Applied Ontology 17 (2022) 365–399
DOI 10.3233/AO-220269

A Goal-Oriented Framework for Ontology Reuse

Cássio C. Reginato[a], Jordana S. Salamon[a,*], Gabriel G. Nogueira[a], Monalessa P. Barcellos[a], Vítor E. Silva Souza[a], Maxwell E. Monteiro[a], Renata Guizzardi[a]
[a] *Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espiríto Santo – Vitória – ES – Brazil*
E-mail: 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

Abstract. Ontologies have been successfully used to assign semantics in the Semantic Web context, to support integration of data from different systems or different sources, and to enable reasoning. 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 present GO-FOR, a framework in which goals are the central elements to promote ontology reuse. GO-FOR comprises a conceptual architecture, a goal-oriented ontology development process and a supporting tool. In GO-FOR we introduce Goal-Oriented Ontology Patterns (GOOPs) 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 patient examination data are also shown in this paper.

Keywords. Ontology, Reuse, Goal Modeling, Pattern, Ontology Integration

Accepted by: Aldo Gangemi

1. Introduction

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 (Davis, 1998; Mealy, 1967). 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 (Heath & Bizer, 2011).

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 & Liebig, 2009). Some of the main reasons for that are: (i) the ontology engineer must have a consistent and mature view

* Corresponding author. E-mail: jssalamon@inf.ufes.br. Address: Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espiríto Santo – Fernando Ferrari Avenue, 514, Goiabeiras - Vitória – ES – Brazil, ZIP Code: 29075-910. Tel. No.: +55 27 4009-2137

of the domain being represented; (ii) ontology representation languages (e.g. OWL, 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 still open to interpretation and face problems such as requirements and scope definition.

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 (Gangemi & Presutti, 2009). 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.

One of the challenges of ontology reuse is the obscurity of the design rationale of the available ontologies (Gangemi & Presutti, 2009). Design rationale concerns the reasons for decisions made during a design process (Jarczyk et al., 1992). 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 (Van Lamsweerde, 2001). GORE has been used to provide design rationale in Software Engineering, explaining from where a requirement came and which stakeholders' goals it meets (Van Lamsweerde, 2001). In Ontology Engineering, we argue that GORE helps explain the rationale behind ontology selection, because motivational elements (i.e., goals) reveal the reasons behind the development of an ontology (or ontology pattern), besides providing a notion of the kind of knowledge that is represented in the ontology.

The work presented in this paper is part of a larger research project whose main research question is how the use of GORE can improve the understanding of ontology scope and design rationale and contribute to ontology reuse. The motivation behind the project came from practical experiences in our research group and from literature. In (Salamon et al., 2018), we investigated the literature about ontology reuse (more specifically regarding ontology integration and merging) and we noticed that the lack of explicit design rationale and of a concern with ontology integration in the ontology development process as a whole has hampered reuse. Note that what makes GORE special is the possibility to use goals to make explicit the 'whys' behind requirements. We were inspired by the approach presented in (Fernandes et al., 2011), which adopts competency questions as a strategy to identify ontology's scope and requirements, and it profits from GORE potential to make clear the design rationale (i.e., the motivation behind CQs). In this work, we benefit from this clarification in a framework to support ontology reuse.

This paper presents 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 design 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. GO-FOR aids in the creation, storage, and reuse of GOOPs, promoting ontology development with and for reuse. A partial view of GO-FOR was introduced in Reginato et al. (2019), focusing on its conceptual architecture. In this paper, we extend (Reginato et al., 2019) by providing a more comprehensive view of GO-FOR and detailing all its components: a conceptual architecture, an ontology development process and a

supporting tool. We also present a new application of GO-FOR in a real case scenario in the health care (particularly, laboratory test) domain.

The main contributions of this paper are: (i) the framework itself (GO-FOR), which allows connecting ontology fragments with the goals they meet (giving rise to GOOPs), storing the GOOPs in a repository and retrieving them for reuse; (ii) the goal-oriented and pattern-based process for ontology development reusing GOOPs and other ontology fragments; and (iii) GOOP-Hub, the tool developed to aid GO-FOR use and support ontology reuse. This work contributes to the state-of-the-art by exploring the use of GORE in the context of ontology reuse and to the state-of-the-practice by providing a tool to store and reuse ontology fragments (GOOPs) based on the goals they achieve.

This paper is organized as follows: Section 2 provides the background for the paper; Section 3 presents the principles behind GO-FOR; Section 4 describes the GO-FOR conceptual architecture; Section 5 presents the GO-FOR process; Section 6 concerns GOOP-Hub, the GO-FOR supporting tool; Section 7 addresses GO-FOR use and reports the application of GO-FOR to build an ontology to integrate patient examination data; Section 8 discusses related works; and Section 9 presents our final considerations.

2. Background

2.1. Ontologies

"An ontology is a formal, explicit specification of a shared conceptualization" (Struder et al., 1998). Here, "conceptualization" refers to an abstract model of some phenomenon in the real world that identifies the relevant concepts of this phenomenon; "explicit" means that the types of concepts used and the constraints imposed on their use are explicitly defined; "formal" refers to the fact that an ontology should be interpretable by machines; and "shared" reflects that ontologies must capture consensual based knowledge accepted by a community" (Struder et al., 1998).

There are several classifications for ontologies. An important distinction differentiates ontologies as conceptual models, called reference ontologies, from ontologies as computational artifacts, called operational ontologies. In general, we advocate for an approach that considers both ontology types (Guizzardi, 2007). First, a reference ontology is constructed with the goal of making the best possible description of the domain in reality, representing a (conceptual) model of consensus within a community, regardless of its computational properties. Once users have agreed on a common conceptualization, operational versions (machine-readable ontologies) of a reference ontology may be implemented. Contrary to reference ontologies, operational ontologies are designed with the focus on guaranteeing desirable computational properties.

2.2. 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, we find no formal definition of ontology reuse widely accepted within the community (Katsumi & 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 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 (Katsumi & Grüninger, 2016).

According to Carriero et al. (2020), nowadays there are several approaches to ontology reuse, with different motivations or implementations. For instance, direct reuse of existing standard or popular ontologies is typically encouraged by institutions and community consortia. Research communities and communities of practice also suggest indirect approaches (i.e., by designing ontologies tailored to a use case and aligning them to standard or popular ones when appropriate), or hybrid approaches.

An important and challenging task of ontology reuse is to select the ontologies to be reused. It is a highly subjective task, often manually performed by experienced ontology engineers, who select ontologies that intuitively fit for the purpose. Different motivations can guide ontology selection, such as (i) reuse by standardization, when the reused ontology is issued by authoritative organizations, like ISO, W3 Consortium, and professional or community consortia; (ii) reuse by popularity, when the reused ontology is selected for being popular (i.e., very reused); and (iii) reuse by cognitive analysis, when the ontology engineer defines the requirements of the new ontology to be developed and considers them to decide what to reuse, e.g., existing design components, such as foundational ontologies or ontology patterns (Carriero et al., 2020).

Ontology patterns (OPs) (also known as ontology design patterns (ODPs)) are an emerging approach that favors 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 Buchmann et al. (2007), 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 (Presutti et al., 2009). Experiments, such as the ones conducted by Blomqvist et al. (2009), 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 (Buchmann et al., 2007). Thus, when applying a pattern, it is important to understand and take its relationships into account (Falbo et al., 2013).

In the literature, 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: Ruy et al. (2015), Gangemi and Presutti (2009) and Caldarola and Rinaldi (2016). According to Ruy et al. (2015), 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 the work presented by Gangemi and Presutti (2009), the researchers 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 the work presented by Caldarola and Rinaldi (2016), a framework for ontology reuse is proposed, but the stored and reused resources are ontologies, not ontology patterns.

Reuse is not an isolated process. It must be performed to support ontology development. Thus, it is necessary to consider reuse in the broader context of ontology development. However, results of a mapping study of the literature (Salamon et al., 2018) showed that, in the context of ontology integration and merging, most works focus on integrating ontologies already selected to be reused (i.e., given two ontologies, how to integrate them), i.e., they do not consider the ontology development process as a whole. Among the works that address reuse in a broader context, we can cite Blomqvist et al. (2016), which addresses reuse in the context of a collaborative, incremental, and iterative method for pattern-

based ontology design, extending the work presented by Gangemi and Presutti (2009). Leung et al. (2011), in turn, define a set of guidelines for selecting ontology reuse methods and proposes an ontology development methodology that integrates ontology reuse methods and a system to support ontology integration. According to Cuenca et al. (2017), reuse is considered to aid ontology development within an ontology network development process. Last, Pinto and Martins (2001) presents a general methodology to guide ontology integration, with activities describing general steps that can be used alongside different ontology building methodologies.

In the last years, some ontology engineering methods have been derived from modern software engineering techniques. They take an agile perspective into account aiming at alleviating ontology development complexity by adopting simple tools, methodological guidance, and reinforcing reuse. Some of these works, are Upon Lite (Nicola & Missikoff, 2016), SAMOD (Peroni, 2017), Test-Driven Development of Ontologies (Keet and Ławrynowicz, 2016), FORZA (Keet et al., 2013) and the work by Blomqvist et al. (2016), cited before.

Upon Lite (Nicola & Missikoff, 2016) proposes an agile method aimed at domain experts with generic steps for ontology engineering. Reuse is mainly addressed in the first steps, where experts must create a domain-specific terminology and enrich it by associating a textual description based on knowledge resources reuse (e.g., domain-related standards, reference textual documents, authoritative resources, and ontologies that, if available, must be linked to the already defined terminology). However, how to perform this link between the proposed terminology and the ontologies is not addressed by the methodology.

SAMOD (Peroni, 2017) prescribes an iterative process with generic steps and principles for ontology development. Some of these principles regard reuse through patterns and knowledge (concepts and relations) from other ontology models. The method does not address how these patterns or concepts will be integrated with the new ontology model.

TDDO (Keet and Ławrynowicz, 2016) is an axiom-driven ontology development method in which users have three scenarios: to develop competency questions and derive axioms from them; to directly input the axioms they know they need to add; and to use generic templates (with the same notion as ODPs) to be instantiated with relevant domain entities. The work presents a possible ontology lifecycle with a core idea that tests will be run in the ontology to identify if the CQs are already being answered. When not, the ontology will be updated with new axioms and the tests will be run again to check if all needs are covered, culminating in a TDD cycle.

FORZA (Keet et al., 2013) is actually the instantiation of a method called GENERATOR for a specific scenario involving the DOLCE foundational ontology. This method describes three general steps that are applied to two use case scenarios, one with reuse of foundational ontologies, and one without it. In both cases there is already a domain ontology being built (with concepts and relations already being defined) and relational ontologies (ontologies with defined relations and axioms that can be reused), so the reuse comes from these pre-existing ontologies and from the foundational ontology.

All the aforementioned methods advocate reuse in some degree, being of concepts, relations, axioms, or patterns. However, none of them explain in detail how to select and integrate pre-existing knowledge into the ontology being built, leaving it to the ontology engineer or domain expert. In general, these methods propose generic processes, not completely addressing some aspects from more traditional ontology engineering methods (e.g., requirements elicitation, design, implementation) or ontology integration activities. Reuse is also addressed in a generic way. On one hand, having a more generic ontology engineering method can be beneficial, bringing more agility and dynamicity to the process. On the other hand, this can leave some particularly important analysis or decisions to the ontology engineer or domain expert.

2.3. Goal-Oriented Requirement Engineering

Goal-Oriented Requirement Engineering (GORE) is the approach that we propose to support ontology reuse. A goal is a condition or state of affairs in the world that the actor (i.e., the agent who has the goal) would like to achieve. It is expressed as an assertion in the representation language. How the goal is to be achieved is not specified, allowing alternatives to be considered (Yu, 1995). 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 (Franch et al., 2016), KAOS (Van Lamsweerde, 2009) and Techne (Jureta et al., 2010), among others. 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 (Liu and Yu, 2004).

GORE has been successfully applied in Requirements Engineering (Van Lamsweerde, 2001) and has also been used to enrich the requirements analysis phase of the ontology engineering process, as in Fernandes et al. (2011) and Salamon et al. (2017).

GORE brings innovation to such process, by helping the ontology engineer understand the domain of interest in the perspective of the involved actors. In summary, the ontology engineer must identify the actors in the domain of interest and develop a goal model for each of them. Additionally, goal models allow the refinement of goals into tasks that can accomplish them. Each goal model represents the actor goals and tasks to be addressed by the ontology, providing a comprehensive view of the ontology scope. Ultimately, the process of designing and analyzing goal models may lead to the derivation of competency questions (Fernandes et al., 2011), which the ontology must be able to answer. These questions are well-known in the Ontology Engineering community, being generally used to delimit the scope and as requirements of an ontology under development. The difference here is hence given by the use of goal analysis, which allows the ontology engineer to make clear the motivations behind competency questions. In fact, goal analysis may also support the creation of such questions, in a way that first, stakeholders' goals are identified and refined, later leading to the conception of a consistent set of competency questions.

3. GO-FOR Principles

Building an ontology through reuse depends on finding suitable ontologies for being reused (Park et al., 2011). 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 being 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 ontology reuse. GO-FOR stands for Goal-Oriented Framework for Ontology Reuse.

GO-FOR is based on four principles related to a subset of ontology design recurrent issues pointed out in Gruber (1991), namely: (I1) Which pieces of information about terms are critical for supporting shareability (e.g., name, textual definition, type)? (I2) How to describe the purposes of a particular ontology? (I3) How to capture and use design rationale? (I4) 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 I1. Keywords and other metadata are often used to enable semantic search. However, it is difficult to say which metadata about the 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, occurs when ontology developers give arbitrary and inexpressive names for ontology structures. Although recently this aspect has been addressed by the community, there are still several models and vocabularies with inexpressive names. For instance, if an ontology engineer searches for an ontology covering the e-commerce domain using the keyword "e-commerce" to find ontologies addressing this subject, he/she may expect to find ontology models covering aspects such as transactions, authentication, shopping and so on. However, such search could return, among others, the Good Relations ontology (Hepp, 2008), 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 (Salamon et al., 2017). 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 (Gangemi & Presutti, 2009), which is directly related to issue I3. 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 for ontology reuse. Moreover, making the design rationale explicit also contributes to solve issue I2, 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 I4, 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.

GO-FOR takes the four aforementioned principles into account and 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) a tool that supports GO-FOR use. In the following sections we describe the GO-FOR components.

4. GO-FOR Conceptual Architecture

The basic elements of GO-FOR are goal-oriented ontology patterns (GOOPs), which are aligned to principle P4. 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. Figure 1 shows an overview of GO-FOR conceptual 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)). 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.

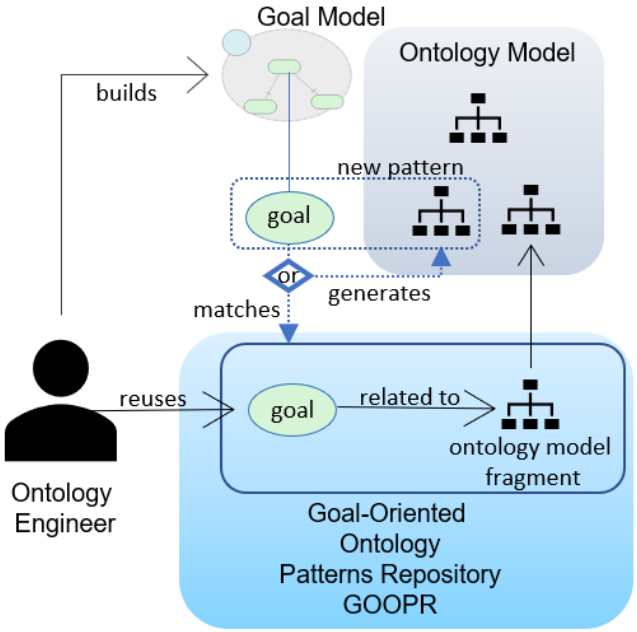

Figure 1. GO-FOR architecture

The ontology engineer is who builds the goal model that guides the ontology development and, thus, he/she is responsible for defining the ontology goals. Although a certain degree of understanding of the goals that the ontology should achieve is required, the use of goals as a way to represent requirements has shown that they tend to be closer to domain experts, promoting better understanding of their needs and underlying motivation (Yu et al., 2011). This is due to the fact that goals not only focus on what should be done but also on why, justifying and explaining the presence of requirement components which are not necessarily comprehensible to clients (Aljahdali et al., 2011). Hence, it is probably easier to start by identifying the goals that the ontology needs to achieve than its detailed requirements (e.g., in the form of competency questions, which can be derived from the goals) (Fernandes et al., 2011).

Next, we discuss aspects of GO-FOR conceptual architecture elements, namely GOOPs and GOOPR.

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 can 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). In the goal modeling literature, goals are usually expressed using passive voice, but there is no defined rule about that. Thus, to make things as simple as possible, we propose to name GOOPs by using a verb (bare infinitive) and a noun (or a noun phrase), e.g., "define course program", "describe product offering". Concerning the verb, by analyzing several ontology model fragments and investigating goals that they are able to achieve, we noticed that the goals can usually be associated with three main types of actions, namely: classification, identification and description. Thus, we suggest a set of verbs to be used to name a GOOP according to the type of action to which its goal refers. 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. The word classification is commonly used with this meaning whether in conceptual modeling or ontology engineering (Partridge et al., 2018). Thus, for model fragments that are used to classify entities we indicate the use of the verb "classify" to label the pattern. 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). Here, the verbs to be used are "identify" or "determine". These verbs were suggested based on the concept of identity discussed in (Guizzardi et al., 2021). Finally, description occurs when a model 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. For description, we suggest using "describe", "characterize", "define", "specify" or "register". The verbs proposed here are the ones we have identified so far and, thus, is not an exhaustive list. The set of verbs can be extended in the future by including new synonyms or even new verbs to represent other actions not addressed so far. 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.

We propose this standard in order to soften the problem of completely arbitrary terminology discussed in P1 and, at the same time, simplify the search for GOOPs. That is, the search for GOOPs becomes more accurate if a controlled vocabulary is used to name them. Moreover, in the supporting tool (presented in

Section 6) 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.

In order to increase reusability, a GOOP has an associated documentation, describing its concepts and other relevant information to support understanding the ontology fragment and, thus, contribute to its reuse.

As an example, Fig. 2 shows a fragment of a goal model and the GOOP created from the Place Names Ontology[1] to achieve the goal *Describe Locality*. The GOOP was created in the context of a Water Quality Ontology, developed by using GO-FOR (Reginato et al., 2019). Note that the goal *Describe Locality* has two subgoals: *Describe Geographic Point* and *Describe Place Name*. Each of them is related to an ontology fragment (indicated by the same color used to represent the goal). The integrated two fragments related to the goal *Describe Locality* form the GOOP *Describe Locality*.

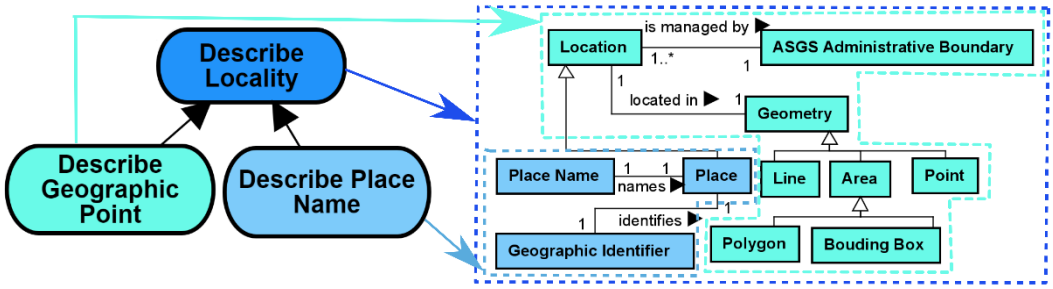

Figure 2. GOOP related to *Describe Locality.*

4.2. Goal-Oriented Ontology Pattern Repository (GOOPR)

GOOPs are stored in a GOOPR, that aims at supporting 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. 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).

Suppose that an ontology engineer has built the goal model shown in Fig. 3[2] for an ontology about the Web Product Offering domain. In the goal model a Provider (an actor) wants to *Describe Web 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*.

[1] http://www.linked.data.gov.au/def/placenames/
[2] The goal model was produced using iStar (Franch et al., 2016) using the piStar tool (Pimentel & Castro, 2018).

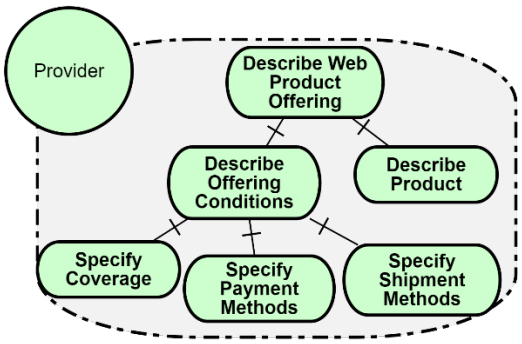

Figure 3. Goal model example.

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. He/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*, he/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, he/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, as we said before, different actors can have the same goal but need different ontology models to achieve them.

5. GO-FOR Ontology Development Process

The GO-FOR ontology development process consists of a set of steps to guide ontology development using the GO-FOR conceptual architecture. Therefore, it is a goal-oriented and pattern-oriented ontology development process with and for reuse. It addresses the ontology development process as a whole (from requirements elicitation to test) and covers the development of reference and operational ontologies. Here, we focus on the process steps referring to the development of reference ontologies because the following steps (design, implementation and test), necessary to produce an operational version of the reference ontology, are very similar to the corresponding steps in other ontology development methods (e.g., Falbo (2014)).

Figure 4 depicts the GO-FOR ontology development process, which consists of six phases: *Ontology Requirements Elicitation, Selection of GOOPs for Reuse, Selection of Ontologies for Reuse*, *Ontology Building*, *Ontology Evaluation,* and *Extraction of GOOPs*. Each phase of the process is described in the

following to indicate the flow between activities. In the figure, the initial node (solid circle) indicates the beginning of the process. Control flows (arrows) represent the sequence of activities. Diamonds indicate decision points, where a choice between possible flows must be made based on a condition. Rectangles with rounded edges represent activities. When they have the ⊞ symbol in the lower right corner, they represent activities that are broken down into others. Rectangles with straight edges represent artifacts, which can be inputs (when they are entering an activity) or outputs (when they are leaving an activity). The end node (non-solid circle) indicates the end of the process.

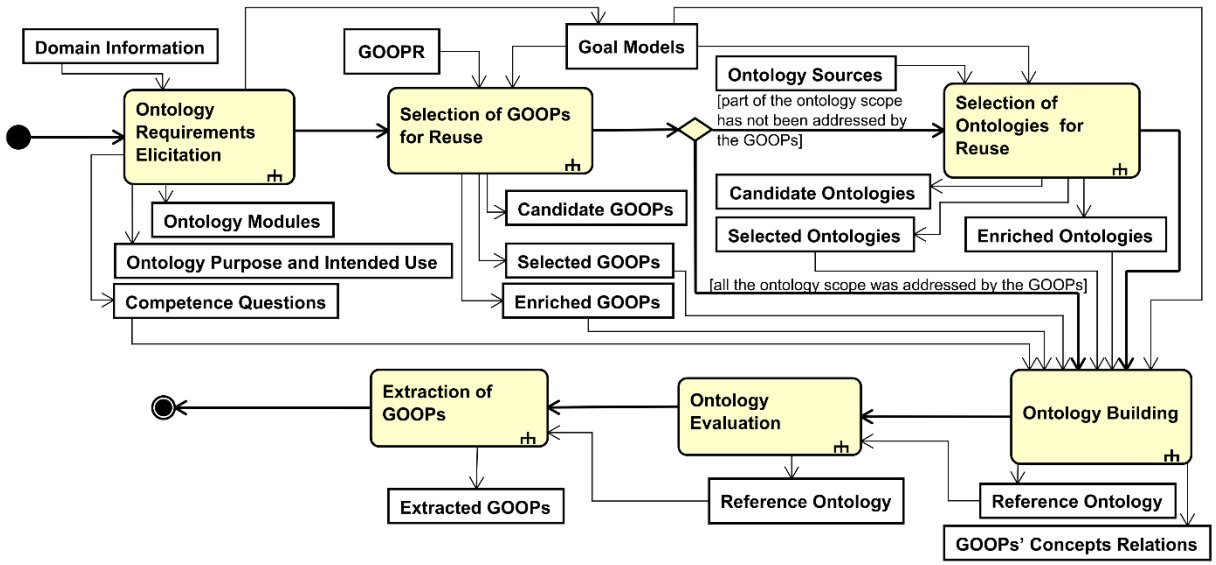

Figure 4. GO-FOR process

5.1. Ontology Requirements Elicitation

The first step to develop an ontology consists of establishing its scope and capturing its requirements. A common way to define the ontology's scope is by describing its requirements as competency questions that define which questions the ontology is supposed to answer, by navigating through its concepts and relations. Competency questions can thus be used as the basis for scope evaluation (Falbo, 2014). However, although competency questions are effective to represent ontology's scope, they are not enough to provide the design rationale behind the ontology (i.e., reason and motivation to develop an ontology). In summary, it is hard to grasp from where the competency questions come. In GO-FOR, GORE helps in this matter. Goals provide a richer context for understanding and interpreting requirements, because they relate at a higher level to the business or application domain (Yu et al., 2011). Thus, it is easier and more accessible for stakeholders to address goals rather than competency questions. *Ontology Requirements Elicitation* consists of three activities:

(i) Goal Modeling: The ontology engineer must define the ontology scope by means of goal models. Thus, he/she must identify the main actors in the domain of interest and develop a goal model for each of them. Each goal model represents the actor goals to be addressed by the ontology, providing a comprehensive view of the ontology scope. For example, in Section 4.2, Fig. 3 illustrates the goal model related to the described Web Product Offering domain. Currently, the goals are expressed in natural language by humans, taking into account the terminology suggested in Section 4.1. Goals can be defined

at several degrees of granularity, from broader to narrower goals. Broader goals usually hide narrower goals, so they can be decomposed into these more specific goals, forming a goal tree where goals are connected to each other by AND or OR decomposition.

In order to specify the ontology requirements, the goal models must be refined until they provide enough understanding of the ontology scope by the ontology engineer. Using a goal model to represent ontology scope, we can make explicit the rationale behind the ontology development. Once the ontology engineer has established the ontology scope, he/she must define the ontology purpose and intended use.

(ii) Competency Questions Elicitation: To detail the ontology's scope, the ontology engineer must derive competency questions from the goal models, and such questions will be further used as a basis to aid the development of the ontology's conceptual model. Considering the example of Fig. 4, we can take the goal *Specify Shipment Methods* and derive the following CQs: *What are the shipment methods? What is the price of a shipment method?.* These competency questions are then used to indicate concepts that must be addressed by the ontology (e.g., *Shipment Method* and *Shipment Method Price*). Thus, the goals indicate the motivation for the ontology engineer to include those concepts in the ontology, making the design rationale behind the ontology clearer.

(iii) Ontology Modularization: By analyzing the produced goal models and the competency questions, the ontology engineer has an idea of the size of the domain (or its portion) the ontology deals with. If the domain of interest is large or complex, he/she should modularize it considering independence, cohesion and size criteria (Falbo, 2014). In addition, goal models can help, enabling to modularize the ontology based on actors (e.g., one module for each actor of the domain), groups of actors (e.g., one module for group of actors that have dependencies among them), goals (e.g., one module for each main goal) or groups of goals (e.g., one module for a given goal decomposition). Modularization can be reviewed or refined in the Ontology Building phase.

5.2. Selection of GOOPs for Reuse

Once the ontology scope is established, the ontology engineer can search the GOOPR aiming to select GOOPs to be reused. This phase comprises three activities:

(i) Identification of Candidate GOOPs for Reuse: Based on the ontology scope, the ontology engineer must search for GOOPs to cover it. The search for GOOPs is based on the goals of the ontology being developed and retrieves GOOPs that meet such goals.

When searching for GOOPs based on the goals of the ontology being developed, the ontology engineer can explore the goal model structure to analyze the candidate GOOPs. That is, considering goals decomposition, he/she can analyze the coverage degree of a candidate GOOP in relation to the subgoals of the goal model. If there are subgoals not covered, the ontology engineer can decide to search for other candidate GOOPs to meet them or to reuse the GOOP and add new concepts and relationships in the ontology model. The competency questions can also help to evaluate the coverage degree of GOOPs by indicating information the GOOP must provide to properly answer the questions and cover the ontology scope.

(ii) Selection of GOOPs for Reuse: The ontology engineer must select among the candidate GOOPs the ones to be reused. The GOOPs that meet the reuse needs, that is, those that best suit the scope to be addressed and are feasible to be integrated to the ontology being developed, should be selected. Criteria such as completeness, understandability, consistency and availability (ISO/IEC 25012, 2008) should be considered.

(iii) Enrichment of GOOPs: The ontology engineer must evaluate if the selected GOOPs will be reused as they are or if reengineering is necessary. If the GOOPs are not in the desired degree of formalism or do not meet the same ontological commitments expected in the ontology being developed, an ontological analysis should be carried out, so that the ontology models to be integrated are anchored in the same

foundational concepts. Reengineering the GOOPs at the light of the same foundational ontology contributes to a proper integration in the next phase.

5.3. Selection of Ontologies for Reuse

If the GOOPs selected in the previous phase are not enough to cover the ontology scope, the ontology engineer can search (outside the GOOPR) for ontologies (or ontology fragments) to be reused. This phase comprises three activities analogous to the ones of the previous phase: (i) Identification of Candidate Ontologies for Reuse, when the ontology engineer identifies ontologies (or ontology fragments) that can be able to meet the goals not covered (or partially covered) by the selected GOOPs; (ii) Selection of Ontologies for Reuse, when the ontology engineer selects among the candidate ontologies (or ontology fragments) the ones to be reused; and (iii) Enrichment of Ontologies for Reuse, when the selected ontologies (or ontology fragments) are reengineered, if necessary.

5.4. Ontology Building

This phase aims to create the ontology model that meets the defined requirements and goals. In other words, in this phase the ontology conceptual model is built by integrating selected GOOPs and adding other necessary concepts and relations. It comprises three activities:

(i) Identification of Relations between GOOPs: The GOOPs selected for reuse may not be related to each other in the GOOPR and, still, relate to each other in some way. For example, a concept (e.g., Department) from a GOOP can be part of a concept (e.g., University) from another GOOP. Thus, the semantic relations between the ontology model fragments must be identified. Examples of possible relations between concepts from different GOOPs are (Ruy, 2017; Safyan et al., 2008): equivalent, part of, intersection, specialization of, generalization of, and acts as. In order to properly establish the relations, the ontology engineer should analyze the concepts description contained in the GOOP documentation.

(ii) Application of Integration Operations: Consists of performing operations to integrate the GOOPs considering the relations identified in the previous activity. Examples of integration operations are: unification, specialization, generalization, aggregation/composition, intersection and role representation. The operation to be applied depends on the semantic relation between the involved elements. For example, when two concepts from two different GOOPs are equivalent, they must be unified, i.e., represented only once in the ontology model. If a concept is part of another concept, both concepts must be represented in the ontology model and related to each other by a part-whole relation (aggregation or composition). If a concept represents a role that can be played by another concept, the concept that represents the role must be related to the other by a specialization relation, meaning that the specialized concept represents a role of the general concept.

(iii) *Development of the Reference Ontology:* The ontology engineer must develop the ontology model by integrating GOOPs (see previous activity) and also adding new concepts, properties, relationships, and axioms so that the resulting ontology model meets the established requirements and goals. The result is the Reference Ontology, which comprises the conceptual model containing the ontology concepts and relations, plus the concept's definitions and the axioms necessary to specify the constraints.

5.5. Ontology Evaluation

This phase aims to ensure the quality of the produced ontology. Therefore, the ontology is evaluated, and the necessary improvements or corrections are identified. In case adjustments are needed, the ontology engineer must return to the activity where adjustments can be made. For example, if axioms

must be reviewed, the ontology engineer must return to the activity *Development of the Reference Ontology*. Ontology evaluation involves two activities:

(i) Structural Evaluation: The ontology engineer must evaluate the ontology aiming at ensuring that there is no unwanted redundancy in the structuring of knowledge or inconsistencies in the model (Leung et al., 2011). Searching for anti-patterns is one way to find inconsistencies in the ontology structure (Guizzardi, 2014).

(ii) Semantic Evaluation: Focuses on evaluating the ontology in terms of its adequacy to the defined requirements and the users' satisfaction. It includes Verification and Validation. In Verification, the ontology engineer must evaluate if the ontology elements (concepts, relations, properties and axioms) are the ones necessary and sufficient to answer the competency questions and achieve the goals defined during *Ontology Requirements Elicitation* (Falbo, 2014). Concerning goals achievement, in cases where a goal is decomposed into others, by achieving the more specific goals (considering AND/OR relations among them), the more generic goal is also achieved. In Validation, the ontology engineer aims to ensure that the ontology fulfills its specific purpose (Falbo, 2014). Ontology users and domain experts must be involved in ontology validation. Ontology users must evaluate if the ontology is suitable for its intended uses. Domain experts should assess whether the conceptualization provided by the ontology reflects the portion of the real world represented in it. Validation can also be accomplished by instantiating the ontology concepts using real-world elements.

5.6. Extraction of GOOPs

The first five phases of the GO-FOR ontology development process address development *with* reuse, since they consider the reuse of existing GOOPs and ontologies to develop new ontologies. This last phase of the process aims at ensuring ontology development *for* reuse, generating new GOOPs and making them available at the repository (GOOPR). It involves two activities:

(i) GOOPs Identification: The ontology model resulting from the previous phase satisfies the goals established in the ontology goal models. In this activity, the ontology engineer must identify, from the ontology model, ontology fragments related to the goals represented in the goal models. Each fragment able to meet a goal (except the model fragments referring to reused GOOPs) should give rise to a new GOOP. Composition relation between GOOPs due to goals decomposition must be identified. For each identified GOOP, the ontology engineer must record its conceptual model, documentation (e.g., concepts description, axioms) and associated goal.

(ii) Storage of GOOPs: At last, the ontology engineer must store the created GOOPs in the GOOPR, making them available for future reuse.

The GOOPs created in this phase are later (during the implementation phase of the process – not addressed here) implemented in OWL and their operational versions are also made available in the GOOPR.

In summary, by performing GO-FOR ontology development process, a virtuous cycle of ontology development for reuse and with reuse emerges: the extraction of GOOPs from developed ontologies enables to populate the repository (GOOPR) promoting development *for* reuse. On the other hand, goal-oriented search to select GOOPs for reuse supports the development *with* reuse. Figure 5 illustrates the virtuous cycle of ontology development for reuse and with reuse that emerges from GO-FOR use.

When developing ontologies using the GO-FOR ontology development process, ontology engineers relate ontology fragments to goals and store them as GOOPs in the repository (GOOPR). In Fig. 5, ontology fragments G1, G2 and G3 associated to the respective goals represent GOOPs extracted from the Ontology A and its goal model. As it can be noticed, G1 is related to Goal 1, G2 is related to Goal 1.2 and G3 is related to Goal 1.1. Goals 1.1 and 1.2 are subgoals of Goal 1. Thus, GOOPs G2 and G3 are part of G1. Once GOOPs are available in the GOOPR, they can be retrieved based on their goals and the

ontology fragments can be integrated into ontologies being developed. In the figure, the goal model of the ontology being developed (Goal Model B) contains the Goal 1.2. Thus, the Ontology Engineer B retrieves the GOOP related to that goal and reuses the ontology fragment G2 by integrating it into the Ontology B.

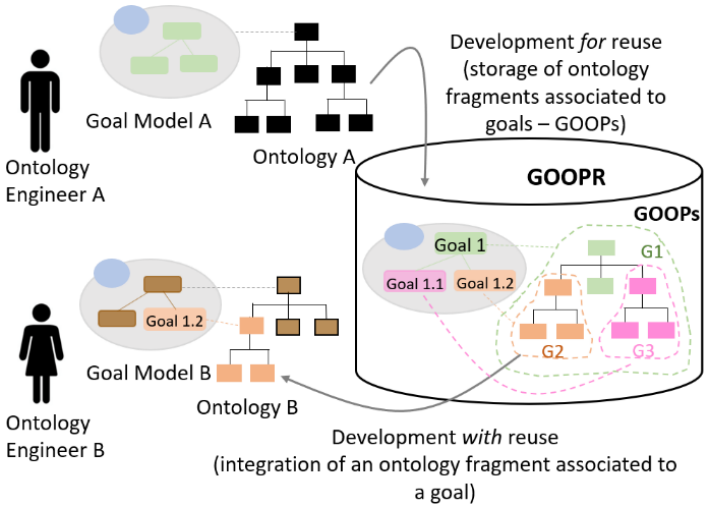

Figure 5. GO-FOR supporting development for and with reuse

6. 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[3], 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 it is possible to make SPARQL queries using the GOOP-Hub metamodel structure as part of the query. In this way, we can take advantage of the language to make inferences based on the meta-model and provide better search results. 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 6 presents the core concepts of the GOOP-Hub metamodel.

[3] GOOP-Hub code is available at https://github.com/nemo-ufes/goophub. The tool can be accessed at http://dev.nemo.inf.ufes.br/goophub/. A short user guide is available at https://nemo.inf.ufes.br/wp-content/uploads/GOFOR/GoopHub_User_Guide_v1.0.pdf.

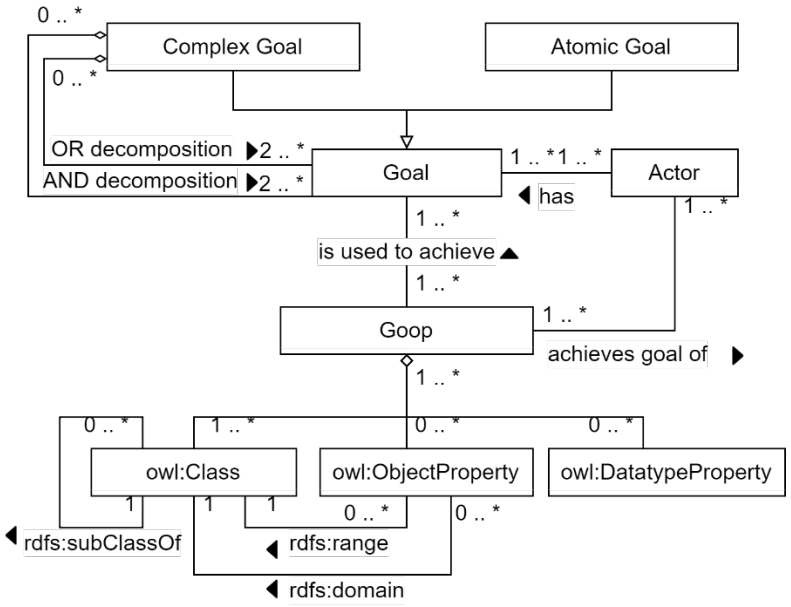

Figure 6. The GOOP-Hub metamodel

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. 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[4], 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[5].

As previously said, different actors may need different ontology fragments to achieve the same goal. In the metamodel shown in Figure 6, the relationships “achieves goal of” between GOOP and Actor, “has” between Actor and Goal, and “is used to achieve” between GOOP and Goal mean that a single

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

GOOP can achieve the goal of one or many Actors and that an Actor may need more than one GOOP to achieve a certain Goal. Moreover, the following constraint must be considered: if a GOOP *gp* is used to achieve a Goal *g* and achieves a goal of an Actor *a*, then *g* must be a goal of *a*.

Currently, there are around 40 GOOPs stored in the GOOP-Hub, covering domains such as measurement, software process, time, biology, chemistry, specimen, scientific activities, patient and locality. They were stored during the development of ontologies built with GO-FOR. Some of them came from works performed in our research group (e.g., (Barcellos et al., 2014), (Ruy et al., 2015) and (Ruy et al., 2016)), others came from external sources (e.g., Interlinking Ontology for Biological Concepts[6] and Place Names Ontology[7]) and others were created from scratch during the development of new ontologies. As we explained in Section 4.1, a GOOP can be created whether using an existing ontology model fragment or building the model fragment from scratch. In the first case, ontologies available in catalogs, libraries, sites, etc. can be used. However, it is important to notice that GOOPs cannot be randomly created. A GOOP is an ontology pattern aimed to achieve a goal, so it can only be created when the goal is known. The random transformation of ontology fragments retrieved from repositories into GOOPs would allow creating artificial goals that could hamper the benefits of using GORE to help ontology reuse. Thus, in order to create a GOOP from existing ontologies, when developing a new ontology, the ontology engineer should identify an ontology fragment that meets his/her goal, reuse it in the new ontology and then create the GOOP by relating the ontology fragment to his/her goal (because once the fragment was used to achieve the ontology engineer goal, it can be related to that goal).

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. The association between a goal and a GOOP is persisted as an OWL relation since the goal and the GOOP are defined as OWL resources. Thus, it is possible to make the goal-based search and find GOOPs by using the metamodel proposed as an OWL structure.

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. Figures 7 and 8 illustrate some screens of GOOP-Hub. Figure 7 shows the initial page of GOOP-Hub and presents the search page, where the user searches for GOOPs based on the goal name (or part of it). Figure 8 presents the GOOP details page, which is reached when the user clicks the button “Show” of a GOOP returned in the search. The page shows the GOOP description, its OWL representation and a picture illustrating its conceptual model. When the ontology engineer selects a GOOP for reuse, he/she can download the GOOP as an OWL file, which can be imported by ontology development supporting tools.

[6] https://bioportal.bioontology.org/ontologies/IOBC
[7] https://raw.githack.com/GeoscienceAustralia/Placenames-Ontology/master/placenames.html

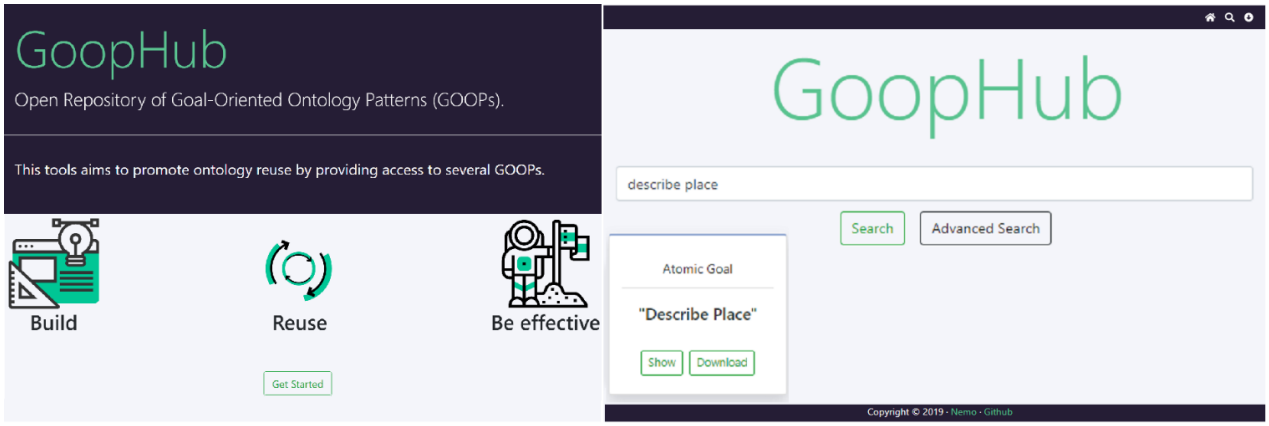

Figure 7. GOOP-Hub initial page and search page

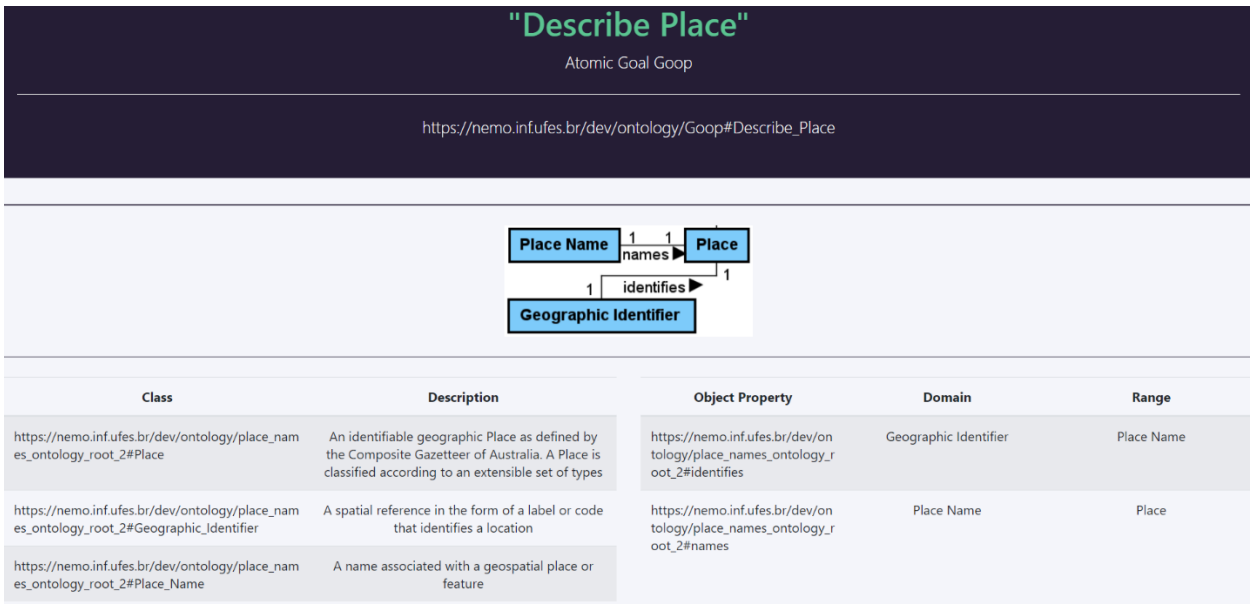

Figure 8. Page detailing the returned GOOP

7. Applying GO-FOR

We have applied GO-FOR in some scenarios in order to evaluate whether it is (i) applicable in real ontology engineering scenarios and (ii) able to assist the ontology engineer to reuse existing ontologies. Moreover, we sought to identify difficulties faced when using GO-FOR. In this section we present some results of GO-FOR use in a research project concerning chronic diseases prevention in Brazil. Brazilian health care data is scattered throughout a number of systems (Unified Health System, private health insurances, private hospitals and clinics). Therefore, it is hard to obtain information on patients' tests and their condition, mostly about chronic diseases such as diabetes and hypertension, which makes it difficult for the government to take direct action to help the population. The referred research project aims to provide an ontology-based solution to integrate health care data and make them available to support

decision making. Hence, an ontology has been used to enable data interoperability. The ontology has been developed by using GO-FOR. Next, we explore a fragment of the ontology to demonstrate GO-FOR use.

As prescribed by the GO-FOR process, the ontology scope was defined by means of goal models. An important actor of the domain of interest is the Laboratory Technician, who is responsible for collecting samples, describing information about patients, obtaining and analyzing data, among others. Figure 9 shows a fragment of the goal model related to the Laboratory Technician, focusing on the goal *Describe Patient Laboratory Test.* The colors in the figure will be further used to identify different GOOPs. Achieving the *Describe Patient Laboratory Test* goal is necessary to verify the patient health condition by analyzing chemical entities properties of collected samples. In the model depicted in Figure 9, such main goal is AND-decomposed in three subgoals, namely: *Describe Patient, Describe Biological Sampling* and *Describe Sample Measurement*. It means that in order to *Describe Patient Laboratory Test*, these three goals must be achieved. *Describe Biological Sampling*, in turn, has an AND decomposition with *Describe Sampling Date/Hour*, *Identify Biological Sample Material* and *Identify Sampling Procedure. Describe Sample Measurement* has an AND decomposition with *Identify Chemical Property*, *Describe Measure*, *Describe Measurement*, *Describe Measurement Analysis*. *Describe Measure* has an AND decomposition with *Define Used Measure* and *Identify Measure Unit*. *Describe Measurement* has an AND decomposition with *Define Measurement Procedure*, *Define Measurement Act* and *Describe Measured Value*. *Describe Measurement Analysis* has an AND decomposition with *Define Measurement Analysis Act*, *Define Reference Values*, *Define Interpretation* and *Define Analysis Result*. At last, *Describe Patient* has an AND decomposition with *Describe Patient Information* and other goals that are not shown here due to scope limitation.

Once the goal model was defined, the next step was to derive the competency questions to detail the ontology scope. For example, some of the competency questions derived from the model depicted in Fig. 9 were: *What was the result of a measurement analysis that analyzed values measured in a biological sample?, Which are the types of procedures to measure samples aiming at patient laboratory test?, What was the measure unit used to express the patient's measured chemical property?, What was the chemical property measured in a measurement?*. The competency questions helped to detail the ontology scope and were particularly useful to evaluate the ontology by indicating which questions the resulting ontology should be able to answer. Based on the goal models, we defined the ontology modularization. For example, taking the goals decomposition shown in Fig. 9 into account, we defined the following modules: Patient, Sampling, and Measurement.

Considering the goal model, we searched the GOOP-Hub looking for GOOPs that meet the established goals. We adopted a top-down approach, first looking for GOOPs related to subgoals of *Describe Patient Laboratory Test.* Since we did not find any GOOP covering those subgoals, we looked for GOOPs related to their subgoals, and so on. We found GOOPs related to the following goals*: Describe Sampling Date/Hour, Identify Sampling Procedure, Describe Measurement, Describe Measure, Define Measurement Analysis Act* and *Define Analysis Result*.

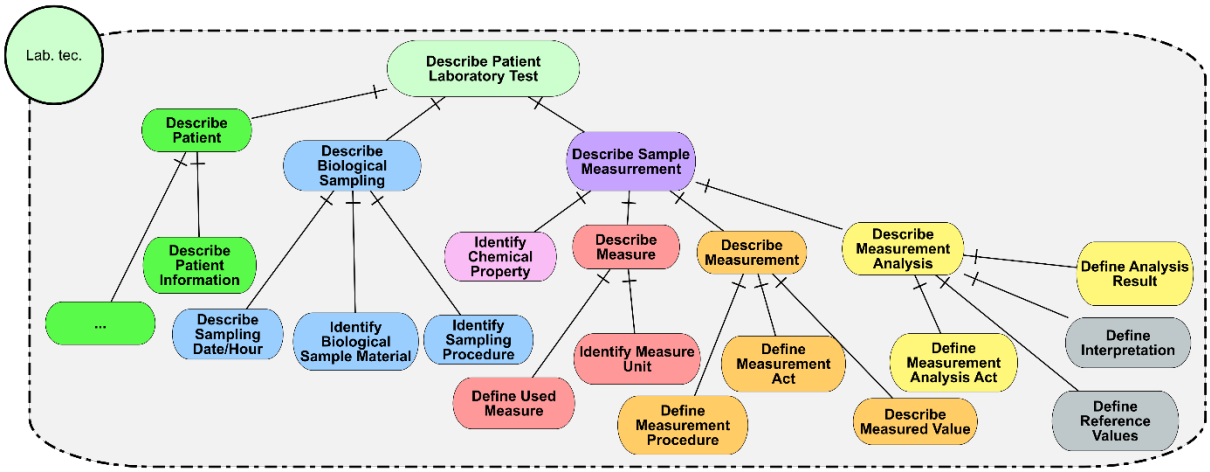

Figure 9. Fragment of the goal model related to Laboratory Technician

Figure 10 presents the GOOP we selected to meet the *Describe Sampling Date/Hour* goal. That GOOP was extracted from the Time Ontology[8] and it had been stored in GOOP-Hub during the development of a Crime Ontology (Santos et al., 2018), built in the context of another project. 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 (Reginato et al., 2019). In Figure 10 and in the other models shown in this section, the concepts' attributes were omitted for simplification reasons. Moreover, the colors used in the model's concepts indicate the goals to which the model is related, according to the colors used in Fig. 9.

Figure 11 depicts an integrated view of the GOOPs we selected to meet the goals *Identify Sampling Procedure, Describe Measurement, Describe Measure, Define Measurement Analysis Act* and *Define Analysis Result*. These GOOPs were stored in the GOOP-Hub during the development of a Measurement Ontology (Santos et al., 2019). In the figure, each region indicates a selected GOOP. Blue region refers to the GOOP selected to achieve *Identify Sampling Procedure*; pink region indicates the GOOP selected to *Describe Measure*; orange region indicates the GOOP selected to *Describe Measurement*, and yellow region indicates the GOOP selected to achieve *Measurement Analysis Act* and *Define Analysis Result*.

A *Measurable Entity* is anything that can be measured, such as a person. Measurable Entity is instance of *Measurable Entity Type* (e.g., Person). Measurable Entity Types are characterized by at least one *Measurable Element* (e.g., Person can be characterized by weight, height, blood pressure). *Measures* are used to quantify Measurable Elements and to characterize Measurable Entity Types. For instance, the measure *weight in kilograms* can be used to quantify the measurable element weight of measurable entities of the type Person. Measures have *Scales* composed of all possible values (*Scale Value*) to be associated by the Measure to a Measurable Element. Measures can be expressed in *Measure Units* (e.g., mg/dL, kilograms).

[8] http://www.w3.org/TR/owl-time/

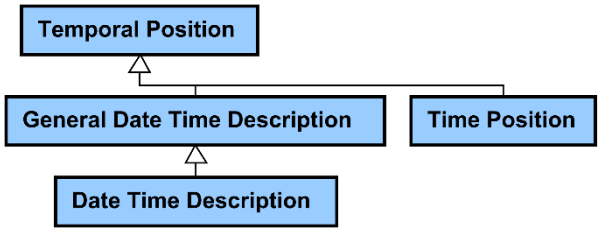

Figure 10. GOOP related to "Describe Sampling Date/Hour"

Measurement is an event that measures a Measurable Element of a Measurable Entity by applying a Measure and adopting a *Measurement Procedure* (which describes the steps to be carried out to collect data for the measure). The result is a *Measured Value*. For instance, the measurement of John's cholesterol by applying the measure cholesterol HDL in mg/dL and adopting the Bichromatic Enzymatic measurement procedure could determine the measured value 51 mg/dL.

Sampling is the act of collecting *Samples* by adopting a *Sample Procedure*. A Sample is a proxy for a Measurable Entity, said a *Sample Represented Measurable Entity*. In other words, a Sample represents a Sample Represented Measurable Entity, which is the Measurable Entity characterized by the Sample. For example, a blood sample represents a person, since by measuring the blood sample it is possible to characterize that person.

Measurement Analysis is the act of analyzing Measured Values to produce an *Analysis Result* that characterizes the measured Measurable Entity. For example, the analysis of the measured value to John's cholesterol indicates that his cholesterol level is acceptable.

After selecting the GOOPs, we used the competency questions related to the respective goals to verify if the GOOPs were enough to meet the goals or if additional concepts were necessary. For example, concerning the goal *Describe Measurement Procedure*, we noticed that the selected GOOP did not provide information to answer the following competency question: *Which are the types of procedures to measure samples aiming at patient laboratory test?*. Thus, we searched outside GOOP-Hub for ontology fragments to complement the selected GOOPs.

Figure 12 shows the selected fragment to address the aforementioned competency question. It was obtained from the Interlinking Ontology for Biological Concepts (IOBC)[9] and needed to be enriched, because its concepts were presented in a tree visualization and considered as being at the same conceptualization level.

We also searched outside GOOP-Hub for ontology fragments to meet the goals not covered by the selected GOOPs. For example, to meet the goal *Identify Chemical Property*, we selected the fragment from the Chebi Ontology[10] depicted in Fig. 14.

In the model, *Chemical Entity* is a physical entity of interest in chemistry, including molecular entities, parts thereof, and chemical substances. A *Chemical Substance* is a portion of matter of constant composition, composed of molecular entities of the same type or of different types. A *Molecular Entity* is any constitutionally or isotopically distinct atom, molecule, ion, ion pair, radical, radical ion, complex, conformer etc., identifiable as a separately distinguishable entity. A *Triglyceride* is any glyceride resulting from the condensation of all three hydroxy groups of glycerol with fatty acids. *Hemoglobin,* in turn, is the iron-containing oxygen-transport metalloprotein in the red blood cells (erythrocytes) of almost all vertebrates.

[9] http://bioportal.bioontology.org/ontologies/IOBC
[10] https://www.ebi.ac.uk/chebi/

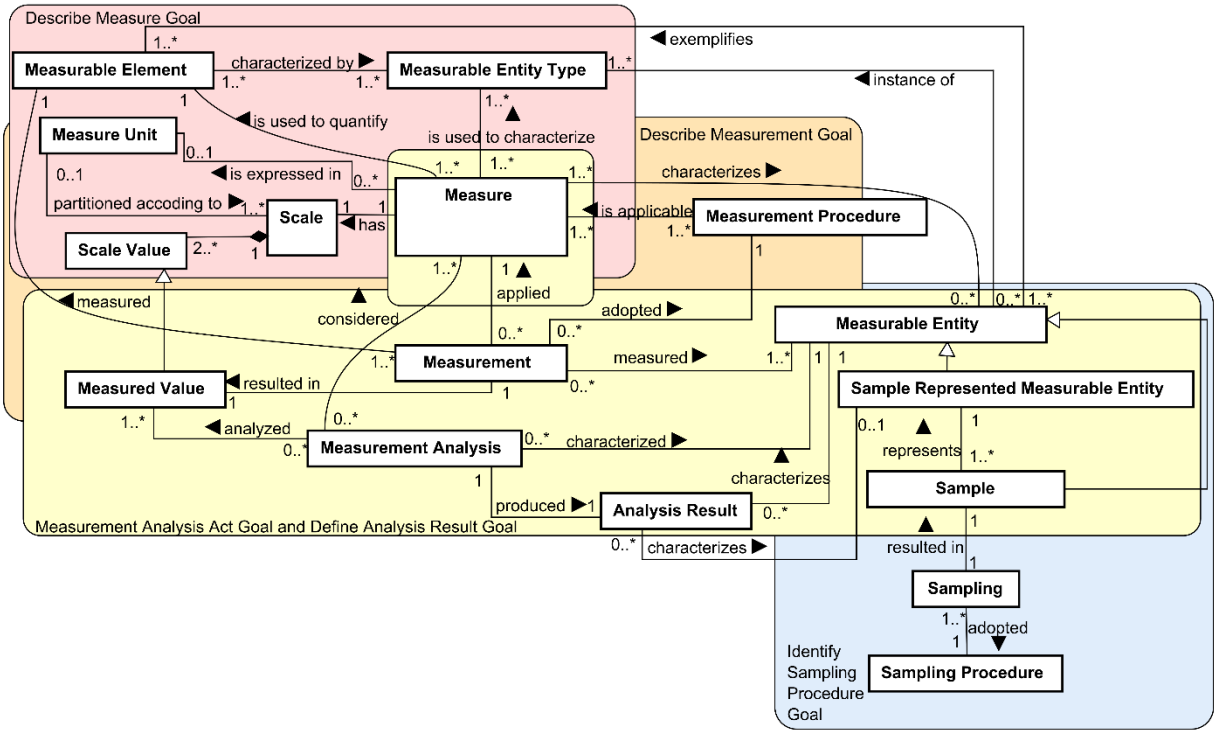

Figure 11. Integrated view of measurement-related GOOPs selected for reuse

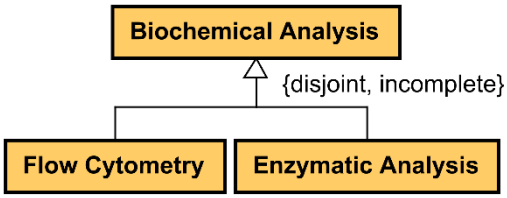

Figure 12. IOBC fragment for complementing the GOOP select to achieve *Describe Measurement Procedure*

After selecting GOOPs and ontology fragments to be reused, we integrated them. We started by identifying the semantic relations between concepts from different GOOPs and ontology fragments. Most of them were relations of specialization, meaning that the concepts from the GOOPs were more generic than the ones from the reused ontologies. In fact, we expected that because the measurement-related GOOPs provided core concepts about measurement and sampling, while the other ontology fragments address specifics of chemical entities examination. When the integration operations were applied, the more specific concepts were related to the more generic ones through specialization relations. When we did not find correspondence between concepts from different GOOPs/ontologies, the concepts were added to the integrated model and suitably related to the other concepts. We also added some new

concepts to address information needs not met by the reused GOOPs and ontologies. Finally, we made some adjustments in concept names to ensure consistency in the terminology used in the ontology.

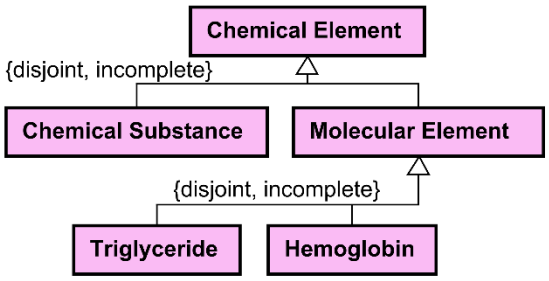

Figure 13. Ontology fragment selected to meet *Identify Chemical Property*

Figure 14 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 Fig. 9. In Figure 14, the concept color indicates the goal to which it relates. Therefore, concepts plus the goal in the same color illustrate fragments of the GOOPs reused to build the ontology. However, it is important to notice that, as shown in Fig. 11, some concepts belong to more than one measurement-related GOOP. The concepts in grey were added to the model to meet the ontology requirements not covered by the reused GOOPs and ontologies. Red is used to indicate relationships created to integrate concepts from different GOOPs as well as new concepts added in order to address the ontology scope.

Biological Sampling refers to a *Sampling* that produces a *Biological Sample*. The values measured in a *Biological Sample* are so-called *Biological Measured Values*. These values are analyzed (during *Measurement Analysis*) by comparing them to *Reference Values of Medical Properties*, which are well-known reference values for a given medical property. Thus, they are values of the *Scale* of the *Measure* used to measure the referred property (*Measurable Element*). Each *Reference Value of Medical Property* leads to an *Interpretation,* which is a lexical value (e.g., "acceptable", "desired", "recommended") considered to determine the *Analysis Result* of a *Measurement Analysis*.

After building the ontology, we evaluated it through verification and validation. During verification, we checked whether the concepts and relations defined in the ontology are able to answer the competency questions. For each competency question (CQ), we identified the elements of the ontology which together are able to address the question. Table 2 shows a fragment of the verification table.

Table 2. Ontology verification (fragment)

Goal	**Describe Biological Sample Material**
Competency Question	CQ14 - Which are the types of biological sample?
Concepts & relations	Biological Sampling resulted in Biological Sample Blood Sample, Swab, Tissue Section subtype of Biological Sample
Goal	**Describe Measurement Analysis**
Competency Question	CQ13 - What is the result of a measurement analysis that analyzed values measured in a biological sample?

Concepts & relations	Biological Measured Value was measured in Biological Sample and is subtype of Measured Value Biological Measured Value is compared to Reference Value of Medical Property that denotes an Interpretation Measurement Analysis analyzed Measured Values and produced Analysis Result that considers Interpretation

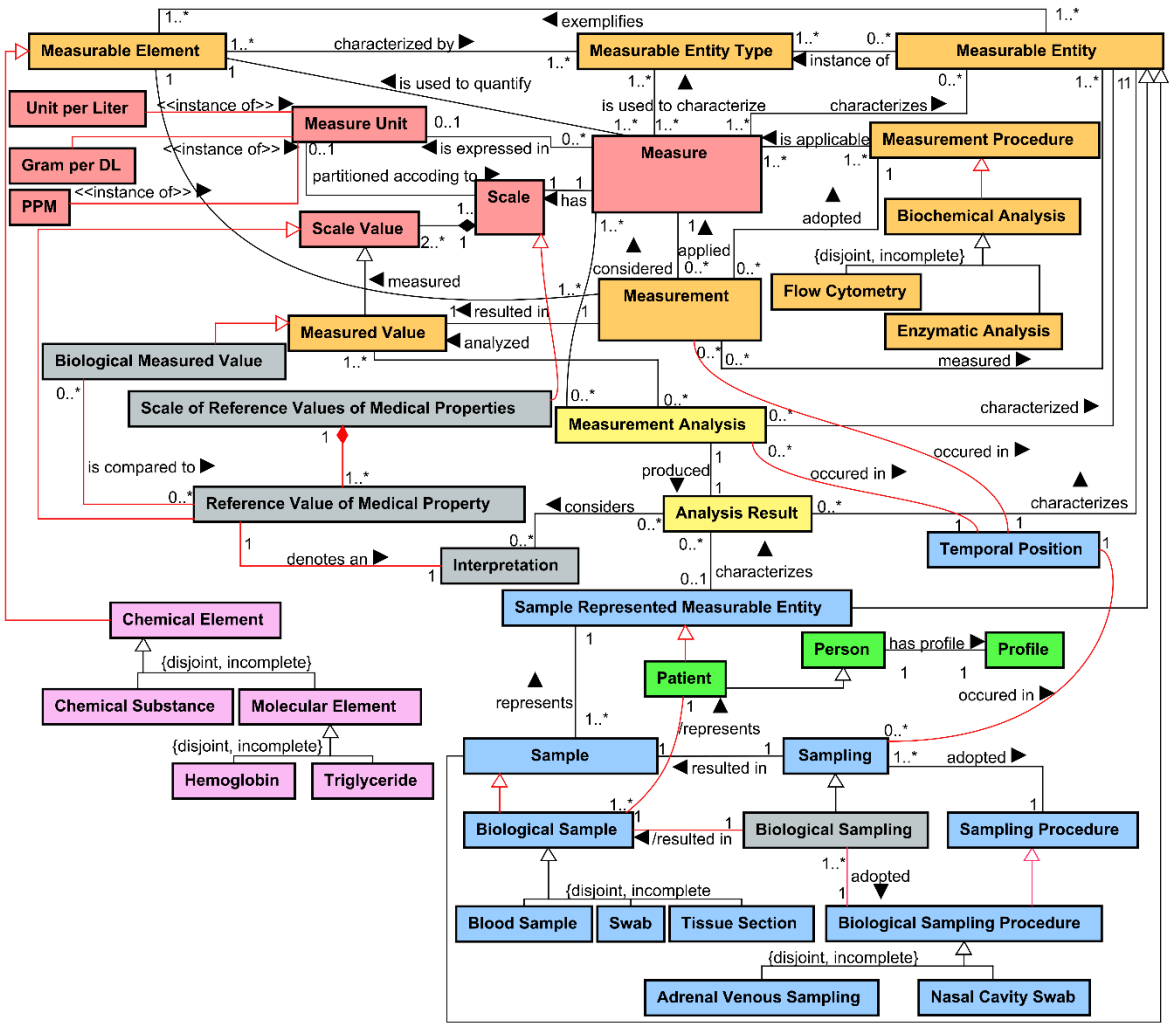

Figure 14. Fragment of resulting ontology

During validation, we aimed to evaluate the ontology by assessing whether it is suitable for representing real-world situations. For that, we instantiated the ontology using data extracted from the research project. Table 3 shows a fragment of the validation table.

Table 3. Ontology verification (fragment)

Concept	Instance
Measurable Entity Type	Person; Blood
Measurable Entity/	Individual X (omitted for privacy reasons); Individual X's Blood Sample
Sample/ Biological Sample /Blood Sample	Individual X's Blood Sample
Sample Represented Measurable Entity/Patient/Person	Individual X
Sampling Procedure/Biological Sampling Procedure	Procedure for Adrenal Venous Sampling
Sampling /Biological Sampling	Act of extracting the blood sample from Individual X and store it in a clinical tube
Measurable Element/ Chemical Element	Hemoglobin
Measure	Hemoglobin in g/dL
Measure Unit	Gram per DL
Measurement	Act of measuring the amount of hemoglobin in the Individual X's blood sample
Measurement/ Biochemical Analysis Procedure	Flow Cytometry
Scale Value/Measured Value/Biological Measured Value	11,8 g/DL
Measurement Analysis	Act of analyzing the measured value of hemoglobin in the Individual X's blood sample
Scale Values/ Reference Value of Medical Property	< 12; 12 to 16; > 16
Interpretation	Low; Normal; High
Analysis Result	Low

Once the ontology was built and evaluated, we extracted new GOOPs to store them in the GOOP-Hub and make them available for future reuse. For example, the fragment shown in Fig. 13 was related to the goal *Identify Chemical Property* and recorded as a new GOOP. Figure 15 presents a GOOP extracted from the resulting ontology to achieve the goal *Describe Biological Sampling*. The GOOP includes concepts from other GOOPs used to build the ontology and also a new concept we added to meet the ontology requirements.

Finally, we implemented the ontology and the GOOPs in OWL and stored them in GOOP-Hub. The ontology has been used to integrate health care data from different sources (spreadsheets, information systems, web sites, PDF files, etc.).

The use of GO-FOR in the case reported in this section showed that using GO-FOR in a real ontology engineering scenario is viable. Moreover, GO-FOR helped reuse existing ontologies, contributing to the quality of the developed model and the productivity of the ontology engineering process. By using GO-FOR, we were able to reuse GOOPs to build most of the model. By reusing GOOPs, we obtained the core conceptualization of our ontology and, thus, we just needed to complement it with concepts from other ontologies and add a few concepts to address specifics of the domain of interest. In the end, we contributed to GOOP-Hub by adding new GOOPs to it, which are available to be reused in the development of other ontologies. As for difficulties we faced when using GO-FOR, we highlight the lack of support to integrate the selected GOOPs and build the ontology integrated model.

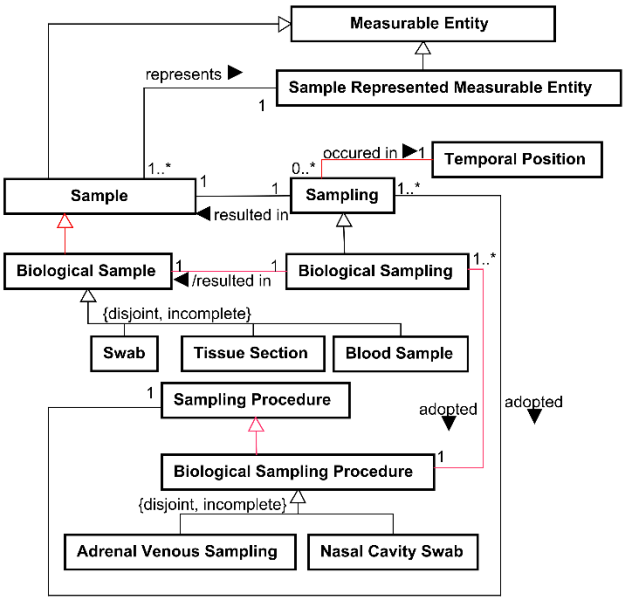

Figure 15. Extracted GOOP

8. 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. Hence, in this section we make some discussions comparing some works cited in Section 2 to GO-FOR.

First, we will consider the cited works devoted to reuse: Ruy et al. (2015), Gangemi and Presutti (2009), and Caldarola and Rinaldi (2016). Our work has some similarities and also differences when compared to them. 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, the basis for the searches and the fact that none of the cited works define a process to create or use the patterns.

In the work presented by Ruy et al. (2015), 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 the work presented by Gangemi and Presutti (2009), 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 the work presented by Caldarola and Rinaldi (2016), design rationale comes from domain experts that help the ontology engineers 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. Regarding computational assistance, Gangemi and Presutti (2009) use a web portal to access the patterns repository; Ruy et al. (2015) use an ontology framework defined in Guerson et al. (2015) to support the definition of pattern libraries in OntoUML (Guizzardi, 2005). Caldarola and Rinaldi (2016) use tools to retrieve from broad search engines ontologies addressing the domain of interest.

Since none of the aforementioned works define a process to support reuse in the context of the ontology development process, we analyzed the GO-FOR ontology development process considering the works by Leung et al. (2011), Cuenca et al. (2017), Pinto and Martins (2001), Keet et al. (2013), Blomqvist et al. (2016), De Nicola and Missikoff (2016), Peroni (2017), and Keet and Ławrynowicz (2016) (here so-called, respectively, LLFT, CLC, PiM, FORZA, XD, Upon Lite, SAMOD, and TDDO).

Again, there are similarities and differences between these works and ours. Concerning similarities, the first five proposals define activities to reuse existing ontologies (or ontology fragments) when developing a new ontology and the last four are inspired by methods or practices from software engineering. As for differences, we highlight coverage to the ontology development process, search for ontologies (or ontology fragments) for reuse, the way requirements are elicited and computational assistance.

Concerning process coverage, like GO-FOR process, XD, LLFT, and CLC cover the ontology development process as a whole, also adding reuse activities to it. PiM, in turn, covers only activities related to the reuse process itself (particularly integration), without taking general ontology development activities (e.g., requirements elicitation) into account. Upon Lite and TDDO cover the development process in general, but they do not add detailed guidelines for reuse. SAMOD provides three general steps for ontology engineering that could be executed iteratively but it does not mention reuse-specific activities albeit having a principle for pattern use. FORZA provides general steps for ontology engineering and has principles for reuse. Only GO-FOR addresses development (thus prescribing specific activities) for reuse.

As for the search of ontologies for reuse, only GO-FOR suggests the use of goal models to support search activities. Moreover, besides GO-FOR, none of the proposals consider the need of enriching the selected ontologies before reuse. XD advocates the search of ODPs in repositories, but the search is not automated in a tool. CLC, Upon Lite, SAMOD and TDDO do not address search of ontologies for reuse. LLFT and PiM suggest searching in ontology libraries and semantic search engines. FORZA leaves it up to the ontology engineer to select the classes they want to use, and suggests the alignment between these classes and a foundational ontology (e.g., DOLCE (Masolo et al., 2003)). This alignment is different from enriching ontologies before the reuse, as addressed in GO-FOR, because in FORZA, it consists of aligning the already defined classes/concepts to the foundational ontology, instead of grounding on a foundational ontology the ontologies selected for reuse.

It is also important to mention the reuse selection criteria. On one hand, XD and TDDO consider competency questions as the main decision criteria. Analogously, CLC evaluates candidate ontologies considering the defined ontology requirements. On the other hand, FORZA preconizes the use of patterns extracted from foundational ontologies, specifically DOLCE. PiM describes a set of questions that help the ontology engineer to evaluate the candidate ontologies for selection. Upon Lite and SAMOD do not address ontology selection criteria. LLFT makes a comparison between key terms defined from competency questions and scenarios and the concepts from the candidate ontologies. The main difference of GO-FOR regarding this aspect is that although competency questions are still applied, goals are added as the main selection criteria, helping the ontology engineer to recognize common goals between the ontology-to-be and the found patterns.

With respect to requirements elicitation, XD uses stories and competency questions to define the ontology requirements, deriving the questions from the stories. It also uses the questions to derive contextual statements and reasoning requirements. LLFT uses scenarios and competency questions, while CLC defines functional and non-functional requirements. PiM and FORZA do not address requirements elicitation. SAMOD uses the so-called motivating scenarios and derives competency questions from them. TDDO advocates for the elicitation of competency questions and axioms, and the questions would be used to derive some ontology axioms as well. Upon Lite goes straight to the creation of a domain-specific terminology, not addressing requirements elicitation. Similar to LLFT, GO-FOR defines

requirements by means of competency questions. However, GO-FOR uses goal models to derive the competency questions and to make explicit motivations for building the ontology.

Last, with respect to computational assistance, only CLC and LLFT present a system to aid the proposed processes. In LLFT there is a key-term system to identify equivalent concepts in candidate ontologies using string-based and linguistic source matching techniques. CLC uses an ontology mapper tool to semi-automatically select the concepts from the reused ontologies and detect duplicates among them. XD has embraced Protégé as its platform using a plug-in for support. FORZA only uses a reasoner that is implemented in a tool to compute properties of part-whole relations. TDDO provides a Protégé plugin to run the tests on the developed ontology. SAMOD does not provide a computational tool, relying on commonly used tools for OWL ontology engineering. Upon Lite suggests the use of spreadsheets for gathering resources for the ontology engineering and the use of plugins to transform the spreadsheets into OWL ontologies. PiM does not mention the use of tools in the methodology. Similar to XD and LLFT, GO-FOR presents a tool (GOOP-Hub) to aid the ontology development process. Different from these works, GOOP-Hub supports the search and storage of ontology fragments related to goals. In GOOP-Hub, the search for candidate GOOPs considers a standard terminology for goals. Moreover, advanced search is provided through a SPARQL end point.

Table 4 summarizes the main aspects analyzed in the works compared to GO-FOR.

Table 4. Summary of aspects analyzed in the approaches

Approach	Coverage	Design Rationale	Search and Selection	Requirements Elicitation	Computational Assistance
Ruy et al. (2015)	Patterns definition and storage	Not explicit	Not addressed	Not addressed	Ontology framework to define pattern libraries
Gangemi and Presutti (2009)	Patterns definition and storage	Not explicit	Search for patterns in repositories	Not addressed	Web portal to access the patterns repository
Caldarola and Rinaldi (2016)	Patterns definition and storage	Not explicit	Not addressed	Not addressed	Tools to retrieve ontologies from broad search engines
Blomqvist et al. (2016)	Ontology development process	Not explicit	Search for patterns in repositories	Stories and competency questions to	Protégé plugin
Leung et al. (2011)	Integration-based ontology development process	Not explicit	Addressed as activities of the ontology development process	Scenarios and competency questions	System to identify equivalent concepts in candidate ontologies
Cuenca et al. (2017)	Ontology network development process	Not explicit	Addressed as activities of the ontology development process	Functional and non-functional requirements	Tool to select concepts from reused ontologies
Pinto and Martins (2001)	Ontology integration process	Not explicit	Addressed as activities of the ontology integration process	Not addressed	None
Keet et al. (2013)	General steps for ontology development with reuse principles	Not explicit	Not addressed	Not addressed	Reasoner to compute properties of relations

Keet and Ławrynowicz (2016)	General steps for ontology development	Not explicit	Not addressed	Competency questions	Protégé plugin
De Nicola and Missikoff (2016)	General steps for ontology development	Not explicit	Not addressed	Not addressed	Spreadsheets and plugins to transform them into OWL ontology
Peroni (2017)	General steps for ontology development with reuse principles	Not explicit	Not addressed	Motivating scenarios and competency questions	Commonly used tools for OWL ontology engineering
GO-FOR	Ontology development process (development with reuse and for reuse) and patterns definition and storage	Made explicit by using goals and goal models	Use of goals to support search and selection	Goal models and competency questions	Tool for pattern definition, storage, and search

In summary, 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 explicitly store design rationale information alongside the patterns or ontologies. Thus, when the ontology engineer follows these approaches, he/she must derive himself/herself the rationale behind the patterns or ontologies found. Finally, only GO-FOR addresses the ontology development process as a whole, considering development with and for reuse, enabling a virtuous reuse cycle.

9. Final Considerations

In this work, we advocate the use of goals as the central elements to define and search for ontology patterns. We propose an evolution of GO-FOR, first introduced in (Reginato et al., 2019). GO-FOR is a goal-oriented and pattern-based framework for ontology reuse, aimed to aid ontology reuse by providing a mechanism for ontology engineers to create and search for ontology patterns based on their goals. By proposing GO-FOR, our intention is to build not only a library of shared, reusable, linkable knowledge, but also a structure to aid the understanding of the underlying rationale and in the reuse of this knowledge.

The main contributions of the work addressed in this paper are the framework itself, the goal-oriented and pattern-based process for ontology development reusing GOOPs and other ontology fragments, and the GOOP-Hub tool. Once the GOOP-Hub is properly populated, it can increase reuse, generating a virtuous cycle of ontology development with reuse and for reuse. A secondary contribution of this paper is the ontology built by using GO-FOR and that has been used to integrate healthcare data.

In GO-FOR, the selection of ontologies (more specifically GOOPs) is made by cognitive analysis (Carriero et al., 2020) and the requirements to be met by the reused ontologies are first defined by means of goals. By doing that, we use a higher level of abstraction than specific requirements (e.g., competency questions) to guide ontology selection. By using goal models, we also enable to address the relations

among the patterns by considering the relations among the goals to which the patterns are associated. Thus, different from catalogues that do not promote the sense of connection among the available patterns, in GO-FOR, GOOPs can be combined according to the goals they to which they are related.

It is worth emphasizing that the purpose of GOOP-Hub is not to extract or consume ontology fragments randomly from ontology repositories, but to record GOOPs when there is a clear understanding of its goal. Thus, human interference is needed to define the goal and connect it to the ontology fragment. We are aware that, on one hand, this limits the reuse of existing repositories. However, on the other hand, by adding to the GOOP-Hub only ontology fragments related to the goals they are able to achieve and following the proposed standardization to record GOOPs, we contribute to the retrieval of GOOPs aligned to the ontology engineer goals and provide detailed information about each GOOP (e.g., description, conceptual model, OWL file).

Besides the project about healthcare data described in this paper, we have also used GO-FOR in previous projects to develop ontologies on water quality (Reginato et al., 2019) and crime (Santos et al., 2018). 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. The results have indicated that the use of GO-FOR is viable and contributes to increase development productivity and improve the quality of the resulting ontology.

As limitations of our work, we highlight GO-FOR evaluation. GO-FOR was used by ontology engineers of our research group. Therefore, we have knowledge of the GOOPs that have been stored in the GOOP-Hub. Moreover, most of the ontology engineers that have used GO-FOR are familiar to GORE. Hence, new applications and studies are necessary to achieve a robust evaluation and increase confidence in the generality of the obtained results. In this sense, we intend to carry out studies, including controlled experimental studies with ontology engineers with different experience levels, as well as new case studies in which the authors will not be directly involved, to assess the usefulness and feasibility of the proposal and the effects of its use on the ontology engineering process (e.g., by evaluating the quality of the resulting ontology and the productivity in the development process, among others).

There are also limitations regarding GO-FOR components. Concerning the GO-FOR process, ontology evaluation must be carried out by the ontology engineer. Thus, it depends on the ontology engineer knowledge and available supporting tools (e.g., OLED (Guerson et al., 2015)). Concerning the definition of goals and goal models, in the GORE field, there are several guidelines on how to define goals and structure them in goal models. These guidelines are also valid when applying GORE to Ontology Engineering. Even so, we believe that some guidelines to help the GO-FOR user to define goals would be helpful. However, we have not defined specific guidelines for defining goals in GO-FOR yet.

As for the GOOP-Hub, up to now, the support we provide to the development of OWL ontology is very basic in terms of exploring the full potential of this language. However, our work focuses on going from the ontology requirements up to developing initial versions of the OWL ontology, placing strong emphasis on conceptual models (e.g., reference ontologies and goal-oriented models) as tools to improve the quality of the ontology under development. We hope that having an OWL draft ontology at hand allows that existing works within the Semantic Web community may be used from then on to improve the ontology's reasoning capabilities.

Concerning the GOOP-Hub search algorithm, the GOOPs are returned based on a search string provided by the ontology engineer based on the goal he/she wants to achieve. Many times, the kind of knowledge is enough to determine that an ontology is appropriate for reuse; other times, however, non-functional requirements such as level of expressivity and reasoning capabilities should be taken into account as well. Thus, although the first criterion we consider to selecting candidate GOOPs is the goal to be achieved, this is not enough. For example, if many GOOPs are returned, it may be difficult for the ontology engineer to identify the one to be reused. Hence, we are working on establishing a set of properties to evaluate GOOPs. Our purpose is to define a set of characteristics (e.g., documentation,

applicability, credibility, etc.) that can help the ontology engineer to identify, among several candidate GOOPs that meet the ontology engineer goal, which one best fits his/her needs. These characteristics can also be useful to estimate the reusability degree of a GOOP. Preliminary results of this study can be found in (Nogueira et al., 2021).

As we explained in Section 1, the work presented in this paper is part of a larger research project. In this paper we presented results obtained in the first part of the research project, which focused on proposing the framework as a whole. We still have a lot of work ahead. As future work, we intend to improve GO-FOR process by providing guidelines on how to define goals and GOOPs (which will also be helpful to evaluate the quality of the defined goals and GOOPs). As for the GOOP-Hub, we plan to populate the repository with new GOOPs and improve the search mechanism to consider different lexicalizations and synonyms. We will also evolve the new feature to assign properties to GOOPs (Nogueira et al., 2021) 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. Alongside, we intend to provide a way for, as GOOPs are used, the user to make assessments of the GOOPs and give information about how they were used (i.e., add context information for reuse). We also plan to include functionalities in the tool to aid the integration of GOOPs to generate new GOOPs and 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.

According to Fernández-López et al. (2019), since ontologies are not being maintained and updated by open communities, a candidate ontology to be reused may not reach minimal adoption requirements or it may not be always available online. A possible approach to lessen these problems could be the creation of communities similar to open software ones in charge of developing and maintaining ontologies. By proposing GO-FOR, we take a step towards this direction. Our intention is to make GOOP-Hub available for people creating, reusing, evaluating and evolving GOOPs.

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

Aljahdali, S., Bano, J., & Hundewale, N. (2011). Goal Oriented Requirements Engineering -A Review. *Proceedings of the 24th International Conference on Computers and Their Applications in Industry and Engineering*, 328–333. https://doi.org/978-1-880843-83-3/ISCA

Barcellos, M. P., Falbo, R. D. A., & Frauches, V. G. V. (2014). Towards a Measurement Ontology Pattern Language. *Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering,* 2014.

Blomqvist, E., Gangemi, A., & Presutti, V. (2009). Experiments on pattern-based ontology design. *Proceedings of the Fifth International Conference on Knowledge Capture - K-CAP '09*, 41-48. https://doi.org/10.1145/1597735.1597743

Blomqvist, E., Hammar, K., & Presutti, V. (2016). Engineering Ontologies with Patterns-The eXtreme Design Methodology. *Ontology Engineering with Ontology Design Patterns- Foundations and Applications*, *25*, 23–50, IOS Press.

Bontas, E. P., Mochol, M., & Tolksdorf, R. (2005). Case Studies on Ontology Reuse. *Proceedings of the IKNOW05 International Conference on Knowledge Management (Vol. 74)*, 345–353. https://doi.org/10.1016/j.sysarc.2006.02.002

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

Caldarola, E. G., & Rinaldi, A. M. (2016). An approach to ontology integration for ontology reuse. *Proceedings of the 2016*

IEEE 17th International Conference on Information Reuse and Integration, IRI 2016, 384-393. https://doi.org/10.1109/IRI.2016.58

Carriero, V. A., Daquino, M., Gangemi, A., Nuzzolese, A. G., Peroni, S., Presutti, V., & Tomasi, F. (2020). The Landscape of Ontology Reuse Approaches. *Studies on the Semantic Web - Applications and Practices in Ontology Design, Extraction, and Reasoning*, *49*(section 1), 21–38. https://doi.org/10.3233/ssw200033

Cuenca, J., Larrinaga, F., & Curry, E. (2017). A unified semantic ontology for energy management applications. *WSP+WOMoCoE*, *Vienna, Austria*, 86–97.

Davis, E. (1998). The Naive Physics Perplex. *AI Magazine*, *19*(4), 51–79.

De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. *Communications of the ACM*, *59*(3), 79–86. https://doi.org/10.1145/2818359

Falbo, R. A., Barcellos, M. P., Nardi, J. C., & Guizzardi, G. (2013). Organizing ontology design patterns as ontology pattern languages. *The Semantic Web: Semantics and Big Data. ESWC 2013. LNCS vol 7882*, 61–75. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38288-8-5

Falbo, R. A., Guizzardi, G., Gangemi, A., & Presutti, V. (2013). Ontology patterns: Clarifying concepts and terminology. WOP'13: *Proceedings of the 4th International Conference on Ontology and Semantic Web Patterns*, 14–26.

Falbo, R. D. A. (2014). SABiO: Systematic approach for building ontologies. *1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*, *1301*.

Fernandes, P. C. B., Guizzardi, R. S. S., & Guizzardi, G. (2011). Using Goal Modeling to Capture Competency Questions in Ontology-based Systems. *Journal of Information and Data Management*, *2*(3), 527–540.

Fernández-López, M., Poveda-Villalón, M., Suárez-Figueroa, M. C., & Gómez-Pérez, A. (2019). Why are ontologies not reused across the same domain? *Journal of Web Semantics*, *57*. https://doi.org/10.1016/j.websem.2018.12.010

Franch, X., López, L., Cares, C., & Colomer, D. (2016). The i* Framework for Goal-Oriented Modeling. *Domain-Specific Conceptual Modeling,* 485–506. Springer.

Gangemi, A., & Presutti, V. (2009). Ontology Design Patterns. *Handbook of Ontologies*, 221–243.

Gruber, T. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, 601–602.

Guerson, J., Sales, T. P., Guizzardi, G., & Almeida, J. P. A. (2015). OntoUML lightweight editor: A model-based environment to build, evaluate and implement reference ontologies. *Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOCW 2015*, 144–147. https://doi.org/10.1109/EDOCW.2015.17

Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. *PhD thesis*. The Netherlands.

Guizzardi, G. (2007). On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. *Databases and Information Systems IV*, *155*, 18–39. https://doi.org/10.1017/CBO9781107415324.004

Guizzardi, G. (2014). Ontological Patterns, Anti-Patterns and Pattern Languages for Next-Generation Conceptual Modeling. *Proceedings of the 33rd International Conference of Conceptual Modeling, ER 2014, Atlanta, GA, USA,* 13–27. https://doi.org/10.1007/978-3-319-12206-9_2

Guizzardi, G., Fonseca, C. M., Almeida, J. P. A., Sales, T. P., Benevides, A. B., & Porello, D. (2021). Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support. *Data and Knowledge Engineering*, *134*. https://doi.org/10.1016/j.datak.2021.101891

Heath, T., & Bizer, C. (2011). Linked Data: Evolving the Web into a Global Data Space. *Synthesis Lectures on the Semantic Web: Theory and Technology,* 1(1), 1-136. https://doi.org/10.2200/S00334ED1V01Y201102WBE001

Hepp, M. (n.d.). Good Relations: An Ontology for Describing Products and Services Offers on the Web*. Knowledge Engineering: Practice and Patterns. EKAW 2008. LNCS, vol 5268, 329-346, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-87696-0_29*.

International Organization for Standardization. (2008). ISO/IEC 25012:2008 - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model. In *ISO/IEC*. https://doi.org/10.1002/(SICI)1097-0142(19990315)85:6<1277::AID-CNCR9>3.0.CO;2-E

Jarczyk, A. P. J., Loffler, P., & Shipmann, F. M. (1992). Design rationale for software engineering: a survey. *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, *ii*, v. 2, 577–586. https://doi.org/10.1109/HICSS.1992.183309

Jureta, I. J., Borgida, A., Ernst, N. A., & Mylopoulos, J. (2010). Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010*, 115–124. https://doi.org/10.1109/RE.2010.24

Katsumi, M., & Grüninger, M. (2016). What is ontology reuse? *Frontiers in Artificial Intelligence and Applications*, 9–22. https://doi.org/10.3233/978-1-61499-660-6-9

Keet, C. M., Khan, M. T., & Ghidini, C. (2013). Ontology authoring with FORZA. Proceedings of the *International Conference on Information and Knowledge Management, Proceedings*, 569–578. https://doi.org/10.1145/2505515.2505539

Keet, C. M., & Ławrynowicz, A. (2016). Test-driven development of ontologies. *Sack H., Blomqvist E., d'Aquin M., Ghidini C., Ponzetto S., Lange C. (eds) The Semantic Web. Latest Advances and New Domains. ESWC 2016. Lecture Notes in Computer Science, v. 9678. Springer, Cham*, 642–657. https://doi.org/10.1007/978-3-319-34129-3_39

Leung, N. K., Lau, S. K., Fan, J., & Tsang, N. (2011). An integration-oriented ontology development methodology to reuse existing ontologies in an ontology development process. *Proceedings of the 13th International Confrence on Information Integration and Web-Based Applications and Services*, 174–181. https://doi.org/10.1145/2095536.2095567

Liu, L., & Yu, E. (2004). Designing information systems in social context: A goal and scenario modelling approach. *Information Systems*, *29*(2), 187–203. https://doi.org/10.1016/S0306-4379(03)00052-8

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., & Schneider, L. (2003). {DOLCE}: a descriptive ontology for linguistic and cognitive engineering. *WonderWeb Project, Deliverable D18 - Ontology Library (Final)*, *Technical Report D18: http://wonderweb.man.ac.uk/deliverables/documents/D18.pdf*

Mealy, G. H. (1967). Another look at data. *Proceedings of theFall Joint Computer Conference - AFIPS '67 (Fall), 14-16*. https://doi.org/10.1145/1465611.1465682

Nogueira, G. G., Barcellos, M. P., & Souza, V. E. S. (2021). Towards a Characterization to Aid in Ontology Reuse. (in press), *Proceedings of the XIV Seminar on Ontology Research in Brazil (ONTOBRAS 2021)*.

Noppens, O., & Liebig, T. (2009). Ontology patterns and beyond towards a universal pattern language. *Workshop on Ontology Patterns*, 179–186.

Park, J., Oh, S., & Ahn, J. (2011). Ontology selection ranking model for knowledge reuse. *Expert Systems with Applications*, *38*(5), 5133–5144. https://doi.org/10.1016/j.eswa.2010.10.002

Partridge, C., de Cesare, S., Mitchell, A., & Odell, J. (2018). Formalization of the classification pattern: survey of classification modeling in information systems engineering. *Software and Systems Modeling*, *17*(1), 167–203. https://doi.org/10.1007/s10270-016-0521-5

Peroni, S. (2017). A simplified agile methodology for ontology development. *Dragoni M., Poveda-Villalón M., Jimenez-Ruiz E. (eds) OWL: Experiences and Directions – Reasoner Evaluation. OWLED 2016, ORE 2016. Lecture Notes in Computer Science, v. 10161. Springer, Cham.* https://doi.org/10.1007/978-3-319-54627-8_5

Pimentel, J., & Castro, J. (2018). PiStar tool - A pluggable online tool for goal modeling. *Proceedings of the 2018 IEEE 26th International Requirements Engineering Conference, RE 2018,* 498-499. https://doi.org/10.1109/RE.2018.00071

Pinto, H. S., & Martins, J. P. (2001). A methodology for ontology integration. *Proceedings of the International Conference on Knowledge Capture KCAP 2001*, 131-138. https://doi.org/10.1145/500737.500759

Presutti, V., Daga, E., Gangemi, A., & Blomqvist, E. (2009). eXtreme design with content ontology design patterns. *Proceedings of the Workshop on Ontology Patterns WOP'09, 516, 83–97, Washington, DC, USA*.

Reginato, C. C., Salamon, J. S., Nogueira, G. G., Barcellos, M. P., Souza, V. E. S., & Monteiro, M. E. (2019). GO-FOR : A Goal-Oriented Framework for Ontology Reuse. *IEEE 20th International Conference on Information Reuse and Integration for Data Science,* Los Angeles, CA, USA, 2019, pp. 99-106, doi: 10.1109/IRI.2019.00028.

Ruy, F. B., Falbo, R. A., Barcellos, M. P., & Guizzardi, G. (2015). Towards an ontology pattern language for harmonizing software process related ISO standards. *Proceedings of the ACM Symposium on Applied Computing*, *13-17-April-2015*. https://doi.org/10.1145/2695664.2695796

Ruy, F. B., Reginato, C. C., Santos, V. A., Falbo, R. A., & Guizzardi, G. (2015). Ontology Engineering by Combining Ontology Patterns. *Proceedings of the 35th International Conference on Conceptual Modeling (ER 2015),* 173–186. https://doi.org/10.1007/978-3-319-25264-3_13

Ruy, F. B., Falbo, R. de A., Barcellos, M. P., Costa, S. D., & Guizzardi, G. (2016). SEON: A software engineering ontology network. *Blomqvist E., Ciancarini P., Poggi F., Vitali F. (eds) Knowledge Engineering and Knowledge Management. EKAW 2016. Lecture Notes in Computer Science,* vol 10024, 527–542, Springer, Cham. https://doi.org/10.1007/978-3-319-49004-5_34

Ruy, F. B. (2017). *Software Engineering Standards Harmonization: an Ontology-Based Approach.* PhD Thesis. Brazil.

Safyan, M., Khan, S., Latif, K., & Masood, A. (2008). Bridging hierarchical ontologies for interoperability and query reformulation. *Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering, ICACTE 2008*, 832–836. https://doi.org/10.1109/ICACTE.2008.212

Salamon, J. S., Reginato, C. C., & Barcellos, M. P. (2018). Ontology Integration Approaches : A Systematic Mapping. *Proceedings of the XI Seminar on Ontology Research in Brazil*, 161–172.

Salamon, J. S., Reginato, C. C., Barcellos, M. P., & Guizzardi, R. S. S. (2017). Using goal modeling and ontoUML for reengineering the good relations ontology. *Proceedings of the X Seminar on Ontology Research in Brazil*, 91–102.

Santos, L. A., Barcellos, M. P., Falbo, R. A., Reginato, C. C., & Campos, P. C. M. (2019). Measurement task ontology. *Proceedings of the XII Seminar on Ontology Research in Brazil*, 95–106.

Santos, L. A., Miranda, G. M., Campos, S. L., Falbo, R. A., Barcellos, M. P., Silva Souza, V. E., & Almeida, J. P. A. (2018). Using an ontology network for data integration: A case in the public security domain. Proceedings of the *XI Seminar on Ontology Research in Brazil*, 209–220.

Struder, R., Benjamins, V. R. & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data and Knowledge*

Engineering, 25 (1-2), 161–197, doi: 10.1016/S0169-023X(97)00056-6.
Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, 249–262. https://doi.org/10.1109/ISRE.2001.948567
Van Lamsweerde, Axel. (2009). Reasoning about alternative requirements options. *Conceptual Modeling: Foundations and Applications, LNCS v. 5600,* 380–397, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02463-4_20
Yu, E. S. K. (1995). Modelling Strategic Relationships for Process Reengineering*. PhD Thesis*. Canada
Yu, E., Giorgini, P., Maiden, N., & Mylopoulos, J. (2011). Social Modeling for Requirements Engineering: An Introduction. *Social Modeling for Requirements Engineering*, 760. https://doi.org/10.1109/SEW.2006.27.