# GOOP-Hub User Guide

V1.0

October, 2021

Cássio C. Reginato, Jordana S. Salamon, Gabriel G. Nogueira, Monalessa P. Barcellos, Vítor E. S. Souza, Maxwell E. Monteiro, Renata S. S. Guizzardi

*Ontology & Conceptual Modeling Research Group (NEMO)*

*Computer Science Department*

*Federal University of Espirito Santo*

*Vitória – ES – Brazil*

*cassioreginato86@gmail.com*
*jssalamon@inf.ufes.br*
*gabriel.g.nogueira@aluno.ufes.br*
*monalessa@inf.ufes.br*
*vitor.souza@ufes.br*
*maxmonte@ifes.edu.br*
*renataguizzardi@gmail.com*

**SUMÁRIO**

1. Introduction .......... 3
2. GOOP-Hub Implementation .......... 4
3. GOOP-Hub Features .......... 5
3.1. Creating GOOPs .......... 6
3.2. Searching GOOPs for reuse .......... 10
3.3. Filtering candidate GOOPs for reuse .......... 13

# GOOP-Hub User Guide

## 1. Introduction

This document presents the main functionalities of GOOP-Hub and how to use it. GOOP-Hub has been developed in the context of a research project that investigates the use of GORE (Goal-Oriented Requirements Engineering) to improve the understanding of ontology scope and design rationale and contribute to ontology reuse.

GOOP-Hub is a component of GO-FOR, a Goal-Oriented Framework for Ontology Reuse. In GO-FOR, ontology models are depicted in fragments (i.e., domain ontology design patterns) related to goals. These model fragments are self-contained ontology structures called Goal-Oriented Ontology Patterns (GOOP), a new type of pattern to be applied to develop ontologies in a goal-oriented approach. In GO-FOR, goals can be used as parameters to support ontology shareability and reuse. In addition to GOOPs, GO-FOR introduces GOOPR (GOOP Repository), a repository to store GOOPs and that serves as an abstraction layer for ontology development. GO-FOR aids in the creation, storage, and reuse of GOOPs, promoting ontology development with and for reuse.

GO-FOR comprises: (i) a conceptual architecture that contains the necessary components to GO-FOR use; (ii) a goal-oriented process that guides ontology development considering GO-FOR architecture; and (iii) GOOP-Hub, a tool that supports GO-FOR use. Figure 1 shows an overview of GO-FOR architecture.

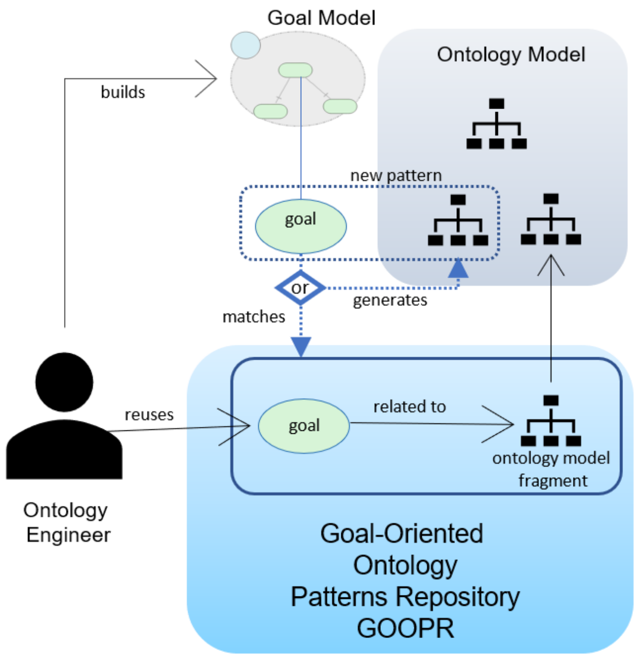


Figure 1. Overview of GO-FOR architecture

In a nutshell, in order to reuse GOOPs for ontology development, the ontology engineer must start by identifying the actors in the domain of interest and developing the goal models that describe the scope of the ontology to be developed (as suggested in Fernandes et al. (2011)[1]). In this way, the ontology purpose is made clear and the design rationale is expressed by means of the goals that guide the definition of what is to be addressed by the ontology and why. For each goal represented in the goal model, the ontology engineer verifies if there is a GOOP in the GOOPR related to it (i.e., if there is a GOOP containing that goal). If this is the case, the ontology engineer can reuse the GOOP by integrating it to the ontology model. In this case, we have *development with reuse*. Otherwise, the ontology engineer can create a new ontology model fragment to achieve the goal. Thus, it can relate the fragment to the goal (resulting in a GOOP) and store it in the repository for future reuse. In this case, we have *development for reuse*. Further information about GO-FOR can be found in (Reginato et al., 2019)[2].

Aiming to provide computational support to GO-FOR and promote its use, we have developed the GOOP-Hub, which enables the creation of and searching for GOOPs. GOOP-Hub consists of an interface that allows ontology engineers to record and retrieve GOOPs and a repository that stores GOOPs (i.e., a GOOPR).

Next, in Section 2, we present information about GOOP-Hub implementation and, in Section 3, GOOP-Hub main functionalities, some of its screens and information about how to use it.

## 2. GOOP-Hub Implementation

GOOP-Hub was implemented by using Web Ontology Language (OWL). The main advantage of this choice is that it is possible to make SPARQL queries using the GOOP-Hub metamodel structure as part of the query. Moreover, since the metamodel uses OWL, a widespread language used in the Semantic Web, it can be referred even outside the GOOP-Hub infrastructure. Figure 2 presents the core concepts of the GOOP-Hub metamodel.

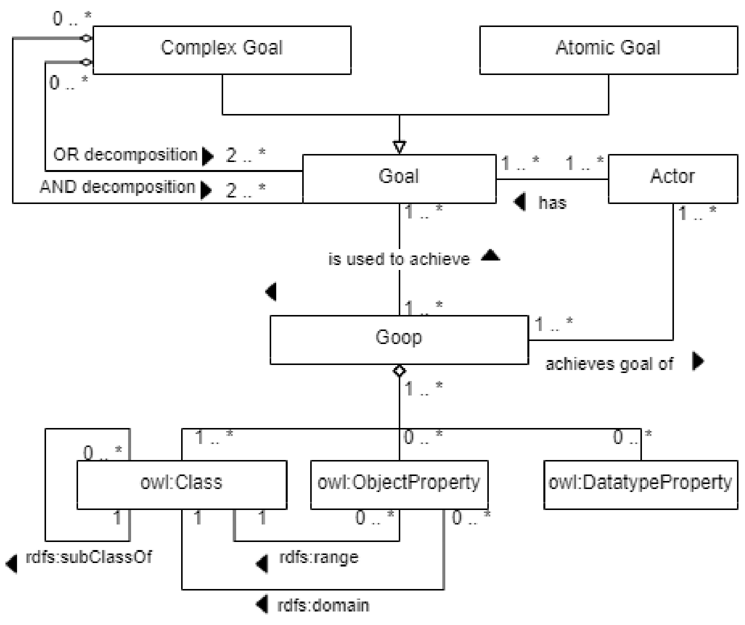


Figure 2. The GOOP-Hub metamodel

[1] .C.B. Fernandes, R.S.S. Guizzardi, G. Guizzardi, Using Goal Modeling to Capture Competency Questions in Ontology-based Systems, *J. Inf. Data Manag. 2* (2011) pp. 527–540.

[2] C.C. Reginato, J.S. Salamon, G.G. Nogueira, M.P. Barcellos, V.E.S. Souza, M.E. Monteiro, GO-FOR : A Goal-Oriented Framework for Ontology Reuse, in: *IEEE 20th Int. Conf. Inf. Reuse Integr. Data Sci.*, Los Angeles, CA, USA, 2019, pp. 99-106, doi: 10.1109/IRI.2019.00028.

The GOOP-Hub metamodel comprises elements of the OWL metamodel alongside goal modeling constructs. A GOOP is composed of OWL classes, object properties and data properties. These three OWL constructs are the main elements used to represent a model fragment in terms of its structure. The hierarchies must be represented by associating classes with its subclasses and domain and range of Object Properties can be used to describe to which classes an object property is associated. Since GOOPs are thought to be generic structures to be reused, we have just used the most basic OWL constructs of OWL LITE. For this reason, OWL DL constructs often used for reasoning purposes are ignored (e.g., equivalence and logical operators). Besides the OWL structure, a GOOP also involves a Goal that can be Atomic or Complex. A Complex Goal is composed of other Goals either by AND or OR decomposition. An AND decomposition is used when all its subgoals must be satisfied for its achievement. An OR decomposition occurs when only one needs to be satisfied. Goals are related to the Actors who want to achieve them. Actors are also related to GOOPs, indicating which model fragment is necessary for an actor to achieve a certain goal.

To add a GOOP into the repository the user informs the goal(s) and actor(s) to which the GOOP relates and uploads the ontology fragment in OWL format, alongside with a picture depicting the conceptual model of the ontology fragment and other information to describe the GOOP. The application performs the conversion to the meta-model, so that the relationships between GOOPs are established according to the relationships between goals. To store the GOOPs, we use Stardog[3], a knowledge database that stores data in triple format. We chose Stardog because its Java API is documented with functional examples and also because it offers tools for data comparison using AI (Similarity Search), as well as textual search based on Apache Lucene[4].

The goal-based search for GOOPs in the GOOP-Hub is made by using textual inputs referring to the ontology goals (e.g., *Specify event interval* or *Describe offering item*). In our implementation, we apply Apache Lucene, which is a java full-text search engine in which, given a search query, the API returns a set of documents (in our case, goals) sorted using a score system such that the documents (goals) most similar to the query are displayed first.

In addition to the textual searching, ontology engineers count with a SPARQL Endpoint, where they can perform complex queries involving all the elements of the metamodel and their instances. For example, the ontology engineer can search for a GOOP with the goal *Describe Location* filtering those that have the concept *City*. As result of the search, the ontology engineer receives a list of GOOPs that meet the parameters. The GOOPs can be visualized (by means of the pictures that depict the conceptual model) and downloaded in OWL format.

## 3. GOOP-Hub Features

GOOP-Hub has three main functionalities: create GOOPs, search GOOPs for reuse, and filter candidate GOOPs for reuse. Figure 3 shows the GOOP-Hub initial page.

[3] https://www.stardog.com/
[5] http://lucene.apache.org/

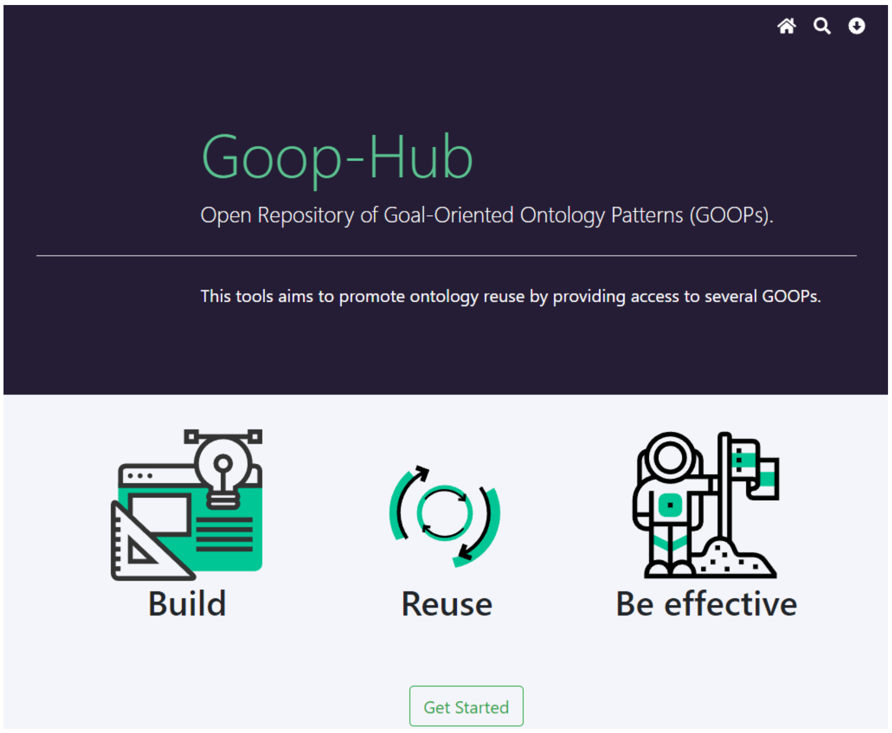

Goop-Hub
Open Repository of Goal-Oriented Ontology Patterns (GOOPs).
This tools aims to promote ontology reuse by providing access to several GOOPs.
Build
Reuse
Be effective
Get Started

Figure 3. GOOP-Hub initial page

## 3.1. Creating GOOPs

For adding a new GOOP to the repository, the user must click on the arrow button in the right upper corner of the initial page, as shown in Figure 4.

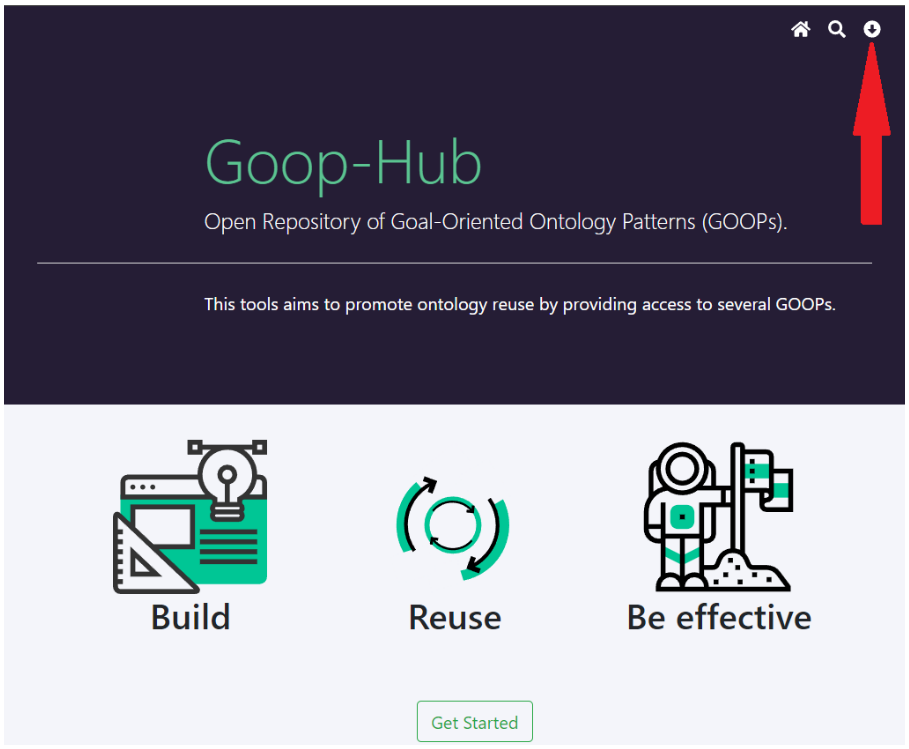
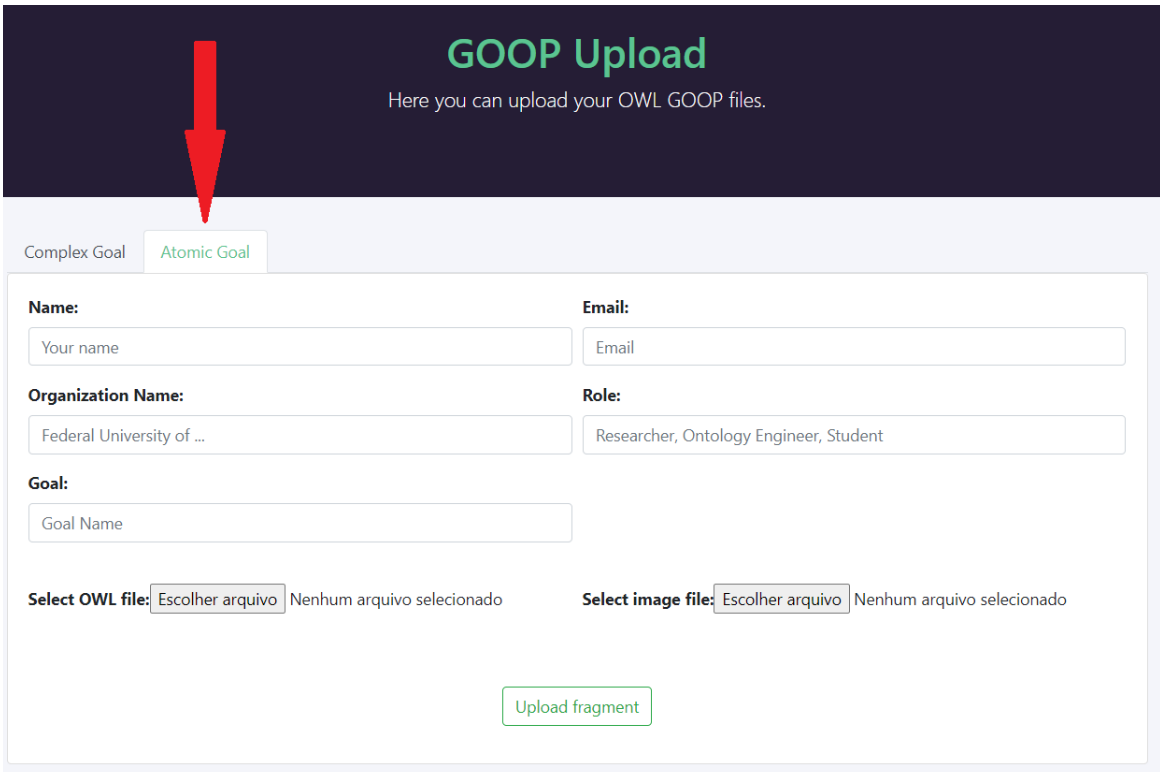

Goop-Hub
Open Repository of Goal-Oriented Ontology Patterns (GOOPs).
This tools aims to promote ontology reuse by providing access to several GOOPs.
Build
Reuse
Be effective
Get Started

Figure 4. How to access the page for creating GOOPs

The user can create GOOPs related to atomic or complex goals. An atomic goal is not decomposed in others. Complex goals, in turn, are goals composed of other goals (atomic or complex) either by AND or OR decomposition. An AND decomposition is used when all the subgoals must be satisfied for the goal to be achieved. An OR decomposition occurs when the goal is achieved if at least only one its subgoals is satisfied.

To create a GOOP related to an atomic goal, the user must click the option "Atomic Goal", as shown in Figure 5.


GOOP Upload

Here you can upload your OWL GOOP files.

Complex Goal | Atomic Goal

Name: Your name

Email: Email

Organization Name: Federal University of ...

Role: Researcher, Ontology Engineer, Student

Goal: Goal Name

Select OWL file: Escolher arquivo Nenhum arquivo selecionado

Select image file: Escolher arquivo Nenhum arquivo selecionado

Upload fragment

Figure 5. Selecting Atomic Goal option to create a new GOOP

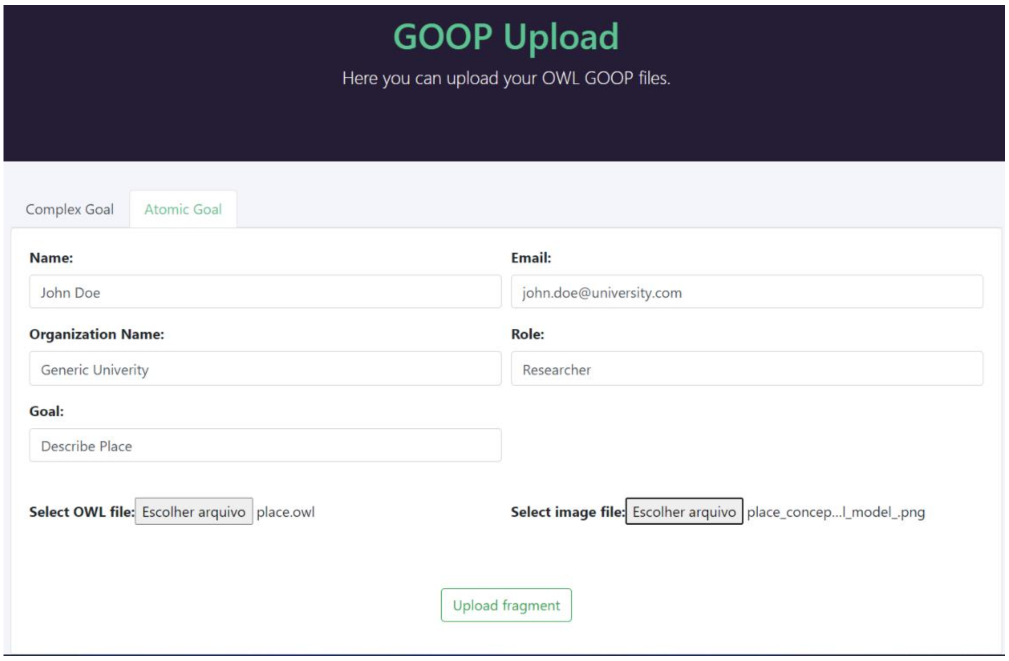
In the page that is shown to the user, he/she must fill out the fields with the required information about who is recording the GOOP (name, organization, contact, role), the goal that the ontology fragment achieves, the ontology fragment conceptual model and its OWL code. The two last are informed through the upload of two files: one containing the GOOP conceptual model (as an image file, at the "Select image file" field) and another containing its OWL code (in the "Select OWL code" field). Concerning the GOOP name, it must be considered the following rule: the GOOP name will be name of the goal to which the GOOP is related and must be composed of a verb (bare infinitive) and a noun (or a noun phrase), e.g., "define course program", "describe product offering".

Regarding the verb, if the goal associated to the GOOP concerns classification, the user must use "classify". Classification applies when the goal is to classify an entity (e.g., a specimen) according to a system of classification that is disposed as taxonomies. If the goal concerns identification, the verbs to be used are "identify" or "determine". Identification occurs when the purpose is to identify a well-known element (e.g., a chemical element of a periodic table) according to a taxonomy or a list of values (enumeration). Lastly, if the goal associated to the GOOP concerns description, the following verbs can be used: "describe", "characterize", "define", "specify" or "register". Description occurs when an ontology fragment is meant to describe an entity (i.e., an event or an object) by representing its relationship with other entities that represent its characteristics. For example, a birth can be described by its date, the mother and the baby that participated in that event. Table 1 summarizes the standard terminology for verbs when naming a GOOP.

Table 1. Terminology for naming GOOPs

| **Type of action** | **Definition** | **Model Structure** | **Verbs to be used** | **Example** |
|---|---|---|---|---|
| Classification | Classify an entity according to a hierarchy | Taxonomy | Classify | Classify a specimen |
| Identification | Identify an entity as a well-known entity | Taxonomy or enumeration | Identify, determine | Identify blood type |
| Description | Describe an entity according to its characteristics | Complex structure | Describe, characterize, define, specify, register | Describe crime |

Regarding the noun, it refers to the entity to be classified, identified, or characterized. In the examples shown in Table 1, the nouns are specimen, blood type and crime. The main guideline is to be clear about the entity to enable the identification of synonyms, generalizations, and specializations. For instance, in the goal "describe a water sample", "water sample" is the noun. "Water sample" has "hydro sample" as synonym, is generalized by "sample" and specialized by "freshwater sample". Therefore, it is possible to search GOOPs considering different keywords for nouns. Figure 6 shows, as an example, a page containing data to record the GOOP "Describe Place".


## GOOP Upload
Here you can upload your OWL GOOP files.

**Name:** John Doe
**Email:** john.doe@university.com
**Organization Name:** Generic Univerity
**Role:** Researcher
**Goal:** Describe Place
**Select OWL file:** Escolher arquivo place.owl
**Select image file:** Escolher arquivo place_concep...l_model_.png

Figure 6. Recording the GOOP "Describe Place", related to an atomic goal

Figure 7 shows, as an example, the image depicting the GOOP conceptual model uploaded when creating the GOOP “Describe Place”.

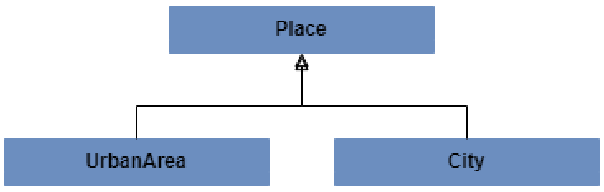
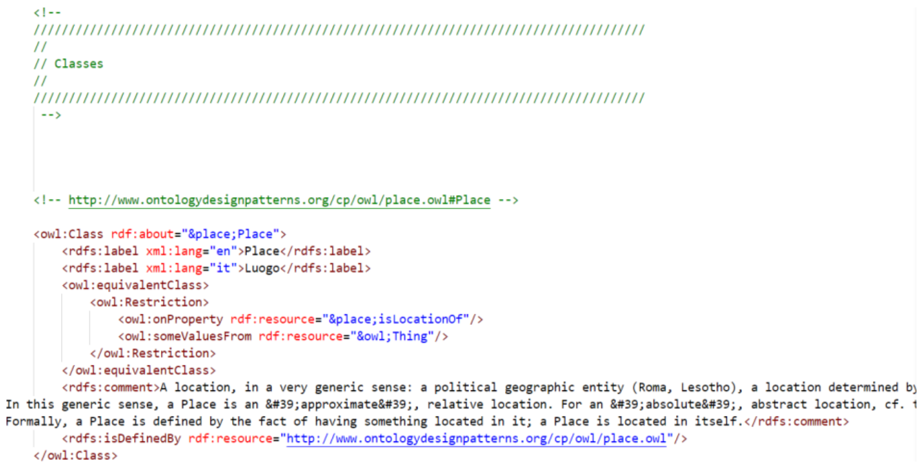


Figure 7. Uploaded GOOP conceptual model

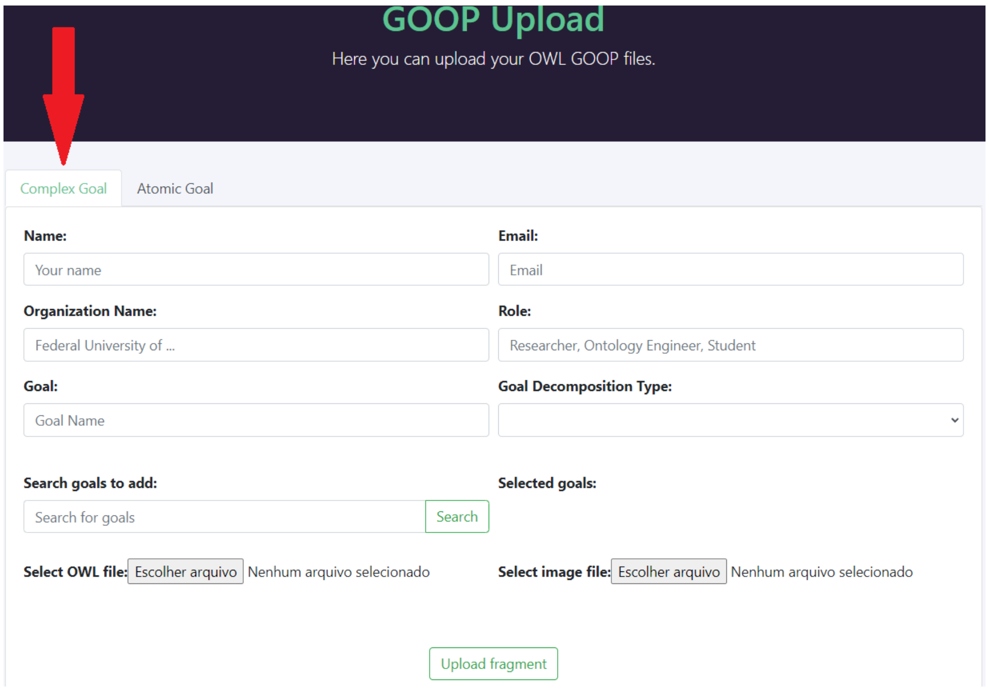
Figure 8 shows, as an example, a fragment of the OWL code that was uploaded when creating the GOOP “Describe Place”.



```
    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.ontologydesignpatterns.org/cp/owl/place.owl#Place -->

    <owl:Class rdf:about="&place;Place">
        <rdfs:label xml:lang="en">Place</rdfs:label>
        <rdfs:label xml:lang="it">Luogo</rdfs:label>
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&place;isLocationOf"/>
                <owl:someValuesFrom rdf:resource="&owl;Thing"/>
            </owl:Restriction>
        </owl:equivalentClass>
        <rdfs:comment>A location, in a very generic sense: a political geographic entity (Roma, Lesotho), a location determined by
In this generic sense, a Place is an &#39;approximate&#39;, relative location. For an &#39;absolute&#39;, abstract location, cf. t
Formally, a Place is defined by the fact of having something located in it; a Place is located in itself.</rdfs:comment>
        <rdfs:isDefinedBy rdf:resource="http://www.ontologydesignpatterns.org/cp/owl/place.owl"/>
    </owl:Class>
```

Figure 8. Uploaded OWL code

To create a GOOP related to a complex goal, the user must click the option “Complex Goal”, as shown in Figure 9.



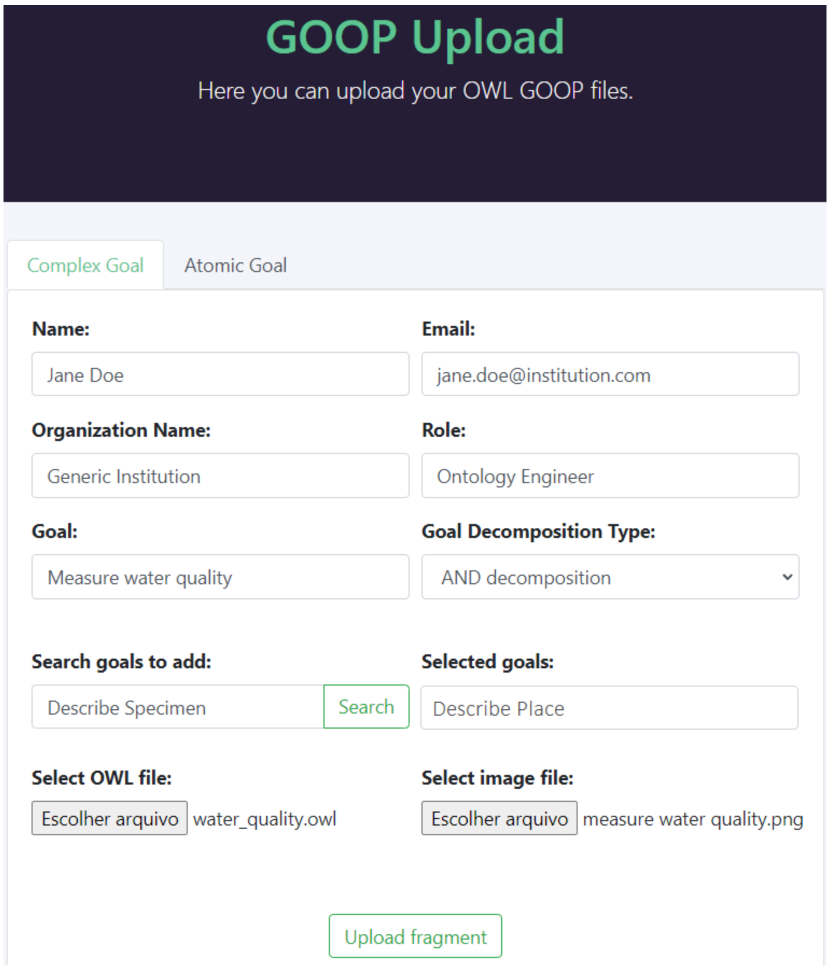
GOOP Upload

Here you can upload your OWL GOOP files.

| | |
|---|---|
| **Name:** Your name | **Email:** Email |
| **Organization Name:** Federal University of ... | **Role:** Researcher, Ontology Engineer, Student |
| **Goal:** Goal Name | **Goal Decomposition Type:** |
| **Search goals to add:** Search for goals | **Selected goals:** |
| **Select OWL file:** Escolher arquivo Nenhum arquivo selecionado | **Select image file:** Escolher arquivo Nenhum arquivo selecionado |

Figure 9. Selecting Complex Goal option to create a new GOOP

In the page shown to the user, he/she needs to provide the required information. The main difference to recording a GOOP related to an atomic goal is that the user needs to indicate the goal composition. For that, he/she must inform the goal composition type on the "Goal Decomposition Type" field, by choosing between an AND (all goals must be satisfied for the complex goal to be achieved) or OR composition (at least one goal must be satisfied for the complex goal to be achieved) and forming the goal tree. The goals that compose the complex goal related to the GOOP (i.e., the goals that compose the goal tree) must be recorded beforehand, with their respective GOOPs, so that the user can reuse them to record the GOOP related to the complex goal by searching for them and selecting the desired ones in the "Search goals to add" field. The selected goals will be shown beside it, in the "Selected goals" field. Figure 10 shows, as an example, a page containing data to record the GOOP "Measure Water Quality".


GOOP Upload

Here you can upload your OWL GOOP files.

| | |
|---|---|
| **Name:** Jane Doe | **Email:** jane.doe@institution.com |
| **Organization Name:** Generic Institution | **Role:** Ontology Engineer |
| **Goal:** Measure water quality | **Goal Decomposition Type:** AND decomposition |
| **Search goals to add:** Describe Specimen | **Selected goals:** Describe Place |
| **Select OWL file:** water_quality.owl | **Select image file:** measure water quality.png |

Figure 10. Recording the GOOP Measure Water Quality, related to a complex goal

## 3.2. Searching GOOPs for reuse

For searching GOOPs for reuse, the user must click the search icon on the top right corner, as shown in Figure 11.

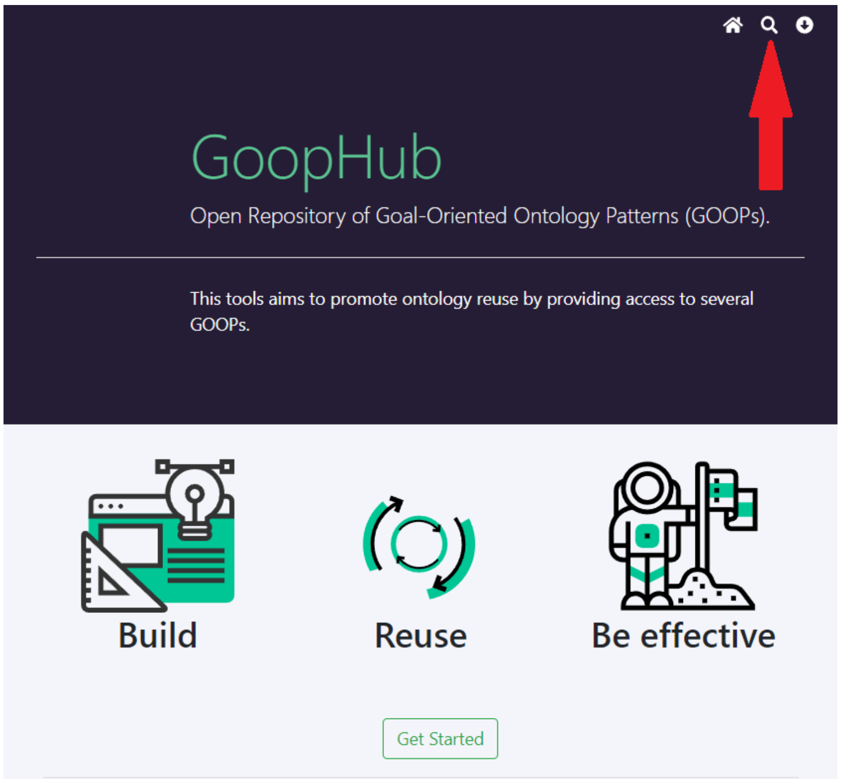

GoopHub
Open Repository of Goal-Oriented Ontology Patterns (GOOPs).
This tools aims to promote ontology reuse by providing access to several GOOPs.
Build
Reuse
Be effective
Get Started

Figure 11. How to access the page for searching for GOOPs

GOOP-Hub adopts a goal-based search, i.e., GOOPs are retrieved based mainly on the goals they achieve. Thus, in the search page, the user must inform the goal he/she want to achieve. For example, Describe place. The search will return all GOOPs associated to goals similar to the one informed by the user. The search engine considers similarity between strings to return the results. Figure 12 shows the GOOP-Hub search page. Figure 13 shows the GOOPs returned considered the informed search string.

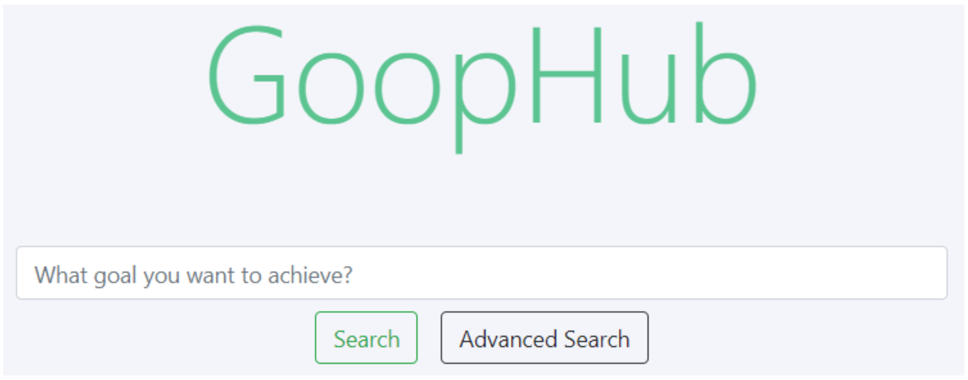


Figure 12. Search page

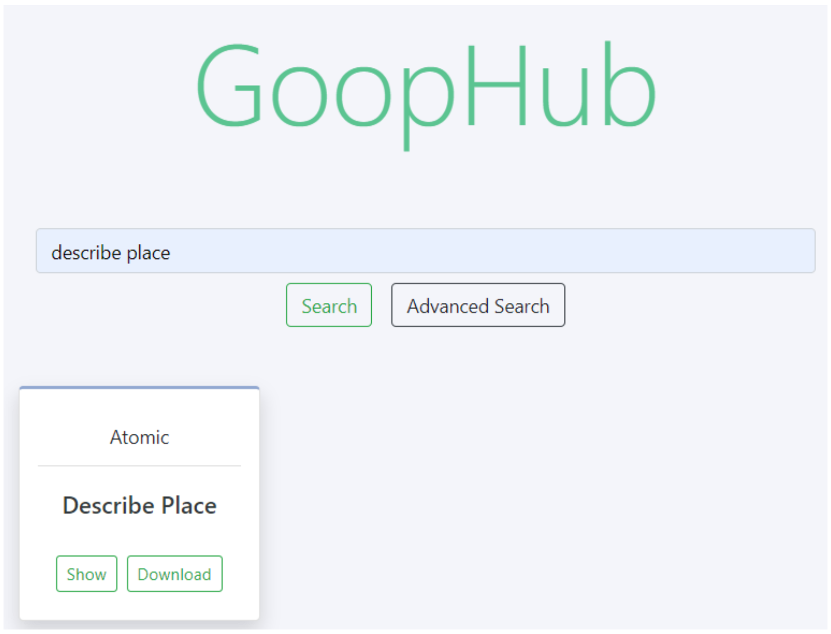
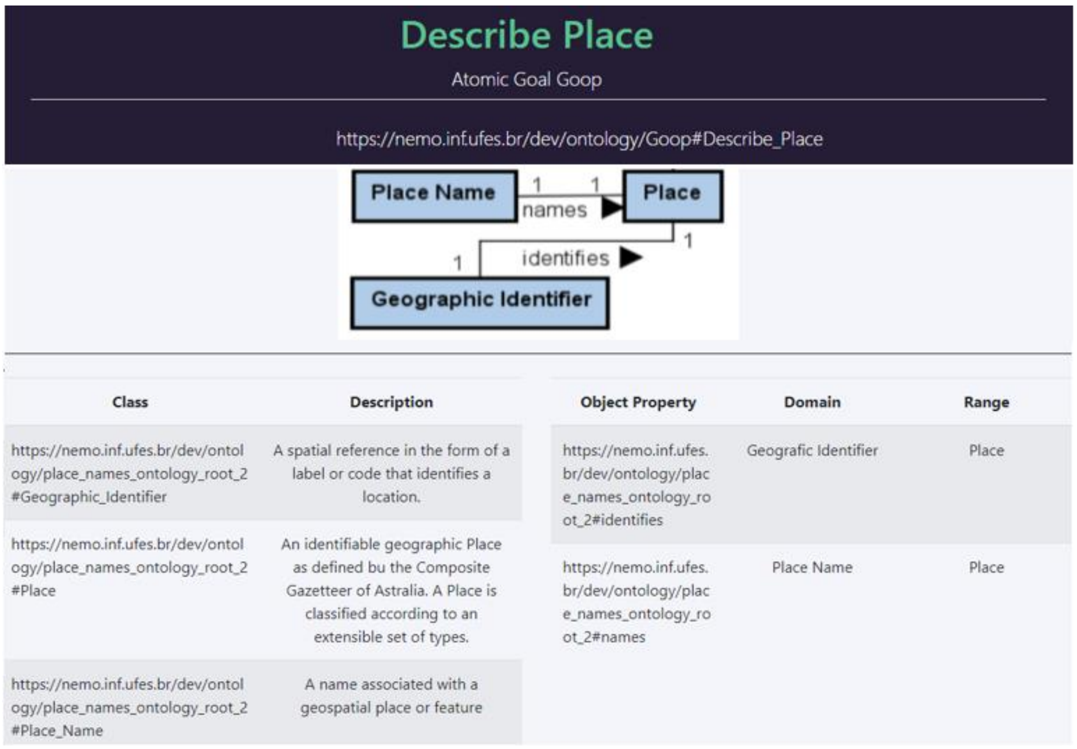


Figure 13. GOOPs returned in the search

The user can visualize information about the returned GOOPs by clicking the button Show in the corresponding card. He/she can also download the GOOP by clicking the button Download. Figure 14 shows the page detailing the GOOP "Describe Place".



Describe Place

Atomic Goal Goop

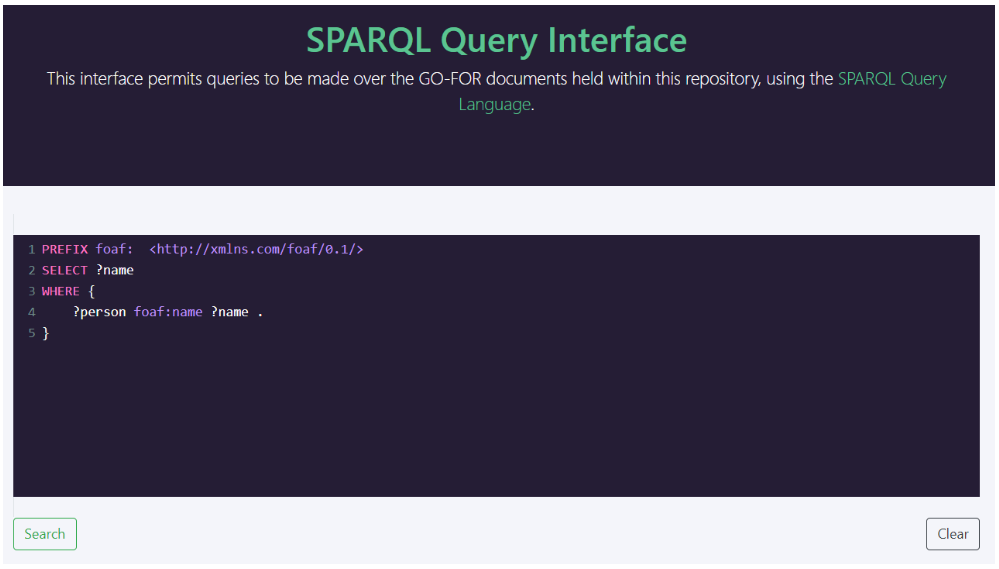
https://nemo.inf.ufes.br/dev/ontology/Goop#Describe_Place

| Class | Description |
|---|---|
| https://nemo.inf.ufes.br/dev/ontology/place_names_ontology_root_2#Geographic_Identifier | A spatial reference in the form of a label or code that identifies a location. |
| https://nemo.inf.ufes.br/dev/ontology/place_names_ontology_root_2#Place | An identifiable geographic Place as defined bu the Composite Gazetteer of Astralia. A Place is classified according to an extensible set of types. |
| https://nemo.inf.ufes.br/dev/ontology/place_names_ontology_root_2#Place_Name | A name associated with a geospatial place or feature |

| Object Property | Domain | Range |
|---|---|---|
| https://nemo.inf.ufes.br/dev/ontology/place_names_ontology_root_2#identifies | Geografic Identifier | Place |
| https://nemo.inf.ufes.br/dev/ontology/place_names_ontology_root_2#names | Place Name | Place |

Figure 14. Page detailing the returned GOOP

In addition to the standard search, searches can also be done by using the advanced search feature, which allows the user to use SPARQL to explore the repository. It is also possible to use reasoner and all SPARQL and StarDog exclusive features. The documentation of SPARQL can be found in https://www.w3.org/TR/rdf-sparql-query/. Figure 15 presents the page where the user can input SPARQL code and run advanced searches.



SPARQL Query Interface

This interface permits queries to be made over the GO-FOR documents held within this repository, using the SPARQL Query Language.

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?person foaf:name ?name .
}
```

Figure 15. Advanced search

## 3.3. Filtering candidate GOOPs for reuse

After running the search (standard or advanced) for GOOPs, the user can filter the returned GOOPs aiming at selecting the one more suitable for his/her needs. Figure 16 illustrates the search page with the filtering options placed in the left side.

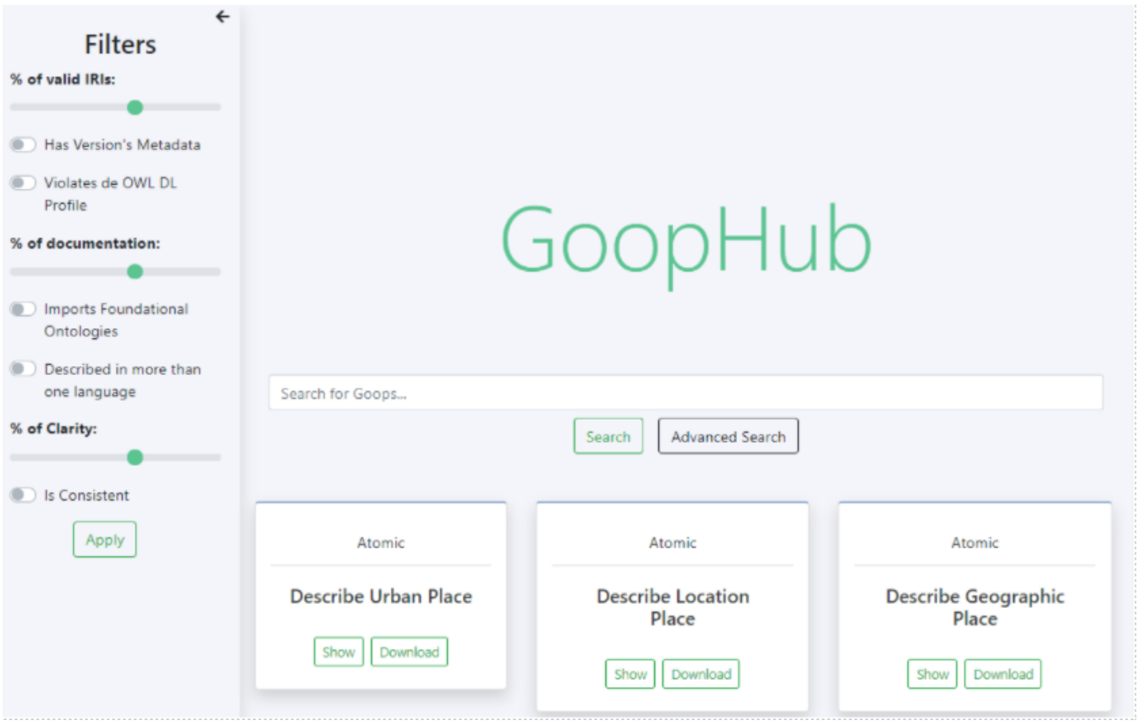
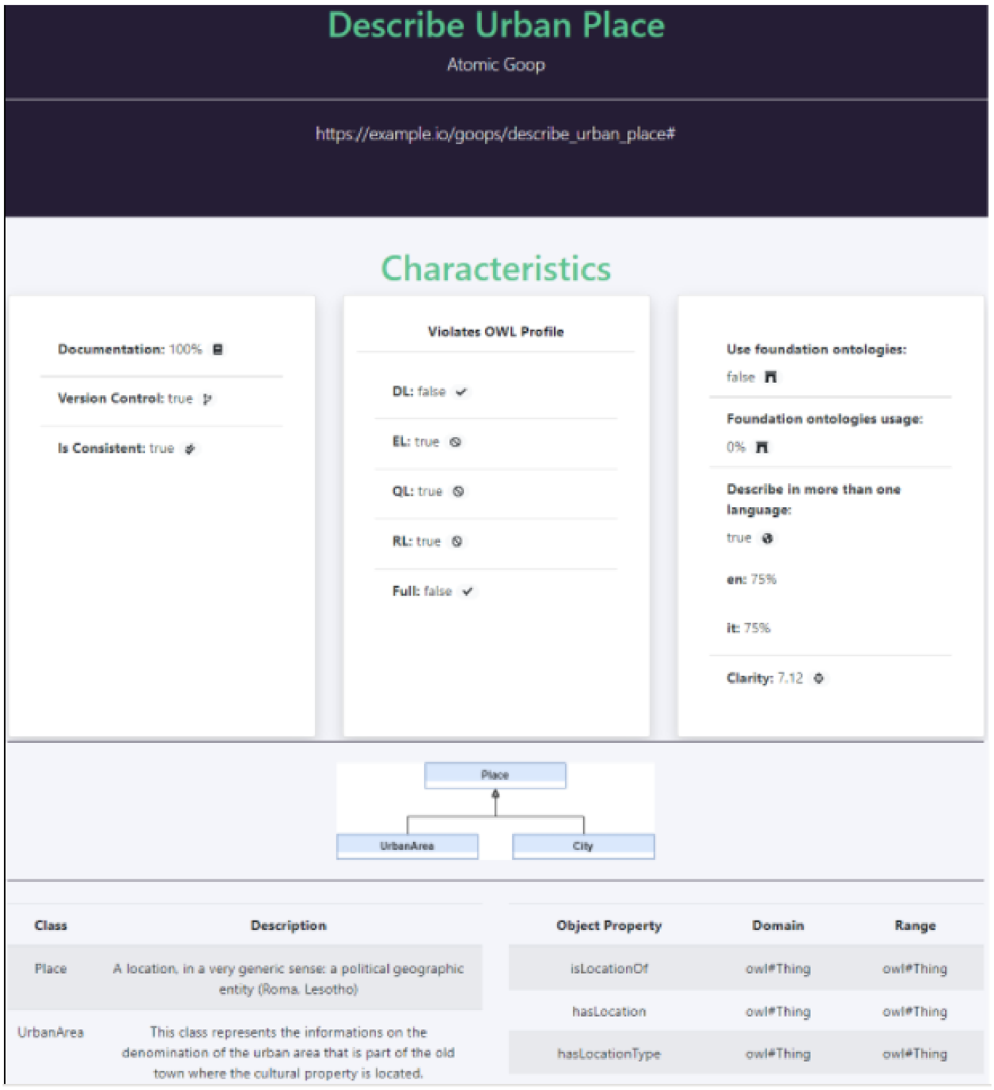


Figure 16. Page to filter returned GOOPs

By using the filters, the user can filter the returned GOOPs by using the following criteria:

valid IRIs, foundational ontologies reuse, clarity, consistency, and description in more than one language, among others. To visualize the values assigned to each property of a particular GOOP, the user must click the button Show in the GOOP card. At the visualization page, the user can view the ontology diagram as an image, the value given to each characteristic that was used to find the GOOP and all concepts addressed in the GOOP with names, description, object properties and data properties.



Describe Urban Place

Atomic Goop

https://example.io/goops/describe_urban_place#

Characteristics

Documentation: 100%

Version Control: true

Is Consistent: true

Violates OWL Profile

DL: false

EL: true

QL: true

RL: true

Full: false

Use foundation ontologies: false

Foundation ontologies usage: 0%

Describe in more than one language: true

en: 75%

it: 75%

Clarity: 7.12

| Class | Description |
|---|---|
| Place | A location, in a very generic sense: a political geographic entity (Roma, Lesotho) |
| UrbanArea | This class represents the informations on the denomination of the urban area that is part of the old town where the cultural property is located. |

| Object Property | Domain | Range |
|---|---|---|
| isLocationOf | owl#Thing | owl#Thing |
| hasLocation | owl#Thing | owl#Thing |
| hasLocationType | owl#Thing | owl#Thing |

Figure 17. Visualization of a GOOP and its properties

Note: this feature (filtering candidate GOOPs for reuse) is under development and thus it is not available yet.