ontology has become useful since it describes a variety of information relevant to ATM operations in a generic mode, considering information exchange, data query and semantic search, integration and the information standardization.

NASA's motivation to develop this artifact was (NASA, 2022) the need to integrate heterogeneous forms of aviation data for aeronautical research applying semantic integration techniques. The ontology is released as six different files with the "*.owl*" extension: *ATM.owl*, *NAS.owl*, *general.owl*, *equipment.owl*, *data.owl* and *atmontocore.ow*l. The ontology serves as a common source of structured knowledge which can also be a formal vocabulary to allow data exchange using machine-readable language.

Data coming from multiple sources can be transformed into ATM Ontology instances (NASA, 2022) and can be loaded into a triple store (data organized as subject, predicate, and object) which can be queried and the results can serve a specific purpose. NASA's Ontology is formatted and implemented as a set of RDF (Resource Description Framework) files using the basic knowledge of OWL (Web Ontology language), RDF and RDFS (Resource Description Framework Schema)(W3C, 2022).

The set of descriptions is organized into eight major sections which describe thematically-related sets of classes. Each one of these is described in its own subsection: *Airspace Structures and Facilities; Navigation: Routes, Fixes, Arrival and Departure Procedures; Traffic Management Initiatives; Operations: Flight, Carrier and Aircraft; Airport and Surface Operations; Weather; Sequences, Sub-sequences, Sequenced Itens.*

The organization, specification and the ontology development were headed by a group of organizations including NASA and FAA, which gave the ontology the nickname "*beauty"*. The entire set of *.rdf* files can be downloaded at NASA's web page and it is open-source for academic purposes (NASA, 2022).

Gringinger et al. (Gringinger, 2020) have presented a comparative evaluation between NASA's ATM (ATMONTO) and the Eurocontrol ontology AIRM-O, derived from the ATM Information Reference Model (AIRM) (AIRM, 2022). The authors have established a comparative mapping between those artifacts in an effort to capture the concepts and semantic relations which could be common.

Such a work presents a deep evaluation made on both ontologies by a team of six human experts and some automated tools. These experts were of two types: the ones who were pilots and ontology experts and the others who were only ontology experts, providing this a fair condition to evaluate the artifacts. The experts have mapped the concept terms from ATMONTO to AIRM-O while indicating the degree of match using a very simple categorical scale. After evaluating the scale of the match among the experts, they produced another evaluation comparing both ontologies again using automated general-purpose ontology matching tools.

Both automated and manual results show there are a lot of differences in the terms presented by both ontologies, for example, the entity *runway* is described on the ATMONTO using only five different metadata and AIRM-O implements more than twenty entities to present the description of the same object.

Those authors have made a huge effort to provide a way to have some kind of harmonization between the United States and European aviation and they have concluded it is not so simple and it has to be done by several-country teams.

3 ONTOLOGY DEVELOPMENT AND PYTHON EXPERIMENTS

Ontology Development

The idea was to focus on the problem: - *the lack of a formal data exchange model to integrate Brazilian ATM information systems with FAA and Eurocontrol through SWIM.* So there is a need to develop a Brazilian Formal Air Traffic Flow Management Reference Model. To develop the ontology, the authors used the approach presented in (Ron & Smith, 2016) and as the technological solution to implement this model as an ontology, using the Web Ontology Language (OWL, 2022).

The purpose of deciding about the OWL language was to follow SWIM specifications (SWIM, 2022) which use this machine-readable language to represent the ATM domain. NASA's and Eurocontrol's artifacts also use this language (Gringinger, 2020). The first step of ontology development (Ron & Smith, 2016) was: "*to identify the primary tasks the ontology will be designed to be able to realize and the essence of its Domain."*

Table 1 - Competency Questions (CQ)

cq1	Is there an entity related to each of all the required fields of the Brazilian Abbreviated Flight Plan Form?
cq2	Can I create an instance for each of all those fields and fill & persist a complete FP Form as an. owl file with a specific name?
cq3	What is the Flight Plan of a specific *Pilot_In_Command* ?
cq4	Can I re-open the named and persisted ".owl" file and am I able to manipulate it?
cq5	Can I list the filled fields of the form?

In this case the authors have implemented a set of *Competency Questions* (CQ) (Potoniec, 2020) to define the Functional Requirements to implement the ontology. As in Table 1, CQ are artifacts destined to define the questions that must be answered by the ontology to validate it and it was built to guide the authors to customize NASA's taxonomies to adapt all the entities and business rules of Brazilian Aviation

laws contained in MCA 100-11 – (*Manual do Comando da Aeronáutica)* (MCA 100-11, 2022). The **Flight Plan Forms** were used as a use case into the Brazilian ATM domain and Table 1 presents the set of CQ and they are going to be validated using Python experiments.

After a set of functional requirements have been specified, the next step was (Ron & Smith, 2016) : *"to identify and evaluate existing ontologies with overlapping domains. Reuse as far as possible the ontology content which satisfies the defined requirements."*

The authors carried out a mapping between those described in the Brazilian Abbreviated Flight Plan Form (MCA 100-11, 2022) and the set of entities described by NASA's ontologies. The goal was to find possible matches between the two different sets of descriptions considering OWL Classes, Object Properties or any other stereotype offered by the language and used in NASA's files and make a complementary entities creation to customize the original ontologies to the Brazilian domain. NASA makes available a manual to describe all the entities of its ontology (NASA, 2022).

Three types of matching terms were considered (Gringinger, 2020):

- **Total Match** - when the Brazilian term of the flight plan form is exactly like a specific term contained in the original NASA's taxonomy. In this case no new entity is created and NASA's original OWL type is used to represent the form field.
- **Partial Match** - when the Brazilian term is similar but has some specific difference in its essence. In this case the authors created an inheritance relationship and a subclass of the original term in NASA's file is created with a specific name to match the Brazilian term.
- **Zero Match** - when there is no term in the original ontology which is at least similar to the Brazilian one. In this case a new term was created as a root Class or Property in the TFM_BR ontology.

Figure 1 presents some of the job of mapping Portuguese terms from MCA 100-11 form, their respective translations and NASA's closest match entity after the comparisons. It also presents an example of *Zero Match* with the "*Número"* cell, which is translated and defined into its essence, but there is no term in NASA's taxonomies which could define this one.

After the mapping job the next step was (Ron & Smith, 2016): "*to arrange the whole set of defined terms to form a backbone is-a hierarchy, in the sense each node at a level which is lower than the root level is connected by a single is-a link to its parent node. The ontology must guarantee a single inheritance."*

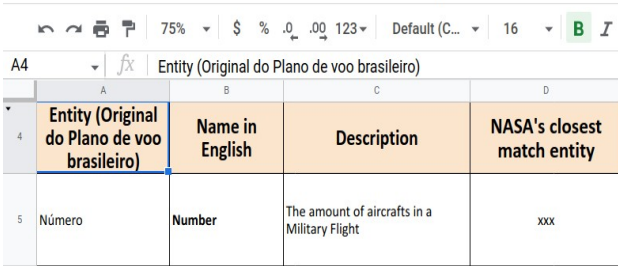

Figure 1 A cut of the terms mapping between Brazilian Flight Plan Form and NASA's terms

The authors took into consideration the results of the mapping task to define all the entities which would fit into NASA's taxonomy to provide the customization of entities and relationships among them to Brazilian aeronautical laws and rules. So, a new empty ontology was created using the software Protegé (Protegé, 2022) with the name **TFM_BR** (Traffic-Flow Management BRAZIL).

The idea about TFM_BR's core was to import NASA's *atmontocore* architecture as presented in Figure 2. The essence of the proposed ontology is to use NASA's definitions without any direct change on the original files. The OWL language has a native stereotype called "*import"* (OWL, 2022) which allows a single ontology to make use of the taxonomy of several other ones just by literally importing them in its code.

Importing the whole set of NASA's files made it possible to use the taxonomy of all its ontologies without making changes on any entity of the original implementation, at the same time it was easy to create new entities into the new ontology to fit the domain definitions to Brazilian features.

As shown in Figure 2, to implement NASA's files, we used the stereotype *Import* of OWL 2 to capture the whole set of NASA's ontologies taxonomy and axioms. Each entity described in red in Figure 2 represents one

ontology, which means one specific *.owl* file of NASA's last release (NASA, 2022).

To complement this import, a set of new entities based on the term mapping job was created to support the specific features of the Brazilian *Abbreviated Flight Plan Form* exactly as it is described by official authorities (MCA 100,11, 2022), as shown in Figure 3. The new entities are presented in Figure 4 in **bold** letters and having a "**_BR**" at the end of their ID, to identify the set of non-imported objects.

<**Import**>http://THE_project.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/ATM</**Import**>
<**Import**>http://THE_project.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/NAS</**Import**>
<**Import**>http://THE_project.comt.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/atmontoCore</**Import**>
<**Import**>http://THE_project.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/data</**Import**>
<**Import**>http://THE_project.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/equipment</**Import**>
<**Import**>http://THE_project.com/Ontologies/NASA_atmontocore/NASA_OwlFIles_owl_xml/general</**Import**>

Figure 2 Importing NASA's ontologies to TFM_BR

Each entity (field) of the form in Figure 3 was implemented in the TFM_BR ontology exactly as its meaning and considering the matching level it was classified and Brazilian business rules. The final position of all terms in NASA's taxonomy was defined considering the result of the terms mapping.

Anexo B - Formulário de Plano de Voo Simplificado

FRENTE

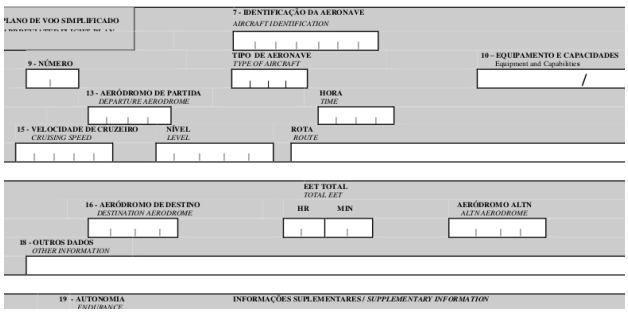

Figure 3 A cut of the Abbreviated Flight Plan Form (MCA 100 - 11, 2022)

Figure 4 presents Protegé and the set of **bold** classes which were created as the result of the positioning of the new terms into their specific hierarchies and meanings at the original taxonomy of NASA's ontology. The non-bold entities are from the original implementation. The Brazilian ontology includes also a set of instances (owl Individuals)(OWL, 2022) pertaining to the ANAC_CODE Class, which define a unique code for each pilot or any other crew member.

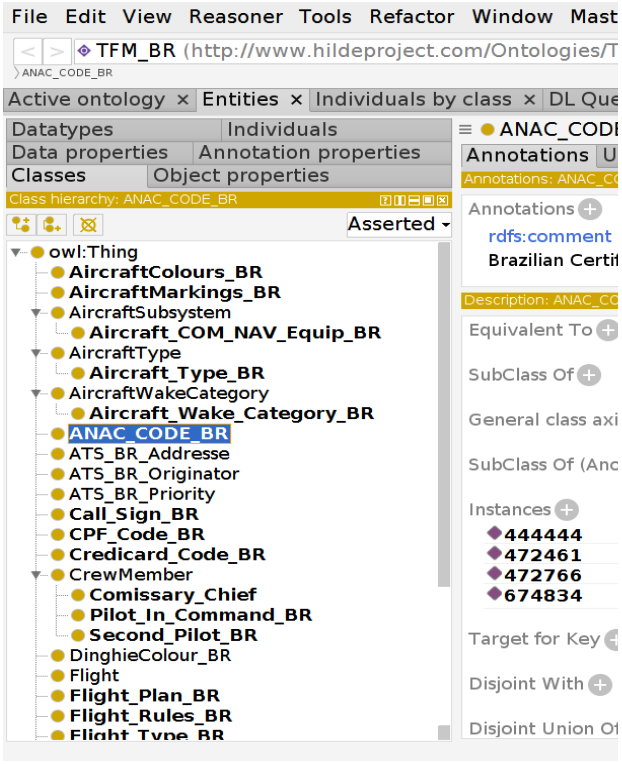

Figure 4 A cut of Protegé screen and the new classes of TFM_BR in bold

The following step was the creation of another set of instances which would serve as parameters to be used by the authors to make CRUD operations *(Create, Read, Update and Delete)* over the TFM_BR ontology during the tests, as presented in the right side of Figure 4.

The new ontology can be seen as the semantic description which serves as an core to carry knowledge which allows a developer to encapsulate it with a software and interact with common airman users to satisfy their real life goals making use of this artifact as a service.

The ontology implementation was completed and the authors were ready to start the Python experiments destined for ontology validation (Potoniec, 2020).

- *Python Experiments*

The idea to present Python experiments is focused on reaching the ability to manipulate ontologies via Python executing CRUD operations and use it to build software based on this kind of semantic representation. The difference from other software is that all CRUD

operations are made by using semantic objects described by the stereotypes of OWL language, not *strings*, *enumerations*, *integer* or any type defined by programming languages.

The goals of Python experiments were:

- Program queries to be executed in the TFM_BR ontology using the owlready2 Python library (OWLREADY 2, 2022) to make searches whose results could answer and validate the CQ in Table 1.
- To create a clone of the ontology TFM_BR using Python and to fill out all the components of the Abbreviated Fight Plan Form (Figure 3) with specific data to simulate a pilot asking for a Flight Plan authorization and filling a Web form.

After the simulation is finished, the ontology file which describes that specific filled Flight Plan Form must be persisted into a *file system* with the *.owl* extension and must be re-opened at any further moment for legal purposes.

All the CQ were considered as the basis to program Python queries to provide the necessary answers. The ability to import other ontologies taxonomies offered by the library owlready2 is defined as (OWLREADY2, 2022): *from owlready2 import* * and it will be omitted on the code descriptions. The Python code solutions for the CQ (Table 1) are:

- **cq1** - asks for the list of all Brazilian entities mapped and inserted into NASA's taxonomy. Table 2 presents a Python script which extracts and prints the list of new implemented Classes, Object Properties and Individuals from the TFM_BR implemented ontology:

Table 2 - Python script for cq1

```
onto = get_ontology("http://www.deproject.com/Ontologies/TFM_BR_and_FlightPlan_Models/TFM_BR.owl")#download from an http address on the internet
onto.load()#puts ontology in RAM memory
onto.classes()#list Classes
onto.object_properties()#list Object Properties
onto.individuals()#list Instances (Individuals)
# or (another way to do it)
print(onto.search(iri = "*_BR"))#search for the new entities
```

The shell result after running this script is presented in Table 3. It is possible to see the reference "*_BR*" at the end of all of the non-imported entities.

Table 3 - Python shell result for cq1

[TFM_BR.AH_2, TFM_BR.AS_350, TFM_BR.AT_27, TFM_BR.Boeing_737_800, TFM_BR.Boeing_787_800, TFM_BR.C_95, TFM_BR.C_99, TFM_BR.F_5, TFM_BR.F_5M, TFM_BR.TFM_BR.WakeCategory_BR_M]
[TFM_BR.AircraftColours_BR, TFM_BR.AircraftMarkings_BR, TFM_BR.Aircraft_COM_NAV_Equip_BR, TFM_BR.Aircraft_Type_BR, TFM_BR.Aircraft_Wake_Category_BR, TFM_BR.Brazilian_Airport_BR, TFM_BR.CPF_Code_BR, TFM_BR.Call_Sign_BR, TFM_BR.Credicard_Code_BR, TFM_BR.DinghieColour_BR, TFM_BR.Flight_Plan_BR, TFM_BR.Flight_Rules_BR, TFM_BR.Flight_Type_BR, TFM_BR.Pilot_In_Command_BR, TFM_BR.Second_Pilot_BR, TFM_BR.Waypoint_BR, TFM_BR.anac_code_CPF_BR, TFM_BR.filledBy_BR, TFM_BR.hasACFTColours_BR, TFM_BR.hasACFTMarkings_BR, TFM_BR.hasAircraftType_BR, TFM_BR.hasAnacCode_BR, TFM_BR.hasCOM_NAV_Equip_BR, TFM_BR.hasCpf_BR, TFM_BR.hasDinghieCoverColours_BR, TFM_BR.hasFillingDay_BR, TFM_BR.hasFlightDay_BR, TFM_BR.hasFlightType_BR, TFM_BR.hasFlight_Rules_BR, TFM_BR.hasNASDay_BR, TFM_BR.hasPilotInCommand_BR, TFM_BR.hasWaypoint_BR, TFM_BR.has_ATS_BR_Addresse_BR, TFM_BR.has_ATS_BR_Originator_BR, TFM_BR.has_ATS_BR_Priority_BR, TFM_BR.has_Call_Sign_BR, TFM_BR.isA_2nd_Alternative_Aerodrome_BR, TFM_BR.isA_GPS_Fix_BR, TFM_BR.isA_NDB_Fix_BR, TFM_BR.isA_TACAN_Fix_BR, TFM_BR.isA_VOR_Fix_BR, TFM_BR.isAn_Airport_BR, TFM_BR.isAn_Alternative_Aerodrome_BR, TFM_BR.aircraft_Fuel_Flow_BR, TFM_BR.airlineSerialNumber_BR, TFM_BR.cruisingAltitude_BR, TFM_BR.cruising_speed_BR, TFM_BR.emergencyRadioCom_BR, TFM_BR.emergencyRadioCom_UHF_243_0_BR, TFM_BR.emergencyRadioCom_VHF_121_5_BR, TFM_BR.personName_BR, TFM_BR.personPassword_BR, TFM_BR.personSignature_BR, TFM_BR.totalEstimatedEnrouteTime_BR]

Table 3 presents the short *IRI* (OWL, 2022) of the results of the actions: to *import* owlready 2 library, to *download* the TFM_BR ontology using a HTTP address as parameter, to put TFM_BR in *RAM memory* and then it will

be able to be manipulated by operating any CRUD operation. After that, the code presents queries about the whole set of owl entities which were created as the result of the term mapping job and the insertion into NASA's taxonomy. It is possible to observe that there is a huge set of different owl types like *Classes*, *Object Properties* and *Individuals* presented in Table 3.

- **cq2** asks to fill out the Flight Plan Form items and to persist it. For a matter of available space, the authors reduced the number of items to simulate the form. The selected items are the following: *Flight Plan*, *Pilot-in-command*, *Call Sign*, *Departure Airport*, *Arrival Airport*, *Alternative Airport*, *Planned Route*, *ANAC_CODE*, *Aircraft Wake Category*, *People on Board* and *Total Endurance*. Python code is presented in Table 4 and Table 5:

Table 4 - Python code for cq2

```
onto = get_ontology("http://www.deproject.com/Ontologies/TFM_BR_and_FlightPlan_Models/TFM_BR.owl")#download from http
onto.load()# put the file in RAM memory
onto.save("/home/rodriguez/Desktop/SITRAER_2022.owl")#persist in filesystem with this name
onto2 = get_ontology("///home/Desktop/Only_Dev_CodigoFonte/SITRAER_2022_PlanoDeVoo_Ontologia_15_Agosto_2022.owl")#download from filesystem
onto2.load()#puts filesystem's ontology in RAM memory
```

Table 4 presents the download of the TFM_BR from a HTTP address and the allocation of this model in *RAM memory*, the persistence of an instance (an instance is the same file "saved as" another name) of TFM_BR with a specific name. After that the original ontology from the internet is destroyed because it serves as a generic model and should not be modified. A specific instance of the internet ontology must be saved as another "*.owl*" file and with another name to identify the file in a history backup. The same process to download and put the new file in *RAM memory* is presented with the *onto2* variable getting the file from the *Desktop* to be able to manipulate the correct persisted file.

The *onto2* variable in Table 4 now is able to allow CRUD operations and the next step was to create the instances and relationships among them to simulate a pilot filling those items from Figure 3 to get a flight plan authorization. Table 5 presents the python code to define the values of those fields being filled by a pilot and their relationships to the related Flight Plan.

Table 5 - Python code for cq2

```
filesystem = get_ontology("/home/rodriguez/Desktop/Only_Dev_OWL_TEMP_Files/Teste_Ontologia_SITRAER_20_Ago.owl").load()
fp = filesystem.Flight_Plan_BR("F_P_SITRAER")
codigo_anac = filesystem.ANAC_CODE_BR("888888")
piloto = filesystem.Pilot_In_Command_BR("Bruce Dickinson")
codigo_chamada = filesystem.Call_Sign_BR("Flight_MAIDEN")
depAirp = filesystem.search(iri = "*SBGL")
arrvAirp = filesystem.search(iri = "*SBNT")
alterAirp = filesystem.search(iri = "*SBRF")
route = filesystem.search(iri = "*SJC_SITRAER")
aircraft = filesystem.search(iri = "*787")
radio = filesystem.search(iri = "radio*")
cod_ANAC = filesystem.search("*472461")
wake_cat = filesystem.search("*BR_J")
endurance = 04.50
pob = 130
fp.hasPilotInCommand_BR = [piloto]
fp.hasDepartureAirport_BR = [depAirp]
fp.hasArrivalAirport_BR = [depAirp]
fp.hasAlternateAirport_BR = [depAirp]
fp.hasCall_Sign_BR = [Flight_SITRAER]
fp.hasPlannedRoute_BR = [route]
fp.hasCOM_NAV_Equip_BR =[radio]
piloto.hasANAC_CODE_BR = [cod_ANAC]
aircraft.hasWakeCategory_BR =[wake_cat]
fp.hasPeopleOnBoard_BR =[pob]
fp.hasTotalFlightEndurance_BR = [endurance]
filesystem.save("/home/rodriguez/Desktop/Only_Dev_OWL_TEMP_Files/Teste_Ontologia_SITRAER_20_Ago.owl")#re-writing on file system
```

Table 5 presents the creation of a set of instances and relationships among them which represent all the fields of the Abbreviated Flight Plan Form (Figure 3). The process to create data and metadata is similar to what a regular information system is programmed to do. When a user fills form fields on a Web page, the system captures text on a screen and persists on a database as a primitive data type OWL, 2022). The authors have transformed text into semantic knowledge, able to be identified as a precisely defined object, with a specific meaning and ID and inserted into a previously defined taxonomy, becoming a semantic entity existing

on a semantically defined domain. Table 5 presents the download of the ontology that was in the file system; it is put in RAM memory and manipulated to create Individuals and triples among them (OWL, 2022).

The legend:

- Brown - Python variables to support the owlready2 methods.
- Black - owlready2 methods.
- Purple - OWL Individuals
- Orange - OWL Classes
- Blue – OWL Object Properties
- Green – OWL Datatype Properties values

It is possible to observe that the code in Table 5 associates all different metadata to the specific instance of Flight_Plan_BR. It is a *1 to N* relationship to refer all the fields on the Flight Plan Form (Figure 3) to a unique reference of a Flight Plan, the *Individual* F_P_SITRAER. The code simulates the whole process: to fill the form, to process and to persist the owl file in the file system to make a documents history. It is possible to observe that the owlready2 library mixes OWL elements with Python variables to execute CRUD operations over the ontology.

Triples are represented in Table 5 by a Brown - Blue - Brown line. These lines associate two python variables and this code creates a triple between two OWL instances, or, two OWL *Individuals* to establish a meaningful relationship, which describes business rules and laws about Brazilian ATM features. This way it is possible to imagine a simulation of a Flight Plan authorization being requested, processed and persisted as an ontology in a ".owl" extension, able to be processed, updated or opened by Protegé (Protegé, 2022). At the last line it is possible to observe the ontology being persisted with the same name to re-write the file.

- **cq3** - asks for the name of the that specific Flight Plan associated to a *Pilot in Command* filled by the user. Table 6 presents a simple query to find what is the instance of the class *Flight_Plan_BR* which is associated with Bruce_Dickinson?

It is possible to observe the python shell result presenting that specific instance of Flight_Plan_BR as the answer for the python query *search()*.

Table 6 - Python code for cq3 - VALIDATED

```
cq3 = filesystem.search(onto2.hasPilotInCommand_BR =
"*Bruce_Dickinson")
print(cq3)
#Shell result:
[TFM_BR.F_P_SITRAER]
```

- **cq4** - asks to re-open the persisted .owl file to show interoperability of the ontology created for the Flight Plan process (Table 5) with all possible future queries.

Table 7 presents python code to re-open the persisted file and a new kind of query, by *iri*, presenting a new way to search for entities.

Table 7 - Python code for cq4 - VALIDATED

```
re_open =
get_ontology("/home/rodriguez/Desktop/Only_Dev_OWL_TE
MP_Files/Teste_Ontologia_SITRAER_20_Ago.owl")
re_open.load()
print(re-open.search(iri = "*SITRAER"))
#Shell result:
[TFM_BR."F_P_SITRAER"]
```

At this point the authors checked that all ontologies generated by the experiments are compatible with any kind of update using owlready2 or Protegé.

- **cq5** - asks to list all the Flight Plan Form fields that were filled to compose the specific instance "F_P_SITRAER".

Python code in Table 8 presents one query for each filled field of the Flight Plan Form and the shell results with the short iri for each one of them:

Table 8 - Python code for cq5 - VALIDATED

```
print(re-open.search(iri = "*SITRAER"))
print(re-open.search(iri = "*Dickinson"))
print(re-open.search(iri = "*_MAIDEN"))
print(re-open.search(iri = "*_787"))
print(re-open.search(iri = "*SJC_SITRAER"))
print(re-open.search(iri = "*GL"))
print(re-open.search(iri = "*NT"))
print(re-open.search(iri = "*RF"))
#Shell result:
[TFM_BR.F_P_SITRAERTFM_BR.Bruce_DickinsonTFM_
BR.Boeing_787TFM_BR.Flight_MAIDENTFM_BR.RouteSJ
C_SITRAERTFM_BR.SBGLTFM_BR.SBNTTFM_BR.SBRF]
```

It is possible to observe the search for each entity by *iri*, which means python searches for the real name of the Individual and it is not a ***string***, it is a semantic description programmable and able to be queried like this. This way all CQ were answered by the new ontology and the functional requirements were validated, the ontology does what it was supposed to be done.

4. CONTRIBUTIONS, CONCLUSIONS AND FURTHER WORKS

Until this point the authors developed a set of semantic descriptions which brings some contributions for Air Transportation and for Software Developers Communities:

- A new extension of NASA's ontology, which imports the complete core of original files and creates a new one, the TFM_BR, which describes Brazilian TFM domain fitting new entities into those taxonomies;

- A new Brazilian TFM formal vocabulary with terms and relationships defined by the terms mapping between NASA's ones and those presented in the Flight Plans Forms. It can be used by air transportation systems as a dialect;

- A new TFM domain ontology in which all the Brazilian terms were fitted into NASA's taxonomy, according to each term's meaning and matching. It is a way to standardize the semantic vocabulary of all air transportation information systems;
- A new formal data exchange model which allows Brazilian ATM software to share information. It can maximize information exchange among airlines systems;
- The ability to use the Python language to develop software which uses ontologies as the core of information. It improves the quality of ATM and TFM developed systems.

These contributions are useful to all players of air transportation, considering airlines, crew members and air-traffic authorities. All information being filled on a Web form MUST be correctly interpreted by humans and software to reduce the risk of accidents caused by information misunderstoods. Air-traffic information must be interpreted by explicit semantic meaning because it allows developers to manipulate semantic objects, not *strings* or *enumerations*, like it is normally done using common programming languages. The authors have shown here python only interacts with the ontology by manipulating OWL Language stereotypes during the whole processing, or, semantic objects.

It is possible to conclude that the fidelity of the meaning of the processed data is very important at this new era of air transportation, which runs for getting more precision, sharing ability and a less cost to exchange data. World's aviation needs synchronization to avoid waste of time, of money and of people's lives and it is the reason for the running for semantic descriptions like the ontology developed in this work, which allows information systems to use them as the core of the domain description and get the semantic level of computing throughput.

References

OWL. Web Ontology Language. W3C recommendation.https://www.w3.org/OWL/. Access 06/1st/2022.

NASA. The NASA Air Traffic Management Ontology.Technical Documentation.Richard M. Keller.Ames Research Center, Moffett Field, California. June 2017.

W3C. World Wide Web Consortium. http://www.w3.org. Access 06/1st/2022.

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

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.

Gringinger, E. et al. A Comparative Study of Two Complex Ontologies in Air Traffic Management.Research, Frequentis AG, Vienna, Austria. Intelligent Systems Division, NASA Ames Research Center, Moffet Field, CA, USA. 2020.

AIRM. The ATM Information Reference Model (AIRM). The reference vocabulary for defining air traffic management information

https://airm.aero/. Access 08/23/2022.

Ron R. & Smith, B. et al. Best Practices of Ontology Development. White Paper. CUBRC, Inc., University at Buffalo, New York College of Technology and Institute for Military Support to Governance. October, 2016.

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.

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

https://www.w3.org/Submission/OWL-S/. Access 06/01/2022.

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.

DECEA. Departamento de Controle e Espaço Aéreo. HTTP://www.decea.gov.br. Access 06/01/2022

OWLREADY2. Python library owlready2-0.37. https://owlready2.readthedocs.io/en/v0.37/. Access 06/01/2022.