2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)

GO-FOR: A Goal-Oriented Framework for Ontology Reuse

Cássio C. Reginato, Jordana S. Salamon, Gabriel G. Nogueira, Monalessa P. Barcellos, Vítor E. S. Souza, Maxwell E. Monteiro
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espiríto Santo – Vitória – ES – Brazil
{cassio.reginato, jssalamon, monalessa, vitorsouza}@inf.ufes.br, gabriel.g.nogueira@aluno.ufes.br, maxmonte@ifes.edu.br

Abstract

Ontologies have been successfully used to assign semantics in the Semantic Web context and to enable integration of data from different systems or different sources. However, building ontologies is not a trivial task. Ontology reuse can help in this matter. The search and selection of ontologies to be reused should consider the alignment between their scope and the scope of the ontology being developed. In this paper, we discuss how goal modeling can be helpful in this context and we propose GO-FOR, a framework in which goals are the central elements to promote ontology reuse. We introduce goal-oriented ontology patterns as a new type of pattern to be applied to develop ontologies in a goal-oriented approach. Results of the use of GO-FOR to build an ontology used to integrate water quality data are also shown in this paper.

Keywords: Ontology, Reuse, Goal Modeling, Pattern

1. Introduction

An ontology is a formal, explicit specification of a shared conceptualization [1]. Ontologies have been recognized as conceptual tools of great importance in Computer Science since the end of the 1960s, mainly in areas such as Data Modeling (conceptual modeling) and Artificial Intelligence [2] [3]. In the last fifteen years, there has been an explosion of works related to ontologies in various segments of Computer Science. This has been motivated by the recognition of the importance of the use of ontologies in semantic interoperability tasks (e.g., system integration and data integration), in general, and by its role in the Semantic Web development, in particular. For instance, applications that process Linked Data available on the Web are programmed to understand well-known vocabularies (schemas, ontologies) at the time of their writing. Defining new vocabularies for every new data set we publish without properly linking to existing vocabularies makes this data unintelligible to existing applications. Reusing the appropriate vocabularies reduces heterogeneity by relying on ontological agreement [4].

Nowadays, ontology engineers are supported by a wide range of ontology engineering methods and tools. However, building ontologies is still a difficult task. Moreover, the emergent scenario has required more comprehensive and high-quality ontologies to solve problems involving semantic issues. Ontology reuse allows speeding up the ontology development process, saving time and money, and promoting the application of good practices [5]. However, reuse is a hard research issue and one of the most challenging areas of Ontology Engineering [6]. For example, ontology engineers still face problems to select the right ontologies for reuse and integrate ontology fragments into a new ontology [7].

One of the challenges of ontology reuse is the obscurity of the design rationale of the available ontologies [8]. Design rationale concerns the reasons for the decisions made during a design process [9]. Unknown design rationale makes it difficult to select the ontologies to be reused as well as to understand them, which is crucial to integrate them properly. We advocate the use of Goal-Oriented Requirements Engineering (GORE) to make the ontology design rationale explicit and promote ontology reuse [10]. GORE has been used to provide design rationale in Software Engineering, explaining from where a requirement came and which stakeholders' goals it meets [10]. In Ontology Engineering, we argue that GORE helps explain the rationale behind the ontology because motivational elements reveal the reasons for developing the ontology and provide a notion of the kind of knowledge is represented in the ontology.

In this paper, we introduce GO-FOR, a Goal-Oriented Framework for Ontology Reuse, in which we apply GORE in Ontology Engineering to express a design rationale to ontology model fragments. In GO-FOR, ontology models are depicted in self-contained fragments (i.e., domain ontology patterns) related to goals. These model fragments are self-contained ontology structures called goal-oriented ontology patterns (GOOP). Thus, goals can be used as parameters to support ontology shareability and reuse.

978-1-7281-1337-1/19/$31.00 ©2019 IEEE
DOI 10.1109/IRI.2019.00028

This paper is organized as follow: Section 2 provides the background for the paper; Section 3 presents the principles behind GO-FOR; Section 4 introduces GO-FOR; Section 5 addresses GO-FOR use and reports the application of GO-FOR to build an ontology to integrate water quality data in an Environmental Quality Research Project; Section 6 discusses related works; and Section 7 presents our final considerations.

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. Currently, there is not even a formal and consensual definition of ontology reuse within the community [11]. In general, reuse can be defined as the process in which available ontological knowledge is used as input to generate new ontologies [12]. It is as 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 [11].

Ontology Patterns (OPs) are an emerging approach that favors reuse of encoded experiences and good practices [13]. 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. [14], a pattern describes a particular recurring problem that arises in specific contexts and presents a well-proven solution for the problem. Thus, OPs are modeling solutions to solve recurrent ontology development problems [15]. Experiments, such as the ones conducted by Blomqvist et al. [16], show that ontology engineers perceive OPs as useful, and that by using OPs, the quality and usability of the resulting ontologies are improved.

Patterns are often considered and applied separately. However, no pattern is an island. Contrariwise, patterns are fond of company: sometimes with one pattern as an alternative to another, sometimes with one pattern as an adjunct to another, sometimes with a number of patterns bound together as a tightly-knit group [14]. Thus, when applying a pattern, it is important to understand and take its relationships into account [17].

2.2 Goal-Oriented Requirement Engineering

In this work, we propose the use of Goal-Oriented Requirement Engineering (GORE) to aid in ontology reuse. GORE emerged to create and study methods that approach requirements engineering from a goal-oriented perspective. Usually, in GORE, goals are elicited and represented in terms of some sort of model, using a notation provided by approaches such as iStar [18], KAOS [19] and Techne [20], among others.

GORE has been successfully applied in Requirements Engineering [10] and has also been used to enrich the requirements analysis phase of the ontology engineering process, as in [21] and [22]. The incorporation of explicit goal representations in requirement models provides a criterion for requirement completeness, i.e., the requirements can be judged as complete if they are sufficient to achieve the goals they refine [23].

In Ontology Engineering, GORE can help understand the domain of interest and the involved actors. For developing the goal models, the ontology engineer must identify the actors in the domain of interest and develop a goal model for each of them. Each goal model represents the actor goals and tasks to be addressed by the ontology, providing a comprehensive view of the ontology scope. Moreover, from goal models it is possible to derive competency questions, which the ontology must be able to answer and are used as a basis to develop the ontology conceptual model, to detail the ontology scope [21].

3. The GO-FOR Principles

Building an ontology through reuse depends on finding suitable ontologies for being reused [7]. The search and selection of ontologies to be reused should consider the alignment between their scope and the scope of the ontology to be developed. That is, for each ontology model candidate to be reused, it should be verified which part of it meets the requirements of the ontology to be built. Therefore, ontology reuse should be based on ontology requirements.

Considering that GORE can be used to establish ontology requirements and ontology patterns favor reuse, we propose GO-FOR, a goal-oriented and pattern-based framework to aid in ontology reuse. GO-FOR is based on four principles related to a subset of ontology design recurrent issues pointed out in [24], namely: (I_1) Which pieces of information about terms are critical for supporting shareability (e.g., name, textual definition, type, etc.)? (I_2) How to describe purposes of a particular ontology? (I_3) How to capture and use design rationale? (I_4) How to identify correspondences between ontologies? To address these issues, GO-FOR follows the principles (P1-P4) described below.

(P1) Standardize Terminology and Semantic Searching. This principle is related to issue I_1. It is difficult to say which piece of information about terms used to name ontology structures (e.g., models, patterns, concepts) is

most critical to support reuse, but it is easier to identify factors that can negatively influence it. One, for certain, is the use of arbitrary and inexpressive names for ontology structures. For instance, if an ontology engineer searches for an ontology about e-commerce and uses the term "e-commerce" to find ontologies addressing this subject, she may expect to find ontology models covering aspects like transactions, authentication, shopping and so on. However, such search could return, among others, the Good Relations ontology [25], whose description states that the ontology provides a vocabulary to e-commerce, when, in fact, it provides a vocabulary to specify offerings on the web, not addressing several aspects related to e-commerce [22]. The lack of a standard to name ontology structures harms the search and retrieval of suitable ontologies and, consequently, can compromise ontology reuse.

(P2) Apply Design Rationale. As discussed in Section 1, one of the difficulties to ontology reuse is the obscurity of the ontologies design rationale [26], which is directly related to issue I_3. It is important to make explicit the reasons for developing an ontology the way it was developed (e.g., what led the ontology engineer to include certain concepts in the ontology). In our view, addressing this problem is a key factor to ontology reuse. Moreover, making the design rationale explicit also contributes to solve issue I_2, since a design rationale approach can support to define ontology purpose.

(P3) Solve Overlaps. When ontology models are developed from scratch, without any concern with reuse, there is a high chance of overlaps with other ontology models. Thus, identifying and solving overlaps between ontologies is important to avoid redundancy and increase reuse. This relates to I_4, since overlap solving involves the establishment of correspondences between ontologies.

(P4) Focus more on Patterns than Models. Ontology models can cover a large scope, covering several requirements. In such cases, it can be hard to identify a model that meets a specific requirement desired for reuse. Moreover, when finding the model, it is necessary to identify its fragment that meets the desired requirement for reuse. Patterns are a better approach in these cases. They promote reuse since they modularize ontology models in a way that is easier to find and reuse.

4. GO-FOR Overview

Figure 1 shows an overview of GO-FOR. The basic elements of GO-FOR are goal-oriented ontology patterns (GOOPs), which is aligned to principle P4. A GOOP consists of an ontology fragment wrapped by a goal. In a GOOP, the goal establishes the scope addressed by the ontology fragment. Thus, GOOPs can be reused based on the goal to which they relate. GOOPs are stored in a goal-oriented ontology pattern repository (GOOPR). Inside the GOOPR, GOOPs relate to each other according to the relationships between their goals.

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 [21]). The use of goal models to define the ontology scope contributes to principle P2, since 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. Moreover, goal models help define the ontology purpose, which is also addressed in P2.

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. Next, we discuss aspects of the main elements of our framework, namely GOOPs and GOOPR.

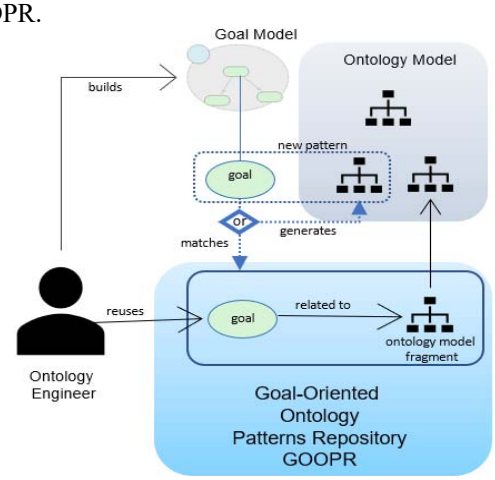

Figure 1 GO-FOR overview

4.1 Goal-Oriented Ontology Patterns (GOOP)

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).

A GOOP is also related to the actor who has the goal contained in the GOOP. Different actors may have the same goal and need different fragment models to achieve

it. For example, a doctor and a researcher may have both the goal "describe disease" and need different concepts to do so (e.g., a researcher may need more technical details about the disease). Thus, when searching for a GOOP to be reused, the ontology engineer should also consider the actor related to the GOOP in order to reuse the GOOP more suitable for the ontology being developed.

Ontology design patterns and ontology models usually have arbitrary names. Contrariwise, GOOPs are identified by their goals and following a standard terminology structure to name them (principle P1). To make things as simple as possible, we propose the use of a verb (bare infinitive) and a noun (or a noun phrase) to define them (e.g., define course program, describe product offering).

By analyzing several ontology model structures and investigating goals that they are able to achieve, we have noticed that goals are usually associated with verbs which are somehow similar. Since ontology structures are fragments of knowledge representation, they usually *classify, specify define or describe* (among others) something. Hence, GO-FOR suggests a set of verbs to be used to name goals. Although the use of a naming standard is proposed, this does not prevent arbitrary names from being given to goals. We propose this standard in order to soften the problem of completely arbitrary nomenclature discussed in P1 and, at the same time, simplify the search for GOOPs. Moreover, in our implementation (presented below) we apply semantic searching considering terms informed as parameters for the search and the terms used to name the goals.

When considering the goals, GOOPs can relate to one another by *a part of* relationship. A GOOP is part of another GOOP when the goal of the former is a decomposition of the goal of the latter. For example, if *g2* is a decomposition (i.e., subgoal) of *g1*, *goop1* contains *g1,* and *goop2* contains *g2*, then *goop2* is part of *goop1*. The part of relation is transitive, and the parts are not exclusive (i.e., a GOOP can be part of several GOOPs). If different GOOPs have common parts, it means that their ontology models overlap. The relationships between GOOPs promote principle P3, because once the relationships are identified, they can structure the overlaps and avoid redundancy.

4.2 Goal-Oriented Ontology Pattern Repository (GOOPR)

GOOPs are stored in the GOOPR, which serves as an abstraction layer for ontology development. This abstraction layer aims to support the reuse of ontology fragments already built, i.e., the GOOPs.

When developing a new ontology, the ontology engineer can search for GOOPs to be reused to address the scope of the new ontology. 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, aiming to identify matchings between them (i.e., to find GOOPs that meet the goals).

Suppose that an ontology engineer has built the goal model for an ontology about the Web Product Offering domain. In the goal model a Provider (actor) wants to "Describe Product Offering". This goal is decomposed into "Describe Product" and "Describe Offering Conditions". The latter, in turn, is decomposed into "Specify Payment Methods", "Specify Shipment Methods" and "Specify Coverage". In a bottom-up approach, the ontology engineer can search for GOOPs to achieve the ultimate goals (e.g., "Specify Payment Methods") and the achievement of the composed goals is obtained by integrating GOOPs related to the goals that form them. The ontology engineer can also adopt a top-down approach and search for GOOPs to achieve goals composed by others. She can explore goal decomposition to select the GOOPs for reuse. For example, if the ontology engineer searches for a GOOP related to "Describe Offering Conditions", she can compare the goal decomposition in the GOOP returned in the search with the goal decomposition in the goal model to verify coverage. If the returned GOOP is related to "Describe Offering Conditions", but it has only "Specify Payment Methods" and "Specify Shipment Methods" as subgoals, the ontology engineer can search for another GOOP with larger coverage, can search for another GOOP to complement the previous one by covering "Specify Coverage", or can decide to reuse the previous GOOP and develop the missing fragment.

By exploring the goals structure, the ontology engineer can also identify parts of the GOOPs that can be discarded. For example, if the GOOP returned to achieve "Describe Offering Conditions" also had "Describe Offering Validity" as subgoal and the ontology engineer was not interested in this goal, she would not reuse the part of the GOOP (which is a GOOP itself) related to this goal. Finally, when searching and selecting GOOPs, the ontology engineer must also take the actor related to the GOOP into account, because different actors can have the same goal but need different ontology models to achieve them.

4.3 GOOP-HUB: the GO-FOR Supporting Tool

Aiming to provide computational support to GO-FOR and promote its use, we have developed the GOOP-HUB (*https://github.com/nemo-ufes/goophub*), 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).

Considering that most of the ontology fragments available on the web use the Web Ontology Language (OWL), we have implemented the GOOP-HUB metamodel in this language. The main advantage of this choice is that someone can 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.

The model comprises elements of the OWL metamodel plus 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 can associate. 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.

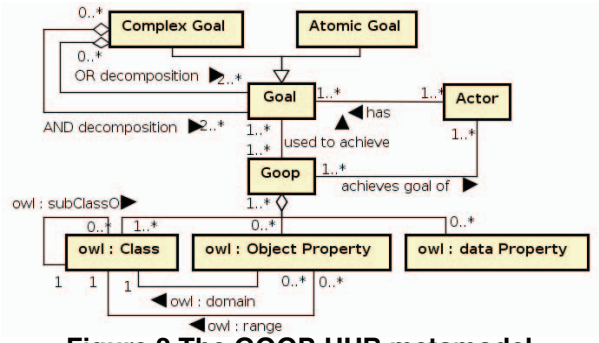

Figure 2 The GOOP-HUB metamodel

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. 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 (*https://www.stardog.com/*), 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 (*http://lucene.apache.org/*).

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 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 downloaded in OWL format.

5. Applying GO-FOR

GO-FOR has been applied in the context of a scientific project concerning the biggest Brazilian natural disaster, occurred on November 5th, 2015. The rupture of the Fundão tailings dam, located in the city of Mariana, in the state of Minas Gerais (MG) discharged 55–62 million m^3 of iron ore tailings slurry directly into the Doce River Basin, an important basin in the Southeast of Brazil. In response to the disaster, autonomous groups of researchers and government officials began to take actions to evaluate its consequences, producing a large volume of data in different areas of knowledge (hydrology, geochemistry, biology, among others). Thus, it becomes necessary to make an effort to make this data available and allow its use together. For that, this project has developed a water quality ontology to support data interoperability.

As prescribed by GO-FOR, the ontology scope was defined by means of goal models. An important actor of the domain of interest is the Researcher, who is responsible for collecting samples, describing information about environment conditions, obtain and analyze data, among others. Figure 3 shows a fragment of an iStar [18] goal model related to the Researcher, focusing on the goal "Describe Fish Specimen". Achieving this goal is necessary to describe the collection of a fish in the river to verify the water quality by analyzing the fish properties. In the model depicted in Figure 3, such main goal is AND-decomposed in three subgoals, namely: "Describe Date", "Classify Fish Specimen" and "Describe Locality". It means that in order to "Describe Fish Specimen", these three goals must be achieved. "Describe Locality",

in turn, has an OR decomposition with "Describe Geographic Point" and "Describe Place Name", meaning that the goal is achieved even if only one of its subgoals is achieved.

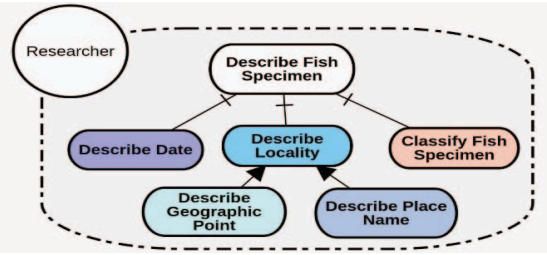

Figure 3 Fragment of the goal model for the water quality ontology

Once the goal model is defined, the next step was to search the GOOP-HUB looking for GOOPs to meet the established goals. The GOOP illustrated in Figure 4 was found as a candidate to meet the "Describe Date" goal. That GOOP was extracted from the Time Ontology (*https://www.w3.org/TR/owl-time/*) and it was stored in GOOP-HUB during the development of another ontology in which the fragment was used to achieve the goal "Describe Data". In other others, that GOOP was once produced *for* reuse and, here, it was reused in the context of development *with* reuse. In the figure, Temporal Position is the most generic concept that has properties to indicate the temporal system used. Time Position has properties to alternatively describe a temporal position using a number (i.e., a temporal coordinate) or a nominal value (e.g., geological time period, dynastic name, archeological era). General Date Time Description has a set of properties to specify the date and time using calendar elements and clock. Finally, Date Time Description transforms the temporal reference system to the Gregorian calendar. In Figure 4 and in the other models shown in this section, the concepts' attributes were omitted due to space limitation.

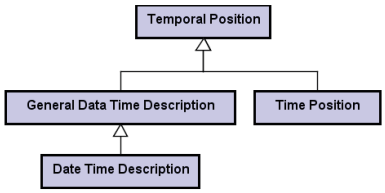

Figure 4 GOOP related to "Describe Date"

When looking for GOOPs to achieve the goals "Describe Fish Specimen" and "Describe Locality", no GOOP was found. Therefore, a decision had to be made: develop the fragments necessary to achieve these goals from scratch or reuse other ontologies. The decision was to reuse existing ontologies and extract new GOOPs from them.

In order to achieve "Describe Locality", a fragment was extracted from the Place Names Ontology (*http://www.linked.data.gov.au/def/placenames/*). The fragment is depicted in Figure 5. In the model, Geometry provides data properties to specify the latitude and longitude of a place. Place Name, in turn, describes a place using its name. Different colors are used in the figure to identify fragments related to each subgoal of "Describe Locality". The full fragment shown in the figure is necessary to achieve the "Describe Locality" goal. However, if the goal is "Describe Geographic Point", only the fragment containing concepts in dark blue color are necessary. Similarly, if the goal is "Describe Place Name", the necessary fragment is the one containing concepts in light blue color. This example illustrates the *part of* relation between GOOPs, resulting from the composition relation between goals. In the figure, there are three GOOPs. One related to "Describe Place Name", one related to "Describe Geographic Point" and one related to "Describe Locality", which is composed of the other two.

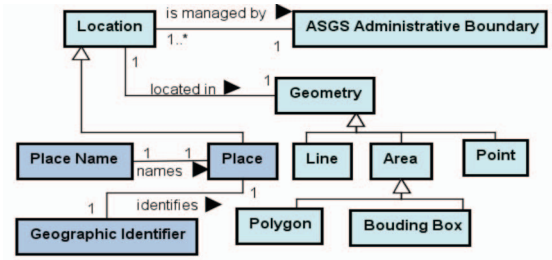

Figure 5 GOOP related to "Describe Locality"

To achieve "Classify Fish Specimen", a fragment was extracted from the ontology NCBITaxon (*http://purl.obolibrary.org/obo/ncbitaxon.owl*) and turned into a GOOP. Figure 6 shows a fragment of the final ontology. For the sake of simplification, we do not present all the concepts related to the goal model presented in Figure 3. For example, the top classes represent categories of NCBITaxon that must be instantiated to describe a fish specimen.

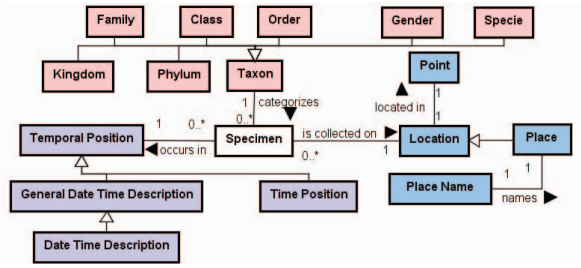

Figure 6 – Fragment (simplified) of the Water Quality Ontology – focus on Fish Specimen

In the model, we do not show the taxon instances and we show only two types of fish. Further, the concept color indicates the goal it relates to. Therefore, concepts plus the goal in the same color illustrates fragments of the GOOPs reused to build the ontology. The concept in white was added to the model to meet the ontology requirements. The GOOPs related to "Describe Fish Specimen" and "Describe Locality", which were created during the development of the ontology, were added to the

repository and made available for reuse. Finally, the ontology engineer can decide to incorporate the reused concepts directly or to create new concepts in their own schema and relate to the original ones through, e.g., owl:equivalentClass, rdfs:subClassOf, etc.

It is worth saying that the model shown in Figure 6 is just a small fragment of the water quality ontology that has been developed to support data interoperability in the cited project. The ontology has been used as a reference model to integrate data from different sources (spreadsheets, information systems, web sites, PDF files, etc.) as well as to support the development of a portal to enable data searching for users in a transparent way.

6. Related Works

The use of goal modeling in Ontology Engineering is recent. Therefore, we did not find any work exploring the use of such approach to support reuse. There are some works presenting proposals on the use and storage of ontology patterns or ontologies to promote reuse in ontology development. Here, we highlight three of them. In [27], reuse is focused on foundational and domain patterns. The former are patterns extracted from the foundations and rules of a foundational ontology and, when reused by analogy, their structure is reproduced in the ontology being developed. The latter are patterns that capture the core knowledge of a domain and, when reused by extension, the new ontology carries the concepts and relations from the pattern. In [8], Gangemi et al. use a Software Engineering approach to catalog content ontology design patterns. Each pattern is associated with a catalog entry including information such as name, competency questions and scenarios. The patterns are cataloged and stored within a repository. In [28], a framework for ontology reuse is proposed, but the stored and reused resources are ontologies, not ontology patterns.

There are some similarities and differences between these works and ours. Regarding similarities, we can highlight the fact that all proposals help define and store ontology patterns or ontologies and provide a string-based search mechanism to retrieve them. As for differences, we point out the way the design rationale is expressed and the basis for the searches.

In [27], the authors argue that when a foundational pattern is reused, its design rationale is carried to the domain pattern. Thus, when the domain pattern is reused, the foundational and the domain design rationales are carried to the new ontology. However, the design rationale is not made explicit in the patterns' definition, thus it is not clear how the design rationale is expressed to the ontology engineer. In [26], the design rationale comes from information associated with the patterns. However, it is not clear how reuse can be conducted based on this information. In [8], design rationale comes from domain experts that help the ontology engineers to understand the domain of interest. However, information about design rationale is not recorded neither used as a basis for the searches, which are made in repositories in general.

In GO-FOR, design rationale is expressed by means of goals that reveal the reasons why concepts, relations and constraints are necessary. The use of goals can help in the identification of suitable patterns, since the ontology engineer can search based on the goals to be achieved. Moreover, goals composition can be explored to help in the search and selection of patterns for reuse. Differently from GO-FOR, none of the aforementioned approaches use goals or design rationale information to aid the search for patterns or ontologies. Goal-based search potentially brings better results because goals are in a higher level than concepts. Hence, it is easier to talk with ontology engineers about their goals than about concepts in the domain of interest. In addition, none of the approaches stores design rationale information alongside the patters or ontologies. Thus, when the ontology engineer follows these approaches, she must derive herself the rationale behind the patterns or ontologies found.

7. Final Considerations

In this work, we advocate the use of goals as the central elements to define and search for ontology patterns. We propose GO-FOR, a goal-oriented and pattern-based framework for ontology reuse, aimed to aid in ontology reuse by providing a mechanism for ontology engineers to create and search for ontology patterns based on their goals.

The main contributions of the work addressed in this paper are: (i) the framework itself, which allows to connect ontology fragments with the goals they meet, giving rise to goal-oriented ontology patterns (GOOPs), store the GOOPs in a repository and retrieve them for reuse; and (ii) the GOOP-HUB, the tool developed to support GO-FOR use and promote ontology reuse. Once the GOOP-HUB is properly populated, it will increase reuse, generating a virtuous cycle of ontology development with reuse and for reuse.

GO-FOR has been used to develop an ontology to integrate water quality data. The results have shown that the use of GO-FOR is viable. The effort demanded to create GOOPs was justified by the benefits of reusing them, and goals have shown to be a good reference to guide the searches. However, new applications and studies are necessary. As future work, we intend to populate the repository with GOOPs and to carry out studies (case studies and experiments) to evaluate the effects of using GO-FOR in the ontology engineering process.

To evolve GO-FOR, we plan to extend it to use other intentional elements in addition to goals (e.g., tasks), allowing to define more detailed goal models and contribute to refine the search for suitable GOOPs.

As for GOOP-HUB, we intend to assign properties to GOOPs so that, when more than one GOOP is returned in a search, they can be ranked not only by coverage (how it currently works) but also by other attributes such as reuse rate. We also plan to include functionalities in the tool to aid in the integration of GOOPs, in order to generate new GOOPs and to help ontology engineers in the integration of GOOPs into their ontologies. Finally, we intend to integrate an ontology editor to GOOP-HUB, aiming to automatically generate code for ontology implementation from ontology models.

Acknowledgment

This research is funded by the Brazilian Research Funding Agency CNPq (Processes 407235/2017-5, 433844/2018-3), FAPES (Process 69382549/2014 and TO 616/2018), CAPES (Process 23038.028816/2016-41 and Finance Code 001).

References

[1] R. Bajcsy and R. McGeer, "Knowledge Engineering: Principles and Methods. Data Knowledge Engineering., 25: pp. 161-197, 1998"

[2] E. Davis, "The Naive Physics Perplex," *AI Mag.*,pp. : 51-51, 1998.

[3] G. H. Mealy, "Another look at data," in *Proc. of the, fall joint computer conference, AFIPS '67 (Fall)*, pp.: *14-16, 1967*.

[4] T. Heath and C. Bizer, "Linked Data: Evolving the Web into a Global Data Space," *Synth. Lect. Semant. Web Theory Technol.*,pp. 1-136, 2011.

[5] M. Poveda Villalon, M. C. Suárez-Figueroa, and A. Gómez-Pérez, "Reusing ontology design patterns in a context ontology network,", Proc. of 2nd WOP, pp. 35-52, 2010.

[6] M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, *Ontology engineering in a networked world*. Springer Science & Business Media, 2012.

[7] J. Park, S. Oh, and J. Ahn, "Ontology selection ranking model for knowledge reuse," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5133–5144, 2011..

[8] A. Gangemi and V. Presutti, "Ontology design patterns," in *Handbook on ontologies*, Springer, 2009: 221–243.

[9] A. P. J. Jarczyk, P. Loffler, and F. M. Shipmann, "Design rationale for software engineering: a survey," *Proc. Twenty-Fifth Hawaii Int. Conf. Syst. Sci.*, 1992.

[10] A. Van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," *Proc. Fifth IEEE Int. Symp. Requir. Eng.*, pp. 249–262, 2001.

[11] M. Katsumi and M. Grüninger, "What is ontology reuse?," in *Frontiers in Artificial Intelligence and Applications*, pp. 9-22, 2016.

[12] E. P. Bontas, M. Mochol, and R. Tolksdorf, "Case Studies on Ontology Reuse," in *Proceedings of the IKNOW05 International Conference on Knowledge Management (Vol. 74)*, 2005, pp. 345–353.

[13] R. A. Falbo, G. Guizzardi, A. Gangemi, and V. Presutti, "Ontology patterns: Clarifying concepts and terminology," in *Proc. of the 4th WOP*, pp. 14-26, 2013.

[14] F. Buschmann, K. Henney, and D. C. Schmidt, *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*. Wiley, 2007.

[15] V. Presutti, E. Daga, A. Gangemi, and E. Blomqvist, "eXtreme design with content ontology design patterns," in *Proc. Workshop on Ontology Patterns, pp. 83-97, Washington, DC, USA*, 2009.

[16] E. Blomqvist, A. Gangemi, and V. Presutti, "Experiments on pattern-based ontology design," *Proc. 5th Int. Conf. Knowl. capture,* pp. 41-48, 2009.

[17] R. A. Falbo, M. P. Barcellos, J. C. Nardi, and G. Guizzardi, "Organizing ontology design patterns as ontology pattern languages," vol. 7882 LNCS, pp. 61–75, 2013.

[18] X. Franch, L. López, C. Cares, and D. Colomer, "The i* Framework for Goal-Oriented Modeling," in *Domain-Specific Conceptual Modeling*, Springer, pp. 485–506, 2016.

[19] A. Van Lamsweerde, "Reasoning about alternative requirements options," *LNCS*, pp. 380-397, 2009.

[20] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proc. of the 18th RE2010*, pp. 115-124, 2010.

[21] P. C. B. Fernandes, R. S. S. Guizzardi, and G. Guizzardi, "Using goal modeling to capture competency questions in ontology-based systems," *J. Inf. Data Manag.*, vol. 2, no. 3, pp. 527, 2011.

[22] J. S. Salamon, C. C. Reginato, M. P. Barcellos, and R. S. S. Guizzardi, "Using goal modeling and ontoUML for reengineering the good relations ontology," in *Proceedings of IX Ontobras*, pp. 91-102, 2017.

[23] L. Liu and E. Yu, "Designing information systems in social context: A goal and scenario modelling approach," in *Information Systems*, vol. 29, no. 2, pp. 187–203, 2004.

[24] T. Gruber, "The role of common ontology in achieving sharable, reusable knowledge bases," *Princ. Knowl. Represent. Reason. Proc. Second Int. Conf.*, pp. 601–602, 1991.

[25] M. Hepp, "Good Relations: An Ontology for Describing Products and Services Offers on the Web", *Proceedings EKAW '0*8, pp. 329-346, 2008

[26] A. Gangemi and V. Presutti, "Ontology Design Patterns," *Handb. Ontol.*, pp. 221–243, 2009.

[27] F. B. Ruy, C. C. Reginato, V. A. Santos, R. A. Falbo, and G. Guizzardi, "Ontology Engineering by Combining Ontology Patterns," in *The 34rd International Conference, ER2015*, 173-186, 2015.

[28] E. G. Caldarola, A. Picariello, and A. M. Rinaldi, "An approach to ontology integration for ontology reuse in knowledge based digital ecosystems," in *Proc. of the 7th MEDES*, pp. 1–8, 2015.