An Ontology Pattern Language for Service Modeling

Ricardo A. Falbo[1], Glaice K. Quirino[1], Julio C. Nardi[2], Monalessa P. Barcellos[1], Giancarlo Guizzardi[1], Nicola Guarino[3], Antonella Longo[4] and Barbara Livieri[4].

[1] Federal University of Espírito Santo, Vitória, Brazil,
[2] Federal Institute of Espírito Santo, Campus Colatina, Colatina, ES, Brazil,
[3] ISTC-CNR Laboratory for Applied Ontology, Trento, Italy
[4] University of Salento, Lecce, Italy
{ falbo, gksquirino, monalessa, gguizzardi}@inf.ufes.br; julionardi@ifes.edu.br; nicola.guarino@loa.istc.cnr.it; {antonella.longo, barbara.livieri}@unisalento.com

ABSTRACT

The notion of service spans several domains, such as healthcare, education, and information and communication technology (ICT). In this context, service ontologies are very useful for establishing a common understanding of the main concepts and relations involved, as well as for serving as basis for modeling services in different domains. In this paper, we present an Ontology Pattern Language, called S-OPL, providing a network of interconnected ontology modeling patterns covering the core conceptualization of services. S-OPL builds on UFO-S, a commitment-based core ontology for services. S-OPL patterns support modeling types of customers and providers, as well as the main service life-cycle phases, namely: service offering, service negotiation/agreement, and service delivery. The use of S-OPL is demonstrated in a real case in the ICT service domain.

CCS Concepts

• Computing methodologies • Software and its engineering

Keywords

Service, ontology, ontology pattern, ontology pattern language.

1. INTRODUCTION

The service sector is one of the largest economic sectors nowadays. Several enterprises (e.g., companies and government agencies) use service-based business models. To interoperate (internally and externally) such enterprises need to represent and share knowledge about their service models. Ontologies are a useful instrument for knowledge representation and sharing. In the service field, there are several ontologies describing the service phenomena, such as [8] and [7]. Despite their importance in improving the understanding about "service", and in acting as reference models to support enterprises build their own service models, they do not provide guidelines for reuse, in particular for reusing specific proper parts of the ontologies.

SAC 2016, April 04-08, 2016, Pisa, Italy.
© 2016 ACM. ISBN 978-1-4503-3739-7/16/04 $15.00
DOI: http://dx.doi.org/10.1145/2851613.2851840

Due to pragmatic reasons, ontology engineers might want to focus only on some parts of the service phenomena. For instance, someone might want to develop an ontology focusing on service agreement aspects (e.g., contracts and obligations), whereas others might want to focus on service offering aspects (e.g., profile of the target customer community). However, it can be hard to reuse fragments of a service ontology without consistent guidelines on how to select parts of the ontology that are suitable for a set of requirements at hands.

Pattern-oriented ontology engineering approaches have been acknowledged as promising for properly dealing with reuse in ontology development [9]. An ontology pattern (OP) describes a particular recurring modeling problem that arises in specific ontology development contexts and presents a well-proven solution for that problem [4]. However, organizing OPs in catalogues is not enough, since they do not address the strong sense of connection among the patterns. This problem is particularly salient in the case of the so-called Domain-Related OPs (DROPs), since these patterns are very inter-related, being very difficult (if not impossible) to apply them in isolation [2].

Differently from catalogues of patterns, an *Ontology Pattern Language (OPL)* [2] favors reuse by providing a network of interconnected DROPs that provides holistic support for ontology development in a given field. An OPL provides a set of interrelated patterns, plus a process model (a procedure, a script) guiding on how to use and combine them in a specific order, and suggesting patterns for solving some modeling problems.

In this paper, we present the new version of S-OPL (Service OPL). S-OPL is a general service OPL that can be used to support the development of service ontologies for specific domains, such as ICT and Healthcare services. S-OPL comprises patterns covering four main groups: (i) *Service Offering*, which includes patterns to model a service offering to a target community; (ii) *Service Provider and Customer*, which deals with defining types of service providers and customers; (iii) *Service Negotiation/ Agreement*, which concerns the negotiation between provider and customers in order to get an agreement; and (iv) *Service Delivery*, which models aspects related to the actions performed for fulfilling a service agreement.

S-OPL was built by extracting patterns from UFO-S [7], a commitment-based core ontology for services, by following the approach described in [11]. Since the extracted DROPs are very interrelated, and since they tend to be applied in combination to develop a service domain ontology, we organized them in an OPL. An initial version of S-OPL was partially published in [10]. This initial version was applied in a real case and, based on the feedback given by the users, it has been improved in the version reported here. In the current version of S-OPL, new patterns have been introduced. However, the main improvement has been in the process guiding users through the use of the patterns.

The remainder of this paper is organized as follows. Section 2 presents the background and related works. Section 3 presents the new version of S-OPL and discusses how its DROPs have been extracted from UFO-S. Section 4 discusses the application of S-OPL in a real case study in the ICT domain. Finally, Section 5 presents our final considerations.

2. BACKGROUND AND RELATED WORK

There are different types of ontology patterns [4]. In this paper, we are interested in Conceptual Ontology Patterns (COPs). COPs are fragments of either foundational ontologies (Foundational OPs - FOPs) or core reference ontologies (Domain-related OPs - DROPs). They are meant to be used during the ontology conceptual modeling phase and focus only on conceptual aspects, without any concern with the technology or language to be used for implementing the ontology [4].

Ruy et al. [11] discuss how FOPs and DROPs can be extracted from Foundational Ontologies and Core Ontologies, respectively. According to them, DROPs should capture the core knowledge related to a domain, and thus they can be seen as fragments of a core ontology of that domain. DROP complexity can vary greatly depending on the domain portion being represented. Sometimes a DROP contains only two related concepts; in other situations, they can contain a complex combination of concepts and relations. It is important to highlight that the same domain portion can give rise to two (or more) variant patterns. Moreover, sometimes a DROP is structurally open in order to be completed by another DROP.

When extracting DROPs from core ontologies, domain aspects come first. The main rule for a DROP is to represent a model fragment that represents recurrent structures in the domain, regardless of its foundational structure. Thus, while FOPs tend to be generally applied, DROPs for a specific field are very interrelated. For this reason, it is usual to apply DROPs in combination for engineering domain ontologies. Thus, instead of recording DROPs in catalogue of patterns, Ontology Pattern Languages (OPLs) [2] can be used to organize them in a systematic guided application process. The notion of OPL provides a stronger sense of connection between DROPs, expressing several types of relationships among them [2]. Thus, an OPL provides explicit guidance on how to reuse and integrate related patterns into a concrete ontology conceptual model. In this sense, an OPL is more than a catalogue of patterns. It includes, besides the patterns themselves, a process guiding the order to apply them according to the problems to be modeled [2].

For developing S-OPL, we employed UFO-S [7]. UFO-S is a core reference ontology on services, which is grounded on the Unified Foundational Ontology (UFO)[5,6]. UFO-S characterizes the service phenomena by considering service commitments and claims established between service participants (service provider and service customer) along the service life-cycle. UFO-S takes the three basic phases of the service life-cycle into account: service offer, service negotiation/agreement and service delivery. As a core ontology, UFO-S presents general concepts that span across several applications domains in such a way that its conceptualization can be broadly reused. As discussed in [2], a core ontology such as UFO-S is a good choice for being the source of DROPs for a service OPL. By providing a network of patterns and a guide to combine them, S-OPL improves the potential for reuse of UFO-S, by enabling the selective use of parts of UFO-S in a flexible way. This is very important due to pragmatic reasons, since ontology engineers developing service ontologies for specific domains might want to focus on selected aspects of the service phenomena.

The use of OPLs is a recent initiative. Thus, there are still only few works proposing OPLs (such as [1], [2] and [3], respectively for the Measurement, Software Process and Enterprise domains). At the best of our knowledge, there is no similar initiative in the service field. In the next section, we present the current version of S-OPL and discuss how S-OPL was designed.

3. SERVICE OPL

As previously mentioned, S-OPL comprises a set of ontology patterns plus a process describing how to combine them to build a service domain ontology (i.e., an ontology about services in a specific application domain). The S-OPL patterns are represented in OntoUML [6], a UML profile that enables making finer-grained modeling ontological distinctions based on UFO. The S-OPL process, in turn, is represented by means of a UML activity diagram, adapted for representing OPLs. In Figure 1, patterns are represented by action nodes (the labeled rounded rectangles). Patterns groups are delimited by blue rounded rectangles. Initial nodes (solid circles) are used to represent entry points in the OPL, i.e., patterns in the language that can be used first, independently of other patterns. Fork nodes (line segments with multiple output flows) are used to represent parallel paths, i.e., if the ontology engineer decides to follow the fork node input path, then she can follow any path leaving them. Join nodes (line segments with multiple input flows) are used to represent multiple dependency, i.e., to follow the join node output path, the ontology engineer must have already traveled all the join node input paths. Decision nodes (represented by diamonds) are used to represent alternative paths. Thus, if the ontology engineer decides to follow the decision node input path, then she has to select exactly one of the decision node output paths. Sub-groups of patterns shown in dotted rounded rectangles aggregate variant patterns, i.e., a set of patterns that solve the same problem but in different ways. Thus, from this set of patterns, only one of them can be selected. Finally, control flows (arrowed lines) represent the admissible sequences of paths that the ontology engineer can follow in the OPL. By default, a control flow is optional, i.e., the ontology engineer can decide to follow it or not, depending on the scope of the ontology being developed. Thus, the ontology engineer can select a certain pattern and decide not to use any other after that, even if there are control flows connecting that pattern to other patterns. However, when a control flow is stereotyped with <<mandatory>>, this means that the path must be mandatorily followed. Patterns in grey are the ones used in the case discussed in Section 4.

In the following subsections, we present the S-OPL process, and some of its patterns. Due to space limitations, we restrict our presentation to the patterns of the *Service Negotiation and Agreement* group, and some patterns of the *Service Provider and Customer* group. The complete specification of S-OPL is available at http://nemo.inf.ufes.br/projects/opl/s-opl/.

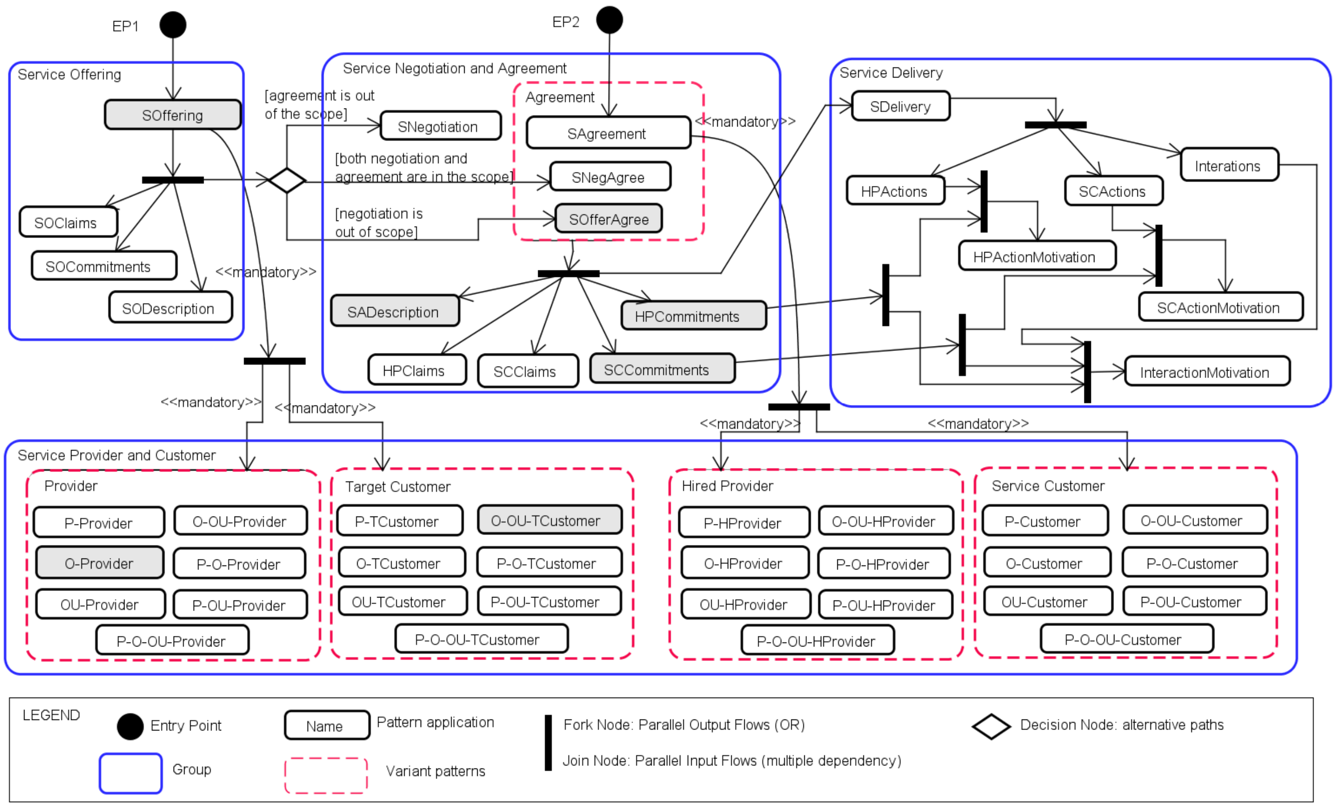


Figure 1 – The S-OPL Process

3.1 The S-OPL Process

As Figure 1 shows, patterns in S-OPL are organized in four groups: *Service Offering*, *Service Negotiation and Agreement*, *Service Delivery*, and *Service Provider and Customer*. These groups were defined mainly based on the modularization of UFO-S, which is decomposed in three sub-ontologies, namely [7]: Service Offer, Service Negotiation, and Service Delivery. A fourth group was introduced here to deal with distinctions among types of providers and customers. UFO-S establishes that providers and customers are roles played by agents of different kinds (persons, organizations and organizational units). This last group makes the possible combinations of types of providers and customers explicit.

S-OPL has two entry points: EP1 and EP2. The ontology engineer should choose one of them, depending on the scope of the specific domain service ontology being developed. When the requirements for her ontology include describing the service offering, then the starting point is EP1. Otherwise, the starting point is EP2.

In case EP1 is chosen, the ontology engineer should use first the *SOffering* pattern for modeling the service offering itself. Next, she must follow the mandatory path: the one that leads to the *Service Provider and Customer* group, which addresses the issue of modeling which types of providers and target customers are involved in the offering. Providers and target customers can be people, organizations or organizational units. Therefore, the ontology engineer must select one of the patterns of the *Provider* sub-group, and one of the patterns of the *Target Customer* sub-group. Besides mandatorily modeling the types of providers and target customers, the ontology engineer can follow the several paths coming out of the fork node. Thus, she can use the patterns *SOClaims* and *SOCommitments*, in the cases in which she is interested in modeling offering claims and commitments, respectively. In addition, she can also chose the *SODescription* pattern, in case she is interested in describing the offering by means of a service offering description.

Once the service offering is modeled, the ontology engineer is able to address problems related to service negotiation and agreement. We should highlight, however, that service offering may be out of the scope of the ontology. In this case, EP2 should be the entry point in the S-OPL process.

If the ontology engineer has already modeled the service offering, she must decide first if she needs to represent service negotiation and/or service agreement. If she wants to model only the service negotiation, without modeling the agreement that could result from it (agreement is out of scope), she should use the *SNegotiation* pattern. If she needs to model both the negotiation and the agreement, then she should use the *SNegAgree* pattern. Finally, if negotiation is out of the ontology scope, then she should use the *SOfferAgree* pattern, which represents an agreement in conformance to an offering.

If EP2 is the entry point in the process, the first pattern to be used is *SAgreement*. In the sequel, the ontology engineer must select one of the patterns of the *Hired Provider* sub-group and one of the patterns of the *Service Customer* sub-group, in order to model the possible types of hired provider and service customer, respectively. The patterns in the *Hired Provider* and *Service Provider* sub-groups are analogous to the ones in the *Provider* and *Target Customer* sub-groups respectively. Note that defining the types of hired providers and service customers is necessary only if the chosen entry point is EP2, since in cases in which the entry point in the process is EP1, the types of providers and target customers would already have been modeled.

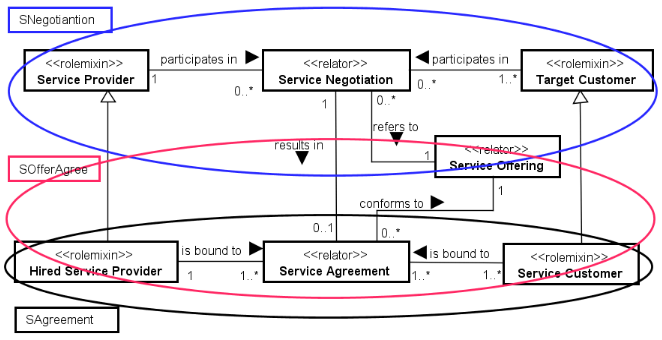
Once the agreement is modeled, the following patterns can be optionally used: *HPCommitments* and *HPClaims*, depending whether the ontology engineer is interested in modeling the hired provider commitments and claims, respectively; *SCCommitments* and *SCClaims*, depending on whether she is interested in modeling service customer commitments and claims, respectively; *SADescription*, in case she is interested in describing the service agreement by means of a description.

After modeling the agreement, the ontology engineer can model the service delivery. In this group, the first pattern to be used is *SDelivery*. In the sequel, if she wants to model the actions involved in a delivery, the following patterns must be applied: *HPActions*, for modeling actions performed by the hired provider; *SCActions*, for modeling actions performed by the service customer; and *Interactions*, for modeling actions performed by both, in conjunction. After that, she can model the relationships between the actions and the commitments that motivate them, by using the following patterns: *HPActionMotivation*, *SCActionMotivation* and *InteractionMotivation*. Since these patterns establish links between commitments and actions, they require the patterns related to the former to be used prior to the patterns related to the latter.

3.2 The S-OPL Patterns

S-OPL patterns were defined according to the approach suggested in [11]. Following this approach, we extracted the patterns from UFO-S by identifying a number of candidate model fragments meaningful for the service domain and framed them as patterns.

The first group of patterns discussed in this paper is the *Service Negotiation and Agreement* group, which consists of nine patterns. This group concerns modeling problems related to the negotiation between target customer and service provider, and the possible agreements between them. Figure 2 shows part of UFO-S Service Negotiation sub-ontology, and how it was decomposed into patterns.



SNegotiantion
<<rolemixin>> Service Provider
participates in
<<relator>> Service Negotiation
participates in
<<rolemixin>> Target Customer
1
0..*
0..*
1..*
1
0..*
refers to
results in
<<relator>> Service Offering
1
SOfferAgree
conforms to
1
0..1
0..*
<<rolemixin>> Hired Service Provider
is bound to
<<relator>> Service Agreement
is bound to
<<rolemixin>> Service Customer
1
1..*
1..*
1..*
SAgreement

Figure 2: Main Patterns of *Negotiation and Agreement* group.

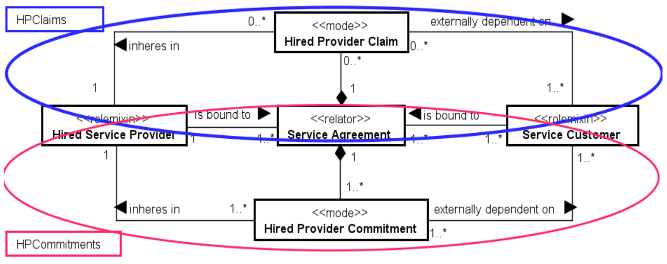
The *SNegotiation* pattern captures the main concepts and relations related to service negotiation. The elements constituting this pattern are circumscribed by the top-most ellipses in Figure 2, and include the following concepts: Service Negotiation, Service Provider, Target Customer and Service Offering. *Service Negotiation* occurs between a *Service Provider* and a set of *Target Customers*, taking as basis a *Service Offering*.

The *SAgreement* pattern captures the main concepts and relations related to service agreement (without representing the negotiation from which it results, neither the offering related to it). The elements constituting this pattern are circumscribed by the bottom-most ellipsis in Figure 2, and include the following concepts: Service Agreement, Hired Service Provider, and Service Customer. This pattern represents the *Service Agreement* between the *Hired Service Provider* and a set of *Service Customers*.

The *SOfferAgree* pattern aims at modeling a service agreement in conformance with a service offering (without representing the service negotiation). The elements constituting this pattern are circumscribed by the second ellipsis (from top to bottom) in Figure 2. Besides modeling the agreement itself, it models also the relationship between *Service Agreement* and *Service Offering* (*conforms to*). When a service agreement is established, the service provider plays the role of *Hired Service Provider*, while the target customer plays the role of *Service Customer*. Moreover, the *Service Agreement* must conform to what was previously established in the corresponding *Service Offering*.

The whole fragment depicted in Figure 2 corresponds to the *SNegAgree* pattern, which models both the negotiation and the agreement, including the relationship between *Service Agreement* and *Service Negotiation* (*results in*).

A *Service Agreement* is composed of commitments and claims (*Hired Provider Commitment, Hired Provider Claim, Service Customer Commitment, Service Customer Claim*) established between the hired service provider and the service customers. Figure 3 shows the patterns describing commitments (*HPCommitments*) and claims (*HPClaims*) of the hired service provider. The *SCCommitments* and *SCClaims* patterns (not shown here) are analogous to these two, respectively, however, modeling commitments and claims of the service customers.



HPClaims
0..*
<<mode>> Hired Provider Claim
externally dependent on
inheres in
0..*
0..*
1
1
1..*
<<rolemixin>> Hired Service Provider
is bound to
<<relator>> Service Agreement
is bound to
<<rolemixin>> Service Customer
1
1..*
1..*
1..*
1
1
1..*
1..*
inheres in
1..*
<<mode>> Hired Provider Commitment
externally dependent on
1..*
HPCommitments

Figure 3: Other Patterns of *Negotiation and Agreement* group.

According to UFO-S [7], *Service Provider* and *Target Customer* are roles played by agents involved in a *Service Offering* (and thus in a *Service Negotiation*), while *Hired Service Provider* and *Service Customer* are roles played by agents involved in a *Service Agreement*. These roles can be instantiated by agents of different kinds (*Person*, *Organization*, and *Organizational Unit*), which may obey different principles of identity. Thus, as the stereotypes in Figure 2 shows, *Service Provider* and *Target Customer* (and their respective subtypes *Hired Service Provider* and *Service Customer*) are defined in UFO-S as rolemixins. According to UFO [6], a *rolemixin* represents an anti-rigid and externally dependent non-sortal, i.e., a dispersive universal that aggregates properties that are common to different roles.

Organizations, *Organizational Units*, and *Persons* can play the roles of service provider and customer. However, depending on the nature of the service being modeled, only certain types of customers and providers are admissible. For instance, the passport issuing service is offered only to people. The car rental service, in turn, is offered to people, organizational units, and organizations. Thus, each pattern in the *Service Provider and Customer* group offers a different option for the ontology engineer to precisely decide what kinds of entities can play the roles of provider and customer in the service domain being modeled.

In fact, these patterns are derived by the application of two foundational patterns (FOPs): the *Rolemixin* FOP [11] and the *Role* FOP [11]. When more than one kind of agent can play a role, the *Rolemixin* FOP applies; when only one kind type of agent can play a role, the *Role* FOP applies. For instance, in the *Target Customer* sub-group of variant patterns, *P-TCustomer* should be used when exclusively persons can play this role. *O-TCustomer* should be used when exclusively organizations can play this role. *OU-TCustomer* should be used when exclusively organizational units can play this role. *O-OU-TCustomer* should be used when both organizations and organizational units can play this role. *P-O-TCustomer* should be used when both persons and organizations can play this role. *P-OU-TCustomer* should be used when both persons and organizational units can play this role. Finally, *P-O-OU-TCustomer* should be used when any of these kinds of entities can play this role. Figure 4 shows two of these patterns. The first is *O-OU-TCustomer*, which is an example of application of the *Rolemixin* FOP. In this pattern, *Target Customer* is a role that can be played by both *Organizations* and *Organizational Units*. Thus, two roles are defined as subtypes of *Target Customer*: *Organization Target Customer*, which is also a subtype of *Organization*; and *Organizational Unit Target Customer*, which is a subtype of *Organizational Unit*. The second pattern shown in Figure 4 is *P-TCustomer*, which is an example of application of the *Role* FOP. In this pattern, since only people can play the role of target customer, *Target Customer* is substituted in the model by *Person Target Customer*, which is a subtype of *Person*.

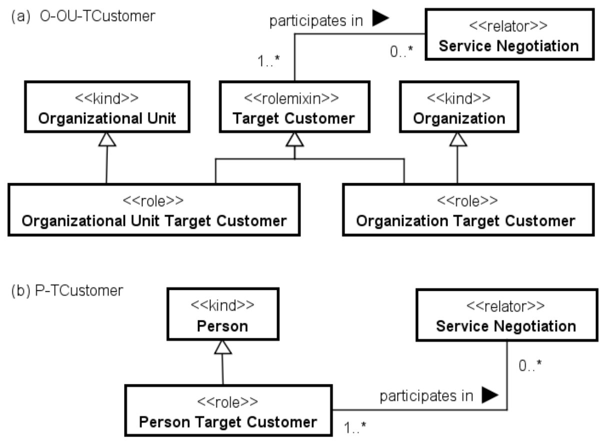


Figure 4: Two Patterns of the *Target Customer* sub-group.

4. APPLYING S-OPL IN A REAL CASE

We have employed S-OPL in a real case study to model an email service in a big Italian company. The IT Department is responsible for this service and decided to hire two underpinning ICT services provided by two different organizations: the *Emailbox service* and the *Networking service*. With the goal of establishing a common understanding among the stakeholders, the Chief Information Officer asked for a conceptual model addressing the main aspects involved in the email service. S-OPL was used to develop this model, named here the *Email Service Ontology (ESO)*. For building ESO, the patterns in grey in Figure 1 were used. It is worth saying that ESO was developed by ontology engineers not involved in the S-OPL development.

In this case, EP1 was the entry point chosen, and thus *SOffering* was the first pattern applied. In the sequel, following the mandatory path leaving *SOffering*, the patterns *O-Provider* and *O-OU-TCustomer* were selected. Since the ICT services were provided by organizations, the *O-Provider* pattern was selected. For modeling target customers, the *O-OU-TCustomer* pattern was selected. This was because the target customers in this application domain include both *Organizations* and *Organizational Units*.

Once service offerings were modeled, patterns from the *Service Negotiation and Agreement* group were selected. The first pattern applied was *SOfferAgree*, for modeling service agreements and their relationship with the corresponding service offerings. Next, the patterns *HPCommitments* and *SCCommitments* were applied to capture the commitments involved in the agreement. Finally, the *SADescription* pattern was used to model the contract describing the agreement. In the remainder of this section, we discuss in more details how we have built the ESO fragment related to Service Agreement (Figure 5).

After modeling the service offering, the *SOfferAgree* pattern was used. According to this pattern, a service agreement involves both the hired service provider and the customer celebrating the agreement. In our case, a *Business Customer Organizational Unit* celebrates *ICT Service Agreements* with the *Hired Emailbox Service Provider Organization* and the *Hired Network Email Service Provider Organization*. Such service agreements are in conformance to the corresponding service offerings.

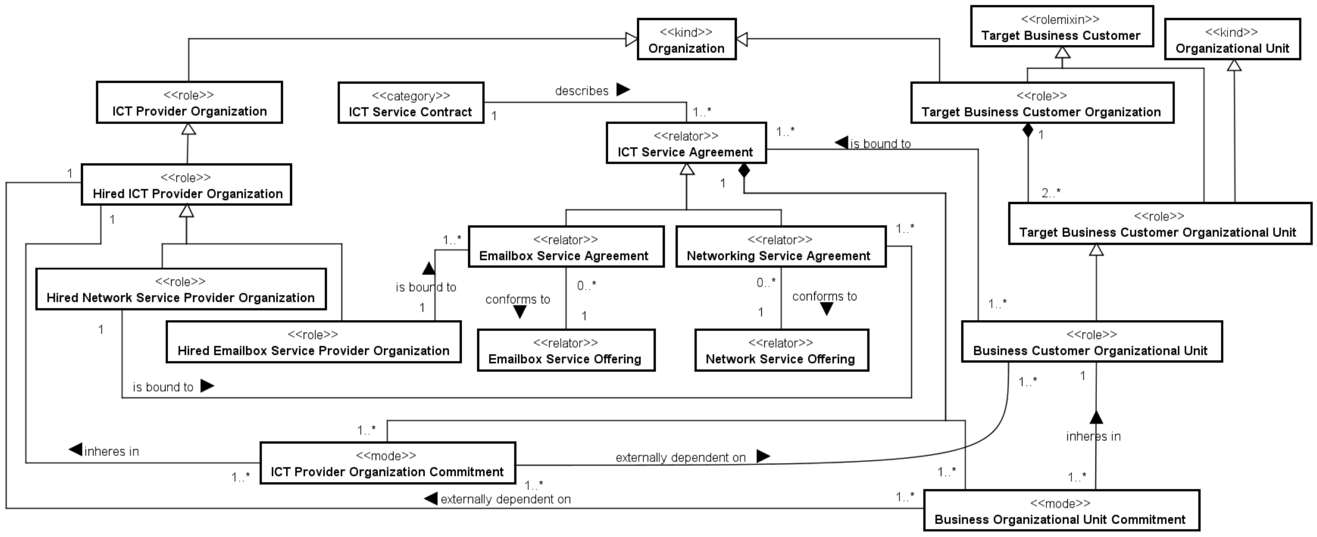
For modeling the terms and conditions of the agreement, two patterns were used: *HPCommitments* and *SCCommitments*. Thus, both commitments of the *Hired ICT Provider Organizations* and of the *Business Customer Organizational Unit* can be specified.

Finally, the *SADescription* pattern was applied. From that, the *ICT Service Contract* was defined for describing the corresponding service agreements.

5. FINAL CONSIDERATIONS

Currently, reuse is recognized as an important practice for Ontology Engineering. Ontology patterns are considered a promising approach that favors reuse of encoded experiences and good practices in Ontology Engineering [9]. Moreover, core ontologies organized as Ontology Pattern Languages (OPLs) have potential to amplify the benefits of ontology patterns [2]. In line with these beliefs, we have been developing S-OPL, a Service OPL. In this paper we presented the current version of S-OPL, discussing how its patterns were derived by following the approach described in [11].

During the development of the real case study discussed in Section 4, we perceived some benefits to the development of the *Email Service Ontology*. Firstly, the resulting ontology tends to contain less inconsistence problems, since many of the potentially recurring source of inconsistencies in the service domain tend to be solved by the basic patterns of the core ontology. Secondly, the accomplishment of the ontology development process tends to be faster by the massive reuse of modeling fragments with their respective axiomatization and modeling decisions embedded in the patterns. Finally, S-OPL guides pattern selection, facilitating their combination.



<<kind>> Organization
<<rolemixin>> Target Business Customer
<<kind>> Organizational Unit
<<role>> ICT Provider Organization
<<category>> ICT Service Contract
describes
<<role>> Target Business Customer Organization
<<relator>> ICT Service Agreement
is bound to
<<role>> Hired ICT Provider Organization
<<role>> Target Business Customer Organizational Unit
<<relator>> Emailbox Service Agreement
<<relator>> Networking Service Agreement
<<role>> Hired Network Service Provider Organization
is bound to
conforms to
conforms to
<<role>> Hired Emailbox Service Provider Organization
<<relator>> Emailbox Service Offering
<<relator>> Network Service Offering
<<role>> Business Customer Organizational Unit
is bound to
inheres in
inheres in
<<mode>> ICT Provider Organization Commitment
externally dependent on
externally dependent on
<<mode>> Business Organizational Unit Commitment
1
1..*
2..*
0..*

Fig. 5. Service Agreement Sub-Ontology.

We should also highlight that, since the patterns constituting S-OPL are described in OntoUML, they carry out the ontological and formal semantics embedded in the micro-theories (e.g., about role mixins, roles, relators, etc.) comprising the Unified Foundational Ontology (UFO) [6]. As a consequence, many of them are systematically constructed via the manifestation of UFO foundational patterns. Moreover, the structures constituting these patterns are carried out and, hence, are also manifested in the ontologies created using S-OPL.

Although we have perceived these benefits, experiments with subjects have to be conducted to truly confirm them. In particular, we envision two interesting experiments: one comparing the construction of domain service ontologies by directly extending UFO-S with by using S-OPL; the other comparing the use of S-OPL with the use of the catalogue of patterns comprising S-OPL without the S-OPL process. The first experiment could allow us to evaluate whether the pattern approach truly improves the ontology development process. The second aims at evaluating the real benefits of providing a guide for using the patterns.

6. ACKNOWLEDGMENT

This research is funded by the Brazilian Research Funding Agency CNPq (Processes 485368/2013-7 and 461777/2014-2) as well as the “Science Without Borders” project on “Ontological Foundations of Service Systems” funded by CAPES/CNPq.

7. REFERENCES

[1] Barcellos, M.P., Falbo, R.A., and Frauches, V.G. Towards a Measurement Ontology Pattern Language. First Joint Workshop Onto.Com/ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering, (2014).

[2] Falbo, R.A., Barcellos, M.P., Nardi, J.C., and Guizzardi, G. Organizing Ontology Design Patterns as Ontology Pattern Languages. Proceedings of the 10th Extended Semantic Web Conference - ESWC 2013, (2013).

[3] Falbo, R.A., Ruy, F.B., Guizzardi, G., Barcellos, M.P., and Almeida, J.P.A. Towards an Enterprise Ontology Pattern Language. Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM (2014), 323–330.

[4] Falbo, R.A., Guizzardi, G., Gangemi, A., and Presutti, V. Ontology Patterns: Clarifying Concepts and Terminology. Proceedings of the 4th Workshop on Ontology and Semantic Web Patterns, (2013).

[5] Guizzardi, G., Falbo, R.A., and Guizzardi, R.S.S. Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): the Case of the ODE Software Process Ontology. Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments, (2008), 244–251.

[6] Guizzardi, G. Ontological Foundations for Structural Conceptual Models, Universal Press. 2005.

[7] Nardi, J.C., Falbo, R.A., Almeida, J.P.A., et al. A commitment-based reference ontology for services. *Information Systems* 54 (2015), 263–288.

[8] Oberle, D., Bhatti, N., Brockmans, S., and Janiesch, C. Countering service information challenges in the internet of services. Journal of Business & Information System Engineering (BISE) 5, (2009).

[9] Presutti, V., Daga, E., Gangemi, A., and Blomqvist, E. eXtreme Design with Content Ontology Design Patterns. Proc. Workshop on Ontology Patterns, (2009).

[10] Quirino, G.K., Nardi, J.C., Barcellos, M.P., et al. Towards a Service Ontology Pattern Language. Proceedings of the 34th International Conference on Conceptual Modeling (ER2015), Springer (2015), 187 - 195.

[11] Ruy, F.B., Reginato, C.C., Santos, V.A., Falbo, R.A., and Guizzardi, G. Ontology Engineering by Combining Ontology Patterns. 34th International Conference on Conceptual Modeling (ER’2015), Springer (2015), 173–186.