Towards a Characterization to Aid in Ontology Reuse

Gabriel G. Nogueira, Monalessa P. Barcellos, Vítor E. Silva Souza

Ontology and Conceptual Modelling Research Group (NEMO), Computer Science Department – Federal University of Espírito Santo
Vitória, ES, Brazil

gabriel.g.nogueira@aluno.ufes.br, monalessa@inf.ufes.br, vitor.souza@ufes.br

Abstract. *Developing ontologies is not a trivial task. Reusing existing ontologies can be helpful in this matter. However, with the high number of ontologies available on the web, selecting the ontology that best fits the needs of the ontology engineer is still a complex task. Knowing the characteristics of the candidate ontologies can help the ontology engineer better decide which one to reuse. In this sense, this paper gives the first step toward an ontology characterization to aid reuse. We propose a set of properties that can be easily accessed and interpreted by the ontology engineer to make a decision about which ontology to reuse. We exemplify the use of the proposed characterization to select ontology patterns by using the GoopHub tool.*

1. Introduction

Nowadays, ontology engineers are supported by a wide range of ontology engineering methods and tools. However, building ontologies is still a complex task even for experts [Noppens and Liebig 2009]. Some of the main reasons for that are: (i) the ontology engineer must have a consistent and mature view of the domain being represented; (ii) ontology representation languages (e.g., Web Ontology Language (OWL), Resource Description Framework (RDF)) are usually not very expressive and leave much room for interpretation; and (iii) even when we aim at developing reference ontologies (i.e., conceptual models without concern with computational properties), developing the right conceptual model is also known to be quite complex.

Moreover, the emergent scenario has required more comprehensive and high-quality ontologies to solve problems involving semantic issues. In this context, developing a new ontology by reusing existing ontologies may be useful. Ontology reuse allows speeding up the ontology development process, saving time and money, and promoting the application of good practices. However, ontology reuse, in general, is a hard research issue, and one of the most challenging and neglected areas of Ontology Engineering [Fernández-López et al. 2019]. For example, ontology engineers still face problems to find and select the right ontologies for reuse and integrate several ontologies into a new ontology [Park et al. 2011].

In a recent study, Fernández-López et al. (2019) point out that although ontology reuse is recommended in the community, in practice it is not yet consolidated. Factors such as language heterogeneity, deficiencies in the documentation and lack of information about the ontology are obstacles for finding and reusing ontologies. Even though one of

the main characteristics of ontologies has been claimed to be reusability, the practice has shown that it has not been achieved yet.

In the literature, there are some ontology engineering methods that include reuse as an important step to develop new ontologies (e.g., SaBiO [Falbo 2014], NeOn [Suárez-Figueroa et al. 2012]). Furthermore, the use of ontology patterns (OPs) has been an emerging approach for ontology reuse, favoring the reuse of encoded experiences and good practices. OPs are ontology fragments referring to modeling solutions to solve recurrent ontology development problems. Experiments, such as the ones presented in [Blomqvist et al. 2009], show that ontology engineers perceive OPs as useful, and that the quality and usability of the resulting ontologies are improved.

Although there are several ontologies and OPs available for reuse (e.g., OPs of the NeOn project[1] [Suárez-Figueroa 2012] and Linked Open Vocabularies (LOV) [Vandenbussche 2017]), to select the ones suitable for reuse when developing a new ontology is still a challenging task [Fernández-López et al. 2019]. Aiming to aid in this matter, Reginato et al. (2019) advocate the use of GORE (Goal-Oriented Requirements Engineering) to make the ontology design rationale explicit, providing a notion of the kind of knowledge that is represented in the ontology. In this context, they propose GO-FOR (Goal-Oriented Framework for Ontology Reuse), which uses goals as the main element to select OPs for reuse. Hence, in GO-FOR, OPs are goal-oriented ontology patterns (GOOPs), i.e., OPs related to the goals that establish the scope addressed by the ontology fragment. GOOPs are stored in a goal-oriented ontology pattern repository (GOOPR) and can be reused based on the goal to which they relate.

Even though a goal-based search for selecting ontologies is a promising approach [Reginato et al 2019], it may not be enough to obtain the best ontology for reuse. In order to properly select the ontology (or OP) that best fits a certain ontology development need, it is necessary to know additional information about it. In this sense, some ontology search approaches consider structural properties such as class matching and semantic similarity [Alani and Brewster 2005]. Others, analyze the ontology as a graph (e.g., [Alani and Brewster 2006] and [Park et al. 2011]). There are also works that take popularity properties into account (e.g., [Ding et al. 2004]). However, a popular ontology does not necessarily indicate a good representation of the concepts it covers. Popularity does not necessarily correlate with good or appropriate representations of knowledge [Alani and Brewster 2006]. Furthermore, structural properties may not be easy to interpret and can drive the ontology engineer not to choose the best ontology for reuse.

Considering the need for a set of properties that can be used to provide relevant information about ontologies and support selecting the ones more suitable for reuse when developing a new ontology, in this work, we give the first step towards a characterization to aid in ontology reuse. Our goal is to reach a set of properties that provide the ontology engineer with useful and accessible information that characterizes ontologies and helps him/her make decisions about which ones to reuse. To illustrate the use of our initial proposed characterization, we applied it to select GOOPs in the GoopHub, the computational tool that supports GO-FOR use.

[1] http://ontologydesignpatterns.org/

This paper is organized as follows: Section 2 provides the background for the paper, addressing ontology reuse and introducing GO-FOR; Section 3 presents the proposed characterization; Section 4 exemplifies the use of the proposed characterization in the GO-FOR context; Section 5 discusses related works; and Section 6 presents our final considerations, points out future work and concludes the paper.

2. Background

2.1. Ontology Reuse

Reusability has long been recognized as a key attribute of ontologies, yet the principles and practice of reuse remain underdeveloped. The current lack of design through reuse presents a serious problem for the ontology community. There is not even a formal and consensual definition of ontology reuse within the community [Katsumi and Grüninger 2016]. In general, reuse can be defined as the process in which available ontological knowledge is used as input to generate new ontologies [Bontas et al. 2005]. It is a special case of design. Intuitively, it refers to the task of taking some existing ontology and manipulating it in some way in order to satisfy the design requirements. Some more specific, related, and sometimes overlapping subtypes of reuse have been defined, such as merging and alignment, integration, modular or safe reuse, and the application of ontology patterns [Katsumi and Grüninger 2016].

Ontology Patterns (OPs) are an emerging approach that favors the reuse of encoded experiences and good practices [Falbo et al. 2013]. Patterns are vehicles for encapsulating knowledge. They are considered one of the most effective means for naming, organizing, and reasoning about design knowledge. According to Buschmann et al. (2007), a pattern describes a particular recurring problem that arises in specific contexts and presents a well-proven solution for the problem.

Currently, there are several ontologies that attempt to model the same domain (or portion of the domain), yet varying the modeling, concepts and relations between the concepts. This creates a problem in reusing existing ontologies since the ontology engineer would have to sift through several ontologies in order to select the ones to reuse [Hlomani and Stacey, 2014]. Aiming to resolve this problem, some works propose properties that can be used to help the ontology engineer in the process of choosing the ontology that best fits his/her necessity. For example, Buitelaar et al. (2004) propose OntoSelect, which allows for searching ontologies for a given knowledge markup task based on coverage, structure, and connectedness. Coverage refers to the number of classes and properties that can be matched by search terms. Structure is given by the number of properties relative to the number of classes of the ontology. Connectedness refers to the number of imported ontologies. Park et al. (2011) propose an approach for ontology selection and ranking based on semantic and lexical matching. It uses measures proposed in [Alani and Brewster 2005] and adds other two that can improve the selection of ontologies, namely, relation match and taxonomy match. The former calculates the degree of semantic similarity of a relationship between search terms in an ontology, and the latter is meant to evaluate the degree of semantic similarity of a taxonomic relation in a taxonomy structure.

Some works define properties to assess aspects of ontology quality. Although not focused on reuse, the properties can be used to evaluate an ontology and verify if it is good enough to be reused. Burton-Jones et al. (2005), for example, propose metrics (e.g., history, authority, accuracy) to assess the syntactic, semantic, pragmatic, and social aspects of ontology quality. D'aquin and Gangemi (2011), in turn, present some characteristics generally present in "beautiful ontologies". According to the authors, a beautiful ontology is one that reflects an elegant solution for modeling a problem and it is at the same time good (in terms of formal quality), usable and practicable. Have a good coverage of the targeted domain, be often easily applicable, and be structurally well designed are some of the characteristics pointed out by the authors.

2.2. Goal-Oriented Framework for Ontology Reuse (GO-FOR)

The search and selection of ontologies (or OPs) to be reused to develop a new ontology should consider the alignment between their scope and the scope of the ontology to be developed. Therefore, Reginato et al. (2019) argue that ontology reuse should be driven by ontology requirements. Based on that, they proposed GO-FOR, a framework that uses goals as the main element to select OPs for reuse. Goal-oriented ontology patterns (GOOPs) are the basic elements of GO-FOR. A GOOP consists of an ontology fragment wrapped by a goal. In other words, it refers to an ontology model fragment that can be used to achieve a goal. A GOOP can be created whether using an ontology model fragment already built (i.e., a fragment of an existing ontology can be used to achieve a goal, giving rise to a GOOP) or building the model fragment from scratch (i.e., a model fragment is built aiming to achieve a goal).

GOOPs are stored in a goal-oriented ontology pattern repository (GOOPR) and relate to each other according to the relationships between their goals. When developing a new ontology, the ontology engineer can search the GOOPR for GOOPs to be reused to address the scope of the new ontology. He/she defines the goals to which the ontology is committed by developing its goal models and uses the goals as a basis to search for GOOPs. This search involves comparing the goals of the new ontology to the goals of GOOPs stored in the GOOPR, to identify matchings between them (i.e., to find GOOPs that meet the goals). Figure 1 shows an overview of GO-FOR conceptual architecture.

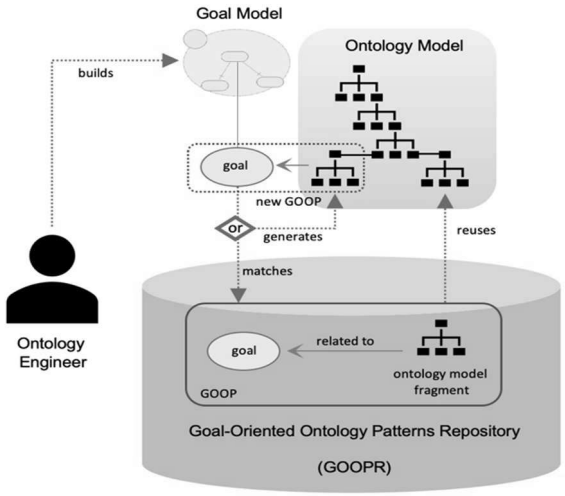

Figure 1. GO-FOR architecture [Reginato et al., 2019].

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]). 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 into 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.

3. Ontology Characterization aiming at Reuse

Building an ontology through reuse depends on finding suitable ontologies for being reused. This is one of the most challenging tasks in ontology reuse, particularly when the ontology engineer has more than one candidate ontology [Park et al. 2011] [Fernández-López et al. 2019]. Hence, we proposed a set of properties to characterize ontologies and help the ontology engineer select the one that best fits his/her needs. The properties were defined based mainly on the works by Gangemi et al. (2005), Burton-Jones et al. (2005), Obrst et al. (2007), d'Aquin and Gangemi (2011), Porn et al. (2016) and De Freitas et al. (2019). It is worth clarifying that we do not intend to define an exhaustive set of properties. Contrariwise, our purpose is to reach a set of properties that can be easily interpreted and accessed, requiring little effort from the ontology engineer. In this sense, we advocate that most of the properties should be able to be automatically obtained. Hence, an operational version of the ontology (e.g., OWL, RDF, RDFS) to be characterized is necessary.

Table 1 shows the current proposed set of properties. For each property, the table presents its name, a brief description, how it is calculated/collected, and the main reference we considered to define it.

Table 1. Properties to characterize ontologies

Name	Description	Calculation	Based on
Applicability	Indicates the effort degree (low, medium, high) necessary to use the ontology. More complex ontologies tend to require more effort to be reused.	Ontology engineers that have reused the ontology must inform (manually) the effort needed to reuse it. Applicability is obtained by assigning the values 1, 2, and 3, respectively, to the low, medium, and high effort degree and calculating the average of the values informed by different ontology engineers.	[Gangemi et al. 2005]
Clarity	Identifies how clear and non-ambiguous is the ontology, based on the terms used to name classes and object properties. Terms that have many meanings often open	Let C the name of the class or property in the ontology. For each C, count A (the number of word senses that the term has in WordNet [Miller 1998]). Then Clarity = A/C.	[Burton-Jones et al. 2005]

Name	Description	Calculation	Based on
	more space for misinterpretation. Thus, the higher the number of different meanings of the same term, the lower the ontology clarity.	The return is the average number of words senses, a value next to 0 mean that the ontology has more clarity.	
Consistency	Indicates the possibility of reaching contradictory conclusions in an ontology, from valid input data. When the ontology engineer creates instances of concepts of an inconsistent ontology, he/she may find invalid statements from axioms.	Obtained by using the Hermit reasoner [Shearer et al., 2008] in order to search for inconsistency inside the ontology. The reasoner returns false in case the ontology allows contradictory conclusions. Otherwise, it returns true.	[Porn et al. 2016]
Described in more than one language	Indicates if the ontology is described in more than one language and the percentage of concepts described in each used language. This property helps the ontology engineer verify if the ontology is described in the same language, he/she has used to develop the new ontology or in another language he/she is familiar with.	Obtained by verifying if the concepts' *rdfs:label* or *rdfs:comment* properties are declared using one or more language. In this case, the return is true, and the percentage of concepts written in each language is calculated.	[De Freitas et al. 2019]
Foundational Ontologies reuse	Identifies if the ontology reuses foundational ontologies and the percentage of concepts declared using classes or object properties of the foundational ontology. Reusing foundational ontologies usually indicates a well-founded ontology.	Obtained by checking if the ontology imports foundational ontologies. If so, the percentage of classes and object properties that are *rdfs:subClassOf* or *rdfs:subPropertyOf* a foundational ontology entity is calculated. Otherwise, the return is false.	[D'aquin and Gangemi 2011]
Has documentation	Indicates the percentage of classes and object properties that have comments, descriptions or labels documenting them. Well-documented ontologies are often easier to understand.	Verify if the ontology concepts contain *rdfs:label* or *rdfs:comment* explaining or presenting examples of how to use them. If so, the return is true and the percentage of concepts with description or example to use is calculated. Otherwise, the return is false.	[De Freitas et al. 2019]
Has version control	Indicates if the ontology version is identified. This information allows the ontology engineer to identify the reused ontology version, even when other versions of the ontology are available.	Return true if the ontology has the *annotationProperty owl:versionInfo*. Otherwise, the return is false.	[De Freitas et al. 2019]

Name	Description	Calculation	Based on
Has violation	Indicates if the ontology violates OWL profiles, which can increase the reasoning complexity and hamper the efficiency of a reasoner.	For each OWL Profile (DL, RL, QL, EL and Full) check if the ontology violates the profile. The return is true or false for each checked profile. Obtained by using the Hermit reasoner [Shearer et al., 2008] combined with the OWL API to check for violations.	[Obrst et al. 2007]
Published by	Informs the person, group or organization that published the ontology. Ontologies developed by trusted people, group or organization tend to be more reliable and have some support in case of doubts.	The name of the person, group or organization is manually informed.	[De Freitas et al. 2019]
Valid IRIs	Indicates if the ontology redirects to valid IRIs. This information is useful to avoid the reuse of ontologies that have not been maintained.	Obtained by calculating the percentage of valid IRIs (i.e., IRIs that return an HTTP response that does not mean error (e.g., 200, 300)).	[De Freitas et al. 2019].

4. Applying the Proposed Characterization in GO-FOR

Aiming to demonstrate the use of the proposed characterization, we applied it in GO-FOR to characterize GOOPs and aid GOOPs selection for reuse. The use of GO-FOR is supported by a tool called GoopHub [Reginato et al. 2019]. The tool allows ontology engineers to select GOOPs for reuse (i.e., development with reuse) as well as store new GOOPs that are made available for future reuse (i.e., development for reuse). Originally, the search for GOOPs in the GoopHub was based only on the goal the ontology engineer wanted to achieve. For example, if the ontology engineer needed a GOOP to "describe location", he/she provided a string representing his/her goal as an input and the GoopHub returned the GOOPs that meet the search string. However, this type of search may result in a high number of GOOPs, requiring the ontology engineer to analyze many ontology fragments to identify the one that best fits his/her needs to develop the new ontology, which demands effort and time. As a consequence, the ontology engineer may not select the best GOOP for his/her needs or may even give up the reuse. Thus, we decided to improve the search for GOOPs by implementing a new feature that enables GOOP's characterization and extending the search to allow ontology engineers to apply filters based on the GOOPs properties, helping them in the decision of which GOOP will be reused.

To implement these improvements, we first extended the GoopHub metamodel proposed in [Reginato et al. 2019]. Figure 2 shows the extended GoopHub metamodel. Concepts in pink (darker background) were added to enable GOOPs characterization. *Characteristic* was added to the model to represent the properties we proposed to characterize ontologies. *GOOPCharacteristic*, in turn, is necessary to store the *value* of a given characteristic when referring to a particular GOOP. By extracting the values of all

characteristics of a certain GOOP, we obtain the GOOP characterization. The complete description of the GoopHub metamodel can be found in [Reginato et al., 2019].

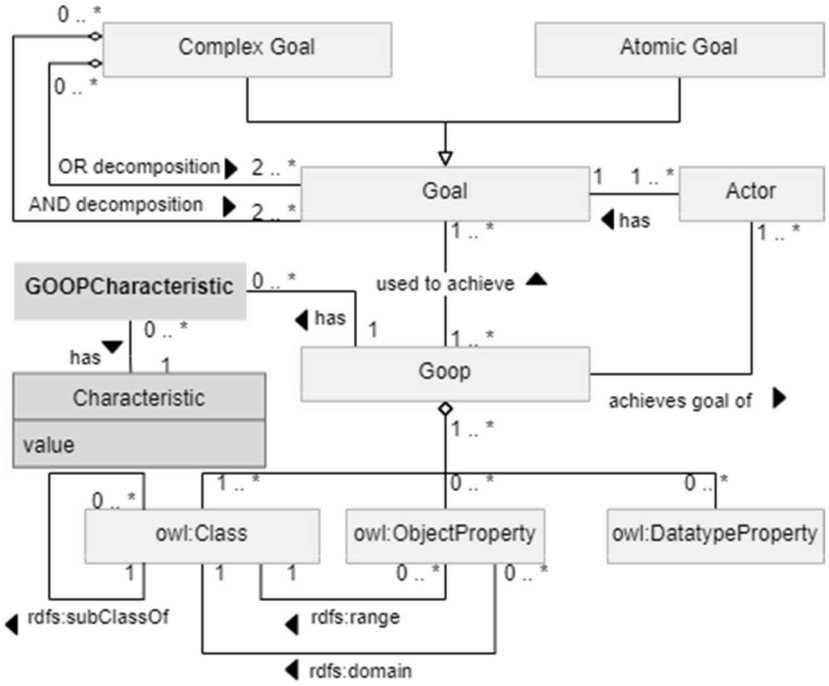

Figure 2. The GoopHub metamodel (extended from [Reginato et al. 2019]).

To store a GOOP in the GoopHub (i.e., development for reuse), the ontology engineer must register the GOOP by informing its name and the goal to which it is related, and uploading the image of the GOOP conceptual model and the GOOP OWL file. Thus, the characterization feature extracts from the OWL file the automatic GOOP properties (see Table 1). Manual properties must be informed by the ontology engineer. The uploaded OWL file is then converted to the GOOP metamodel and data is inserted in a triple store database. This way, the GOOP is made available for retrieval and reuse. When developing a new ontology, the ontology engineer searches for GOOPs to be reused (i.e., development with reuse) by informing the goal he/she wants to achieve. Thus, the GoopHub returns the GOOPs that satisfy that goal and the ontology engineer can filter the returned GOOPs by using the properties he/she considers relevant for achieving his/her needs. As a result, the number of candidate GOOPs is reduced and the ontology engineer can reach a more effective decision.

To extract the values for the automatic properties from the GOOP OWL file, we followed the calculation procedures indicated in Table 1 and used the OWL API (https://github.com/owlcs/owlapi). The API has features to navigate and manipulate the concepts declared on the OWL file.

To illustrate the use of the proposed characterization to support the selection of GOOPs, we considered a scenario in which an ontology engineer is developing an ontology for the soccer championship domain and, in this context, he/she needs to describe the locations where the matches will take place (city, state, country, stadium, etc.). Thus, the ontology engineer used the GoopHub to search for GOOPs able to meet the "Describe Place" goal. The search returned three GOOPs, namely: "Describe Urban

Place", "Describe Location Place" and “Describe Geographic Place". Figure 3 shows the GoopHub search page and the three GOOPs returned in the search.

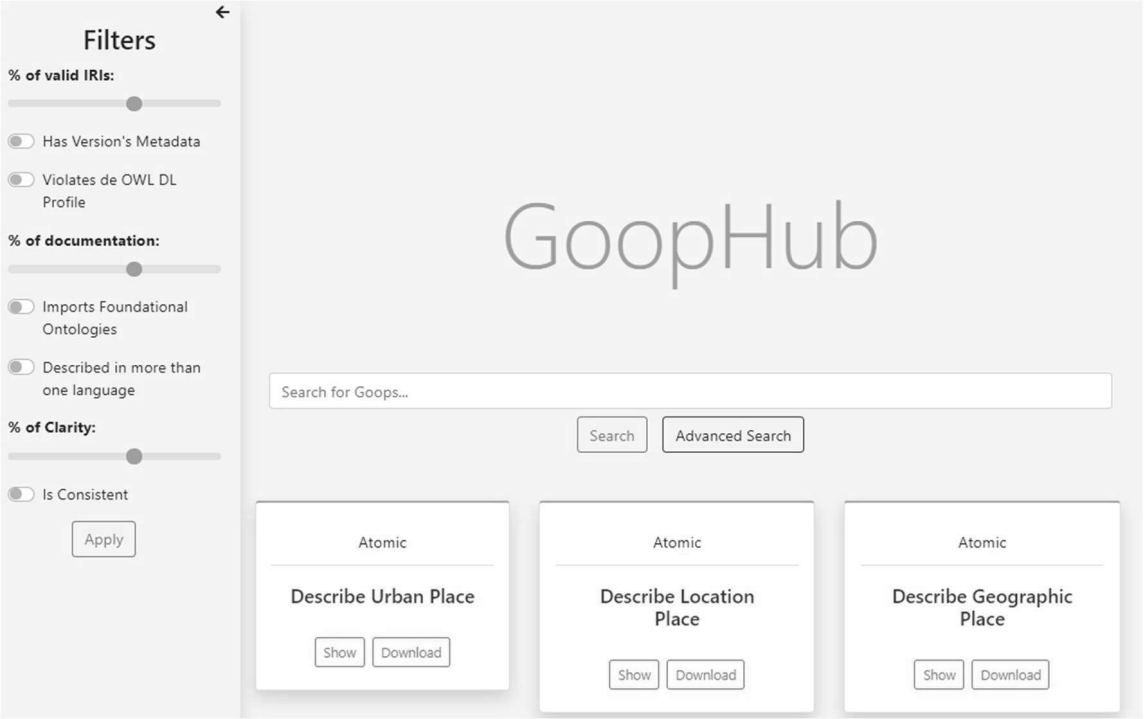

Figure 3. GoopHub search page showing three returned GOOPs.

Aiming to decide which of the returned GOOPs best fits his/her needs, the ontology engineer takes advantage of the GOOPs characterization and applies the filters available in the left sidebar of the page to help him/her select the GOOP more suitable for the ontology under development. As the ontology engineer changes the filter parameters, the search result is updated to include only the GOOPs that meet the properties values indicated by the ontology engineer. After applying the filters, the ontology engineer reached the “Describe Urban Place” as the GOOP suitable for reuse in the soccer championship domain ontology.

Besides applying the filters, by clicking the “Show” button in the GOOP card, the ontology engineer has access to a detailed view of the GOOP, which includes, among others, an image of its conceptual model, the description of its concepts, and the value of each property. Thus, if the search returns more than one GOOP even after the ontology engineer applying the filters, he/she can analyze each GOOP in detail to make a decision about which one to reuse. Figure 4 shows a partial view of the page providing a detailed view of the “Describe Urban Place” GOOP.

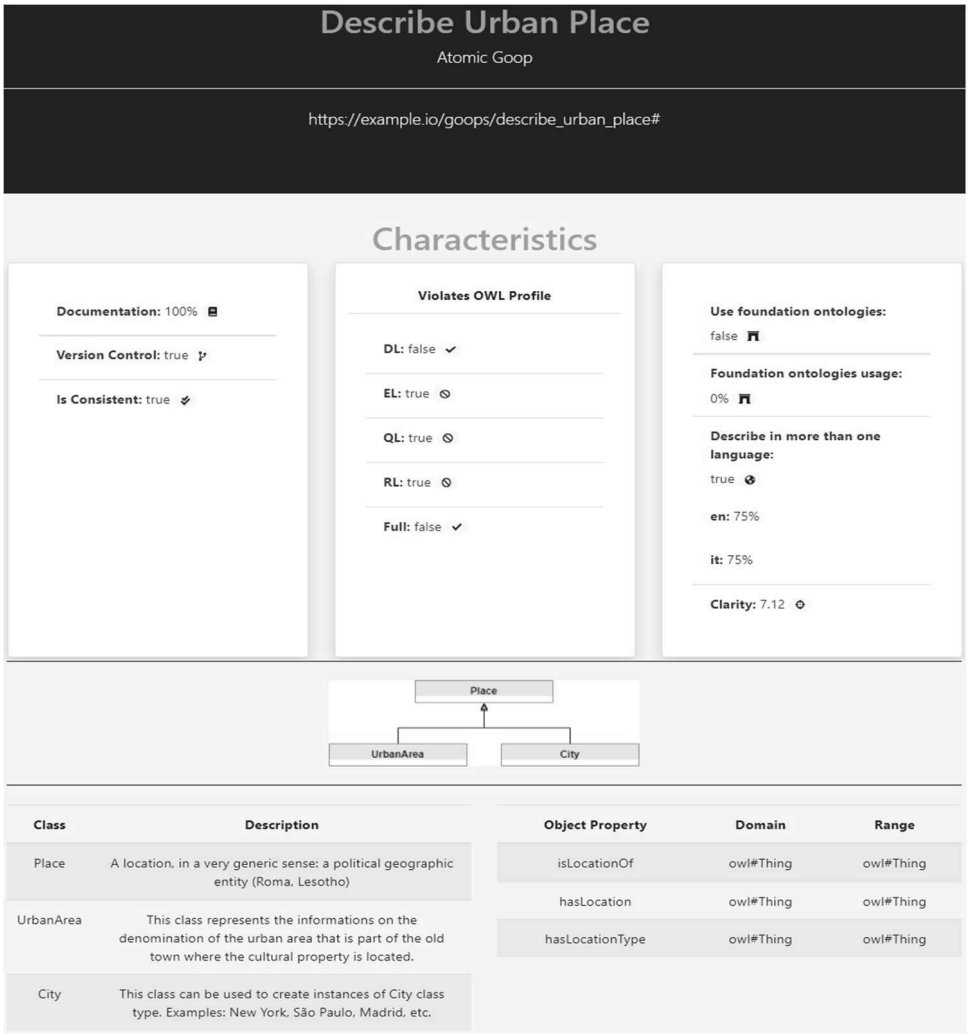

Figure 4. GoopHub page showing the GOOP "Describe Urban Place".

Table 2 summarizes the values of the properties automatically obtained for the GOOPs returned in the search considered in the example discussed in this section.

Table 2. Characteristics of each GOOP returned in the search

Properties	"Describe Location Place"	"Describe Urban Place"	"Describe Geographic Place"
Clarity	7.95	7.12	2.80
Computational efficiency	DL: False EL: True QL: True RL: True Full: False	DL: False EL: True QL: True RL: True Full: False	DL: True EL: True QL: True RL: True Full: False
Consistency	True	True	True
Described in more than one language	True English: 100% Italian: 100%	True English: 75% Italian: 75%	False English: 100%
Foundational Ontologies reuse	False	False	False

Properties	"Describe Location Place"	"Describe Urban Place"	"Describe Geographic Place"
Has documentation	80%	100%	100%
Has version control	True	True	True
Valid IRIs	11.36%	75%	100%

By analyzing the GOOPs characteristics, it is possible to notice that there are similarities among the GOOPs. However, an important distinction for the ontology engineer in this particular example is that he was interested in an ontology described in both, English and Italian. This led him to select the GOOP "Describe Urban Place", because it is consistent, documented, has the second-best value of clarity (the lower the value, the better the clarity), has version control, a good rate of valid IRIs and it is defined in English and Italian.

5. Related Work

In the literature, there are some works presenting proposals involving properties to evaluate ontologies considering different purposes. Some of them aim to rank ontologies. Ding et al. (2004), for example, proposed Swoogle, a Semantic Web search engine that crawls, indexes and stores Semantic Web documents in a triple store. It contains 10,000 ontologies and uses a PageRank-like method to rank ontologies by analyzing links and referrals between ontologies. Swoogle considers most popular ontologies the ones most referred. Also aiming at ranking ontologies, Alani and Brewster (2006) proposed AKTiveRank, an ontology ranking engine based on an internal analysis of the concepts in the ontologies. It applies four measures: class matching, centrality, semantic similarity and density. Another example is the work by Park et al. (2011), OntoRank, which proposes a ranking model based on better semantic matching capabilities and extends AKTiveRank by adding the measures relation matching and taxonomy matching. Although these works measures properties to evaluate and rank ontologies, they are not devoted to supporting reuse, as our work. Moreover, several measures used in these works focus on the ontology structure as a graph and can be hard to be understood by less experienced ontology engineers.

As we discussed in Section 2, some works have defined properties to help assess ontology quality. An example is the work by Burton-Jones et al. (2005), which proposes a quality model composed by ten properties (authority, accuracy, clarity, comprehensiveness, history, consistency, interpretability, lawfulness, relevance and richness) to provide a theory-based framework that developers can use to develop high-quality ontologies and that applications could use to choose appropriate ontologies for a given task. Gómez-Pérez (2001), in turn, considers five properties (consistency, completeness, conciseness, expandability and sensitiveness) as important to evaluate ontology quality, while Gangemi et al. (2005) consider other seven (cognitive ergonomics, compliance to expertise, compliance to procedures, computational integrity and efficiency, flexibility, meta-level integrity and organizational fitness). Since these works are concerned with ontology quality in a broader sense, they consider several properties that are difficult to be automatically obtained and do not provide an ontology repository or a search engine for ontologies. In fact, many of them depend on human interpretation to be measured.

In summary, our work differs from the ones intended to rank ontologies mainly because our focus is on higher-level properties (e.g., applicability, consistency). On the other side, our proposal differs from the works devoted to assessing ontology quality because, even focusing on higher-level properties like these works, we are more interested in properties that can be easily accessed and interpreted and, thus, we have prioritized the ones that can be automatically obtained (except for applicability, which is not amenable to automation). Moreover, different from all the cited works, our purpose is to define a set of properties that can aid the reuse of ontologies, reducing the effort of the ontology engineer and making smoother the process of selecting an ontology for being reused. In order to bring a more concrete and precise approach for measuring the properties, most of them can be obtained by processing the OWL ontology file.

6. Final Considerations

In this paper, we presented a set of properties to characterize ontologies. It is the first step towards an ontology characterization to aid reuse. In order to demonstrate the use of the proposal set of properties, we extended the GoopHub tool [Reginato et al., 2019] to enable it to support ontology pattern (particularly goal-oriented ontology pattern – GOOP) characterization and help ontology engineers in the selection of GOOPs when developing a new ontology with reuse. We proposed a total of ten properties, eight of them automatically obtained from the ontology OWL file. The characterization was implemented in the GoopHub and can be used as a new filter to search GOOPs.

In the literature, there are several approaches proposing properties to evaluate ontologies with different purposes, such as raking ontologies (e.g., [Park et al., 2019] or assessing their quality (e.g., [Gangemi et al., 2005]). We took some of these works into account and selected properties present in more than one approach, while aiming at an ontology engineer independent collection by obtaining the properties based only on the operational ontology. To demonstrate the viability of the proposed set of properties, we performed a proof of concept using a new filter feature developed in the GoopHub. This feature allows a refinement in the search of GOOPs, decreasing the effort to select a GOOP for reuse, particularly when several GOOPs are returned by the search. The new feature provided an improvement on the goal-based search, allowing the ontology engineer to refine the results.

The main contributions of the work addressed in this paper are the proposed set of properties used to characterize ontologies and the new version of GoopHub, containing a feature to automatize GOOP characterization and selection for reuse. It is important to emphasize that the work addressed in this paper is a work in progress and has limitations. For example, the proposal has only been applied to select GOOPs. As future work, we will extend the initial set of properties proposed in this paper to make it more comprehensive and improve the support to ontology selection. Aiming to identify properties to be added to the current set of properties, we intend to keep investigating the literature and to conduct a study with ontology engineers to identify properties they consider relevant when selecting an ontology for reuse. We also plan to perform a case study in a real setting by applying the new version of GoopHub to support ontology search and selection for reuse.

References

Alani, H., and Brewster, C. (2005). Ontology ranking based on the analysis of concept structures. In Proceedings of the third international conference on knowledge capture (K-CAP'05), Banff, Canada

Alani, H., and Brewster, C. (2006). Metrics for ranking ontologies.

Bontas, E.P., Mochol, M., Tolksdorf, R., Case Studies on Ontology Reuse, in: Proc. IKNOW05 Int. Conf. Knowl. Manag. (Vol. 74), 2005: pp. 345–353. https://doi.org/10.1016/j.sysarc.2006.02.002.

Buitelaar, P., Eigner, T. and Declerck, T. (2004). OntoSelect: A dynamic ontology library with support for ontology selection. In *In Proceedings of the Demo Session at the International Semantic Web Conference*.

Burton-Jones, A., Storey, V. C., Sugumaran, V. and Ahluwalia, P. (2005). A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering*, *55*(1), 84-102.

Buschmann, F., Henney, K., Schmidt, D.C., Pattern-Oriented Software Architecture: On Patterns and Pattern Languages, Vol. 5, John Wiley & Sons, 2007. https://doi.org/10.1093/intimm/dxu027.

De Almeida Falbo, R. (2014, September). SABiO: Systematic Approach for Building Ontologies. In *ONTO. COM/ODISE@ FOIS*.

De Freitas, M. L., Guizzardi, R. S. S. and Souza, V. E. S. (2019). GRALD: an Approach for Goal and Risk Analysis in the Development of Information Systems for the Web of Data. *J. Softw. Eng. Res. Dev.*, *7*, 4.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y. and Sachs, J. (2004, November). Swoogle: A semantic web search and metadata engine. In *Proc. 13th ACM Conf. on Information and Knowledge Management* (Vol. 304, pp. 10-1145).

D'Aquin, M. and Gangemi, A. (2011). Is there beauty in ontologies?. *Applied Ontology*, *6*(3), 165-175.

Falbo, R. D. A., Guizzardi, G., Gangemi, A. and Presutti, V. (2013, October). Ontology patterns: clarifying concepts and terminology. In *Proceedings of the 4th Workshop on Ontology and Semantic Web Patterns*.

Fernandes, P. C. B., Guizzardi, R. S. and Guizzardi, G. (2011). Using goal modeling to capture competency questions in ontology-based systems. *Journal of Information and Data Management*, *2*(3), 527-527.

Fernández-López, M., Poveda-Villalón, M., Suárez-Figueroa, M. C. and Gómez-Pérez, A. (2019). Why are ontologies not reused across the same domain?. *Journal of Web Semantics*, *57*, 100492.

Gangemi, A., Catenacci, C., Ciaramita, M. and Lehmann, J. (2005, December). A theoretical framework for ontology evaluation and validation. In *SWAP* (Vol. 166, p. 16).

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of intelligent systems*, *16*(3), 391-409.

Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, *45*(2), 61-65.

Hlomani, H. and Stacey, D. (2014, August). An extension to the data-driven ontology evaluation. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)* (pp. 845-849). IEEE.

Katsumi, M., Grüninger, M., What is ontology reuse?, in: Front. Artif. Intell. Appl., 2016: pp. 9–22. https://doi.org/10.3233/978-1-61499-660-6-9.

Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

Noppens, O. and Liebig, T. (2009, October). Ontology patterns and beyond: towards a universal pattern language. In *Proceedings of the 2009 International Conference on Ontology Patterns-Volume 516* (pp. 179-186).

Obrst, L., Ceusters, W., Mani, I., Ray, S. and Smith, B. (2007). The evaluation of ontologies. In *Semantic web* (pp. 139-158). Springer, Boston, MA.

Park, J., Oh, S. and Ahn, J. (2011). Ontology selection ranking model for knowledge reuse. *Expert Systems with Applications*, *38*(5), 5133-5144.

Porn, A. M., Huve, C. G., Peres, L. M. and Direne, A. I. (2016). A Systematic Literature Review of OWL Ontology Evaluation. In *15th International Conference on WWW/Internet* (pp. 67-74).

Reginato, C., Salamon, J., Nogueira, G., Barcellos, M., Souza, V. and Monteiro, M. (2019, July). GO-FOR: A Goal-Oriented Framework for Ontology Reuse. In *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)* (pp. 99-106). IEEE. Blomqvist, E., Gangemi, A., & Presutti, V. (2009, September). Experiments on pattern-based ontology design. In *Proceedings of the fifth international conference on Knowledge capture* (pp. 41-48).

Shearer, R., Motik, B. and Horrocks, I. (2008, October). HermiT: A Highly-Efficient OWL Reasoner. In *Owled* (Vol. 432, p. 91).

Suárez-Figueroa, M. C., Gómez-Pérez, A. and Fernández-López, M. (2012). The NeOn methodology for ontology engineering. In *Ontology engineering in a networked world* (pp. 9-34). Springer, Berlin, Heidelberg.

Vandenbussche, P. Y., Atemezing, G. A., Poveda-Villalón, M. and Vatant, B. (2017). Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semantic Web, 8(3), 437-452.