An Ontology Design Pattern for Representing Relevance in OWL

Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero*

Department of Computer Science and Artificial Intelligence
E.T.S. Ingeniería Informática y Telecomunicaciones, University of Granada
c/. Periodista Daniel Saucedo Aranda, s/n 18071 Granada Spain
fbobillo@decsai.ugr.es, mdelgado@ugr.es, jgomez@decsai.ugr.es

Abstract. Design patterns are widely-used software engineering abstractions which define guidelines for modeling common application scenarios. Ontology design patterns are the extension of software patterns for knowledge acquisition in the Semantic Web. In this work we present a design pattern for representing relevance depending on context in OWL ontologies, i.e. to assert which knowledge from the domain ought to be considered in a given scenario. Besides the formal semantics and the features of the pattern, we describe a reasoning procedure to extract relevant knowledge in the resulting ontology and a plug-in for Protégé which assists pattern use.

1 Introduction

Semantic Web is nowadays more than a promise and ontology-based applications are blooming here and there, permeating the Web environment with RDF and OWL aromas (mixed with microformats, folksonomies and other contributions from the Web 2.0 brand). Nevertheless, despite the efforts that research community has put on providing better tools to manage formal metadata, a classical issue in Artificial Intelligence is still paining the neck of developers: knowledge acquisition bottleneck. Acquiring, reusing, representing and eliciting knowledge to build an ontology becomes frequently an exhausting, time-wasting and frustrating experience, even when collaborative experts, proper tools and sound methodologies are in play.

Consequently simple recipes which support ontologists to apprehend aspects of their application domain are highly appreciated. This is the objective of ontology design patterns: to describe, more or less formally, recurrent modeling scenarios and to provide guidelines for incorporating this knowledge into ontologies correctly. By *correctly* we mean obtaining as a result accurate, transparent and reasonable representations, as pointed out in [1].

One of these common situations is the need of representing relevance of information. It is usual that an intelligent system manages so many information resources that it is impractical to provide a user with all the available data in

* Corresponding author. Tel: +34 958243194. Fax: +34 958243317

response to a query, since it will take too long to filter them manually or simply he will not be able to process it. This issue has been pointed out in the literature with the name of "information overload" [2]. In such case, only relevant information should be delivered and, in order to do so, relevance relations must be represented in the system knowledge base.

As a result of our work in Knowledge Mobilization –the effort to make knowledge available for real-time use, no matter where decision-making processes are taking place–, we strongly believe that what is relevant (or significant) for a user mostly depends on his circumstance[1]. The *circumstance* can be regarded as a mix of environment facts, beliefs, intentions and needs, i.e. (in a wide sense) his context. Accordingly, relevance should be represented in a knowledge base by defining relations among descriptions of usage cases and subsets of the domain knowledge. Having a domain and a scenario connected with such a relation means that this information is important and must be considered in that situation.

In this work we present a design pattern for representing and managing relevance relations in an OWL ontology. Besides the formal semantics of the model, we provide an algorithm to extract context-dependant summaries by reasoning within the ontology. We also introduce a graphical tool which eases the application of the pattern in the ontology development process.

The remaining of this document is structured as follows. Section 2 reviews some related work about ontology design patterns and context representation. Section 3 describes our design pattern; use cases, notation, and formulation are detailed. This section is clarified with the example in Section 4. Section 5 overviews our supporting software tool for applying the pattern. Section 6 includes a discussion of our proposal, as well as an analysis of its computational complexity. Finally, Section 7 points out some conclusions and directions for future work.

2 Related work

Design patterns are concise guidelines which identify common design problems and suggest how to resolve them. Patterns have been recognized as a valuable tool since the very beginning of design sciences, from architecture to software development. Analysis and design patterns are important meta-artefacts which support the design process of software systems, as stressed by [3].

Templates and patterns to build knowledge bases have been proposed in several papers, some specific for a concrete application domain (e.g. [4]), some more general and (even) language-independent (e.g. [5]). Ontologies for the Semantic Web have their own peculiarities, so a task force inside the W3C Semantic Web Best Practices and Deployment Working Group[2] was settled to elaborate best practices and patterns for OWL, namely the Ontology Engineering and Patterns

[1] Although he probably was not considering mobile knowledge-based systems, Spanish philosopher Ortega y Gasset (1883-1955) summarized this idea in his maxim "I am myself and my circumstance" (*Meditaciones del Quijote*, 1914).

[2] http://www.w3.org/2001/sw/BestPractices/

Task Force[3]. The work of this task force was partially inspired by [6], a classical ontology-development guide which includes some tips to build them properly.

Ontology development patterns can be considered the extension of software engineering ones. In [7] some differences between both are described from a Semantic Web perspective, remarking that more formality is required in the presentation of the former, which are called CODePs (Conceptual Ontology Design Patterns). In [1] different design-support artifacts for Semantic Web ontologies are overviewed and some examples of patterns are briefly presented.

Regarding representation of relevance of information depending on context, the aim of this paper, to the best of our knowledge there does not exist any design pattern specifically fitted to OWL particularities. Our proposal is OWL-DL compliant, unlike other approaches about contextualizacion of knowledge models which concerns non-monotonic formalisms, i.e. models which are satisfiable or not depending on some circumstances. It is interesting however to remark the contribution in [8], which examines some classical works about contexts and microtheories in Artificial Intelligence, and extends some of these ideas to solve context-dependant aggregation problems in the Semantic Web. Similarly, [9] proposes C-OWL, an extension to OWL to define mappings between locally-interpreted and globally-valid ontologies. To end up, we shall mention that the idea underlying our model is quite similar to the multi-viewpoint reasoning in [10], though it concentrates on the conditional interpretation of a model (how to reduce an ontology depending on the viewpoint submodel), whereas we focus on their relevance (in which circumstances a submodel should be considered).

On the other hand, some lessons can be learned from recent works in Pervasive Computing, as they are concerned with context awareness, content filtering and significance representation [11]. Moreover, as the example in Section 3.1 shows, we are especially interested in Ubiquitous Computing and Knowledge Mobilization, so the connection is even clearer. Not surprisingly, ontologies have been proposed to be used for modeling context knowledge in some recent developments in Pervasive Computing, e.g. [12] [13] [14].

3 Definition of the pattern

This section describes the formal semantics of our proposal, the so called Context-Domain Relevance (CDR) pattern. We follow the recommendations sketched by [5] and [7], covering aspects as use cases, notation (in Description Logics language) and syntax (formulation).

3.1 Use case

Let us suppose a physician who needs to consult a patient's clinical data in order to set a proper treatment for him. If the healthcare act is taking place

[3] http://www.w3.org/2001/sw/BestPractices/OEP/

inside the hospital, the doctor will be allowed to access the Hospital Information System (HIS) and to retrieve all the patient's Electronic Health Records (EHRs). Having enough time and knowledge, the specialist will rule out all the useless pieces of information and will get the ones he is interested in. We can consider now another physician in an emergency-assistance unit which is caring at the road for a patient injured in an accident. Knowing some data about his clinical history will be helpful as well in this situation; for instance, some data about patient's adverse drug events (ADEs) may have been recorded. Nevertheless it is not probable that the HIS can be accessed from outside the hospital (even less using a portable device as the one which will be likely used in emergency healthcare) and, if possible, the doctor would not have enough time to review all the stored electronic records.

In the latter situation, a brief report including those pieces of the patient's clinical data which ought to be considered would be very valuable. The clinical procedure which is going to be carried out would determine which data should be part of this summary. For example, is the patient is slightly unconscious and has an hemorrhagic laceration, information about if he has been diagnosed of bad reactions to procaine (an anesthetic drug which reduces bleeding but is also often badly metabolized and triggers allergic reactions) should be taken into account, among others. This would be a prototypical sample of a Knowledge Mobilization application, in contrast to the former example which depicts a typical use case of a classical Information System.

Two different kinds of knowledge are to be managed by such mobile system: (i) domain knowledge about the problem which must be resolved (this is made up by the patients' electronic health records), and (ii) context knowledge about the scenarios where the domain knowledge will be used (for our doctor, this would be a vocabulary to briefly describe the situation of the patient he is going to attend). To state which knowledge from the domain must be considered in each scenario, links between both submodels can be defined. Continuing our example, a link asserting that 'data about previous anesthetic drugs reactions' should be considered when 'the patient has a penetrating wound' should be created. Other links can be similarly included following recommendations of clinical and ADE guidelines. Building these links is the aim of the CDR pattern.

3.2 Notation

As it is known, ontologies rely on Description Logics (DL), a family of logics for representing structured knowledge. In this section we overview the basics and the notation of DL. This notation will be used to describe the pattern and is directly translatable to OWL syntax.

In the remaining of this section we will consider the minimal subset proposed in the logic $\mathcal{ALC}$ (attributive concept description language with complements), since it is expressive enough to encode our pattern. $\mathcal{ALC}$ is less expressive than $\mathcal{SHOIN}(\mathcal{D})$ (almost equivalent OWL-DL, the highest descriptive level of OWL which ensures decidibility) and therefore its complexity is lower. The reader is referred to [15] for further readings about DL.

Formally, an ontology is a triple $O = \langle K_R, K_T, K_A \rangle$, where K_R (the Role Box or RBox) and K_T (the Terminological Box or TBox) comprise the intensional knowledge, i.e. general knowledge about the world to be described (statements about roles and concepts, respectively), and K_A (the Assertional Box or ABox) the extensional knowledge, i.e. particular knowledge about a specific instantiation of this world (statements about individuals in terms of concepts and roles).

In $\mathcal{ALC}$ there is no RBox, since no axioms involving roles are allowed. In more expressive logics, K_R consists of a finite set of role axioms stating restrictions as subsumption, transitivity, cardinality, etc.

An $\mathcal{ALC}$ TBox K_T consists of a finite set of general concept inclusion (GCI) axioms of the form $C_1 \sqsubseteq C_2$, which means that concept C_1 is more specific than C_2 , i.e. C_2 subsumes C_1. A concept definition $C_1 \equiv C_2$ (C_1 and C_2 are equivalent) is an abbreviation of the pair of axioms $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. Concept expressions for C_1, C_2 can be derived inductively starting from atomic primitives. Valid constructs for $\mathcal{ALC}$ are: $C_1, C_2 \rightarrow A$ (atomic concept) | $\top$ (top concept) | $\bot$ (bottom concept) | $C_1 \sqcap C_2$ (concept conjunction) | $C_1 \sqcup C_2$ (concept disjunction) | $\neg C_1$ (concept negation) | $\forall R.C_1$ (universal quantification) | $\exists R.C_1$ (full existential quantification).

An $\mathcal{ALC}$ ABox consists of a finite set of assertions about individuals (noted a and b). An assertion is either a concept assertion $a : C$ (a is an instance of C) or a role assertion $(a, b) : R$ ((a, b) is an instance of R).

A DL ontology not only stores axioms and assertions, but also offers some reasoning services, such as KB satisfiability (or consistency), concept satisfiability, subsumption or instance checking. In $\mathcal{ALC}$ most inference services are mutually reducible, so only some of them are usually considered.

3.3 Formulation

In Sect. 3.1 we mentioned two knowledge sub-models that are involved in our design pattern. These correspond to the domain ontology and the context ontology and are the basis over which the relevance ontology is built.

The domain ontology $O^D = \langle K^D_R, K^D_T, K^D_A \rangle$ contains the knowledge required to solve the concrete problem that the system is facing. As expected, concepts of this ontology represent entities with associated semantics, roles establish connections among them, and instances represent individuals of this world. This ontology can be arbitrarily complex and is closely related to the problem. We will use the notation $D_j \mathring{\in} O^D$ to name complex concepts expressions D_j built using elements in O^D and ontology constructs. Note that, in principle, these D_j are not part of the domain ontology.

The context ontology $O^C = \langle K^C_R, K^C_T, K^C_A \rangle$ contains the knowledge required to express the circumstances or the *surroundings* under which the domain knowledge will be used. The context ontology can be seen as a (formal) vocabulary or *lingo* with which these situations can be described. Being strict, context knowledge is not part of the original problem, though it can be indispensable to solve it; in fact, it would be possible to reuse the same context model in completely different areas. Context knowledge can range from low-level sensor data (like location, time or humidity) to abstract information (like preferences, desires or mental state). We will use the notation $C_i \overset{\circ}{\in} O^C$ to name complex concepts expressions C_i built using elements in O^C and ontology constructs. Like in the previous case, these C_i are not necessarily part of the context ontology.

Intuitively, we can guess that a CDR ontology will be made of new classes (the so called profiles) which will relate C_i context concepts with D_j domain concepts through quantified roles. We must note that, accordingly, our proposal only considers the intensional component of the knowledge base.

Regarded this we define constructively a CDR ontology as follows:

Definition 1. *Let O^D and O^C be, respectively, the domain ontology and the context ontology, $C_i \overset{\circ}{\in} O^C$ a context concept built with K_T^C classes, and $D_j \overset{\circ}{\in} O^D$ a domain concept built with K_T^D classes.*

The CDR ontology which relates the set of pairs of concepts $\{(C_i, D_j)\}$ (i.e. states that D_j is interesting when C_i happens) is an ontology $O^P = \langle K_T^P, K_A^P \rangle$ which satisfies:

1. $K_A^P = \emptyset$
2. K_T^P include definitions for the concepts $P_\top, C_\top, D_\top, P_{i,j}, C_i, D_j$, where:
   (a) $P_\top, C_\top, D_\top$ are the super-classes Profile, Context and Domain:
$$P_{i,j} \sqsubseteq P_\top \wedge P_\top \equiv \bigcup_{i,j} P_{i,j}$$
$$C_i \sqsubseteq C_\top \wedge C_\top \equiv \bigcup_i C_i$$
$$D_j \sqsubseteq D_\top \wedge D_\top \equiv \bigcup_j D_j$$
   (b) R_1 is the bridge property linking profiles and context concepts:
$$P_\top \sqsubseteq \forall R_1.C_\top$$
   (c) R_2 is the bridge property linking profiles and domain concepts:
$$P_\top \sqsubseteq \forall R_2.D_\top$$
   (d) $P_{i,j}$ is the profile linking named context C_i and named domain D_j :
$$P_{i,j} \equiv \exists R_1.C_i \sqcap \exists R_2.D_j$$
3. O^P is consistent.

Figure 1 depicts the meaning of this definition. It shows how $P_{i,j}$ concepts are a reification of the “relevance” relation between context and domain concepts. Representing relevance as a concept and not as a role presents some advantages, e.g. possibility of reusing previously-defined profiles or defining new properties (with associated semantics) for them.

The main reasoning task within the CDR ontology will be to find the domain restricted by a context, that is, to find all the classes of the domain ontology which are associated using profiles (i.e. are relevant) with a given concept built with the context vocabulary. This can be expressed as follows:

Definition 2. *Given O^C, O^D and O^P, the restricted domain of the scenario $S \overset{\circ}{\in} O^C$ (being S a complex concept expressed in K_T^C vocabulary) considering O^P comprises all the classes I such as:*

$$\left\{ I \in K_T^D \mid (S \sqsubseteq C_n) \wedge \left(P_{n,m} \in K_T^P\right) \wedge (I \sqsubseteq D_m) \right\}$$

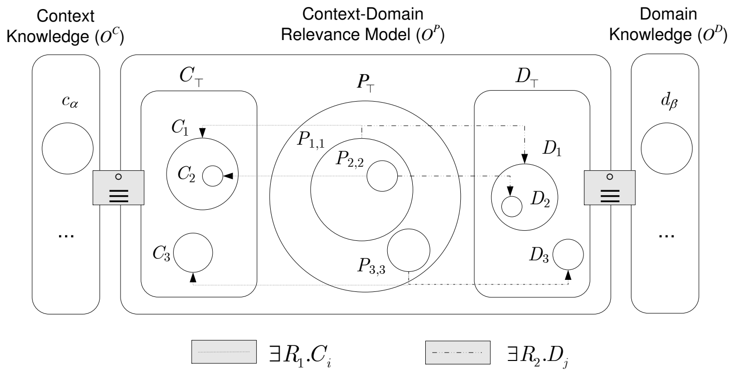

Fig. 1. Schema of the Context-Domain Relevance ontology

Algorithm 1. *The restricted domain of a scenario S considering O^P can be computed in practice as follows:*

1. *Retrieve all the named contexts C_n which subsume S:*
$$\{C_n \sqsubseteq C_\top | S \sqsubseteq C_n\}$$
2. *Retrieve all the named profiles $P_{k,l}$ which include C_n contexts (via R_1):*
$$\{P_{k,l} \sqsubseteq P_\top | \, (P_{k,l} \sqsubseteq \exists R_1.C_k) \wedge (C_n \sqsubseteq C_k)\}$$
3. *Retrieve all the named domains D_m which are related to $P_{k,l}$ profiles (via R_2):*
$$\{D_m \sqsubseteq D_\top | P_{k,l} \sqsubseteq \exists R_2.D_m\}$$
4. *Retrieve all the classes I from K_T^D which are subsumed by D_m:*
$$\left\{I \in K_C^D | I \sqsubseteq D_m\right\}$$

4 Example

Continuing with the medical case we have sketched in Sect. 3.1, let us suppose the following sample ontologies:

- A domain ontology O^D abstracting the information units managed by the hospital information system. Among others, it includes concepts as *Patient*, *ElectronicDocument* or *ElectronicRegisterCoagulationDisorder*, and properties as *relatedToPatient* (with domain equals to *Patient* and range equals to *ElectronicDocument*). Instances of this ontology are the concrete values of patient's electronic health records.
- A context ontology O^C defining a suitable vocabulary to describe patient situations. It will contain concepts as *Hemorrhage*, *Unconsciousness*, *Trunk* or *High*, and properties like *hasSeriousness* (from *ClinicalFact* to *Seriousness*).

Using the definition of the CDR model, an ontology O^P can be built to reflect which information from the information system must be considered when facing each clinical case. With $hasClinicalFact$ and $hasElectronicRegister$ being the bridge properties R_1 and R_2, respectively, the profiles in Table 1 will be valid. It can be observed that $C_3 \sqsubseteq C_2 \sqsubseteq C_1$ and $D_1 \sqsubseteq D_3$, $D_2 \sqsubseteq D_3$.

Table 1. Example of a Context Domain Restriction Ontology

Top concepts
$P_\top \sqsubseteq \top$ $C_\top \sqsubseteq \top$ $D_\top \sqsubseteq \top$
Profile 1,1. When the patient is "unconscious" and "hemorrhagic", registers about "blood pressure disorders" must be checked
$C_1 \equiv Unconsciousness \sqcap Hemorrhage$ $D_1 \equiv ElectronicRegisterBloodPressureDisorder$ $P_{1,1} \equiv \exists hasClinicalFact.C_1 \sqcap \exists hasElectronicRegister.D_1$
Profile 2,2. When the patient is "unconscious", "hemorrhagic" and has a "penetrating wound", registers about "drug intolerances" must be checked
$C_2 \equiv Unconsciousness \sqcap Hemorrhage \sqcap PenetrationWound$ $D_2 \equiv ElectronicRegisterDrugIntollerance$ $P_{2,2} \equiv \exists hasClinicalFact.C_2 \sqcap \exists hasElectronicRegister.D_2$
Profile 3,3. When the patient is "unconscious", with a "highly serious" "hemorrhage" and has a "penetrating wound", registers about "blood pressure disorders", "drug intolerances" and "coagulation disorders" must be checked
$C_3 \equiv Unconsciousness \sqcap (Hemorrhage \sqcap \exists hasSeriousness.High) \sqcap PenetrationWound$ $D_3 \equiv ElectronicRegisterBloodPressureDisorder \sqcup ElectronicRegisterDrugIntollerance \sqcup ElectronicRegisterCoagulationDisorder$ $P_{3,3} \equiv \exists hasClinicalFact.C_3 \sqcap \exists hasElectronicRegister.D_3$

Given this profile set, if the doctor is attending to a "hemorrhagic and unconscious patient with a penetration wound", the system will answer that electronic records about "drug intolerances" should be checked. This is achieved by using the previous algorithm to calculate the restricted domain of a context concept. The process is shown in Table 2.

The final information to be delivered to the doctor's mobile device will be the instances of the $ElectronicRegister$ classes in I filtered by patient ID, which must mirror the data stored in the hospital information system. It would be possible to store these instances in a RDF specific repository, avoiding the overload of having them embedded in the ontology.

Table 2. Resolution of the example

(1)	$S \equiv Hemorrhage \sqcap Unconsciousness \sqcap PenetrationWound$ $C_n = \{C_1, C_2\}$
(2)	$P_{k,l} = \{P_{1,1}, P_{2,2}\}$
(3)	$D_m = \{D_1, D_2\}$
	Let us suppose that $ElectronicRegisterBloodPressureDisorder$ is a leaf concept in O^D and $ElectronicRegisterDrugIntollerance$ has two subclasses: $ElectronicRegisterProcaineIntolerance$ and $ElectronicRegisterPenicillinIntolerance$. Then:
(4)	$I =$ $ElectronicRegisterBloodPressureDisorder$ $ElectronicRegisterDrugIntollerance$ $ElectronicRegisterProcaineIntolerance$ $ElectronicRegisterPenicillinIntolerance$

Note that in Algorithm 1 descendants of S are not inferred during the reasoning process, since these concepts corresponds to more specific context situations –which will probably drive to more specialized domain information–. However, it may be interesting to calculate the profiles involving these subcontexts and to provide them as feedback information to the user, in order to recommend him to describe further details of the current scenario. For instance, in this example, C_3 in $P_{3,3}$ is subsumed by S:

$$Unconsciousness \sqcap (Hemorrhage \sqcap \exists hasSeriousness.High) \sqcap \\ PenetrationWound \sqsubseteq Unconsciousness \sqcap Hemorrhage \sqcap PenetrationWound$$

Consequently the doctor could be advised to carry out other clinical trials to see if the specific part of this restriction ($\exists hasSeriousness.High$ qualifier of $Hemorrhage$) is present but has not been diagnosed yet. If this knowledge is supplied afterwards, more information about the patient (information unit $ElectronicRegisterCoagulationDisorder$) will be provided.

5 CDR Plug-in for Protégé

We have developed a plug-in for the Protégé platform (the ontology development tool from the University of Stanford[4]) which allows to create, edit, test and reason with a CDR ontology. Our plug-in adds a new tab to the Protégé-OWL environment (Protégé enhanced with the OWL plug-in[5]) where a simplified view of the CDR ontology is displayed and queries can be introduced. A preliminary version can be downloaded in http://arai.ugr.es/iaso/cdrplugin/.

We can distinguish four sections in the tab, depicted in Figure 2:

[4] http://protege.stanford.edu

[5] http://protege.stanford.edu/plugins/owl/

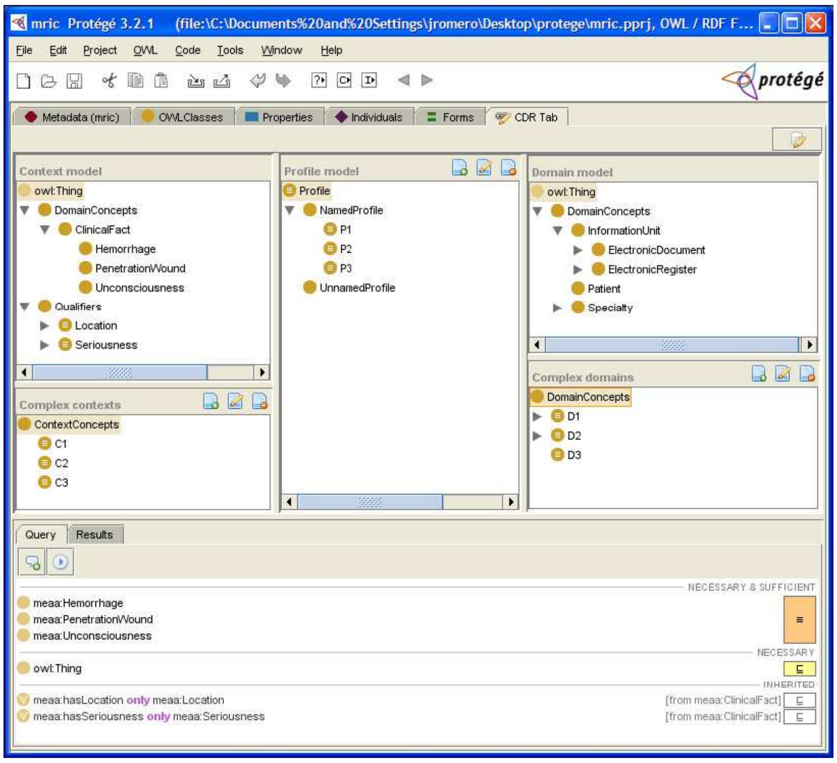

Fig. 2. CDR Tab plug-in in Protégé IDE

1. *Context side.* The left side of the tab shows the context ontology (O^C) and the complex contexts (C_i) existing in the profile ontology. The context ontology can be optionally hidden. New complex contexts can be created using the context vocabulary; existential restrictions for the new C_i are automatically added. It is also possible to edit or delete existing profiles.
2. *Domain side.* The right side mirrors the context side but changing context knowledge by domain knowledge (O^D). New complex domains D_i can also be easily created and editing and deleting are as well allowed.
3. *Profiles.* The central section of the tab shows the profiles in the ontology O^P. This is probably the most interesting part, since it simplifies the task of creating new profiles. To build a new profile, the user has just to select a complex context in the left box (C_i) and a complex domain in the right box (D_j), and then push the 'new profile' button. The new profile ($P_{i,j}$) will be created as a subclass of the selected profile and the corresponding existential restrictions will be automatically generated.
4. *Reasoning.* The bottom section allows to retrieve the domain relevant to a given context, i.e. it implements Algorithm 1. When a new complex concept for querying is created, its restrictions are shown and the 'run query' but-

ton is activated. Results are displayed in the 'Results' tab of this reasoning section and further information about the obtained classes can be consulted.

From the formal description of the pattern in Section 3.3 it can be deduced that some additional configuration is needed to make the CDR plug-in work correctly. This involves stating the URIs of the external ontologies (O^C, O^D), the top concepts for the profiles, the contexts and the domains ($P_\top$, $C_\top$, $D_\top$) and the URI of the DIG reasoner which will be used. To assist this procedure, a wizard-like window is presented to the user when pushing the 'properties' button on the top toolbar.

The plug-in has been developed with the APIs for Protégé and Protégé-OWL version 3.2.1. It also relies on the CDR-API, our library to manage programmatically models created with the pattern. Installation is easy; as any other Protégé add-on, it just has to be copied to the plug-in directory of the Protégé installation.

6 Discussion

Next we summarize some of the highlights of the CDR pattern. Studying computational properties of the resulting ontology deserves its own subsection, where complexity is detailed.

6.1 Features

– *Reusability.* By definition, design patterns must be applicable to different problems and domain areas. Our pattern effectively fulfills this objective, since it provides a general guideline for representing relevance without imposing application-dependant restrictions on the domain and the context ontologies.
– *Standardization.* One of the main a priori requirements for our pattern was OWL-DL compliance, that is, the resulting ontology should not include new constructors nor be in OWL-Full. As explained in Section 3.3, the pattern generates a new OWL-DL ontology whose complexity is bounded by context and domain models. Thus, though the reasoning process may seem little straightforward, current tools (e.g. inference engines) can be directly used, without having to extend, modify or re-implement them.
– *Formalization.* We have provided a formal specification of the pattern which goes further than usual text descriptions. This is possible because the target language, OWL, relies on a logic-based formalism, DL.
– *Modularization.* The pattern promotes ontology modularization, as it clearly separates the three involved models. Nevertheless, a limitation is imposed by OWL importing mechanism: the profile ontology O^P must import completely O^C and O^D, as partial including is not allowed. This forces the model to be globally interpreted and valid, which would not be desirable if different (and probably contradictory) relevance criteria and contexts are to be represented.

– *Expressivity.* The pattern allows to represent relevance taking the most of OWL expressivity. For instance, profile hierarchies can be defined to assert inclusion relations between them. In fact, the resulting model is an OWL ontology and can be modified as needed. Further improvements may be considered, e.g. definition of several bridge properties with different semantics to qualify the connections between contexts and domain, or adding properties to profile classes.

6.2 Complexity analysis

Computational complexity of the inference within the CDR model is conditioned by complexity of context and domain expressions ($C_i \overset{\circ}{\in} K_T^C$ and $D_j \overset{\circ}{\in} K_T^D$), since $P_{i,j}$ definitions are included in $\mathcal{ALC}$ level. In the simplest case, that is O^C, O^D and O^P ontologies are in $\mathcal{ALC}$, reasoning within the CDR ontology is asymptotically bounded by ontology classification complexity, which is ExpTime for $\mathcal{ALC}$ with GCIs according to Table 3 [16].

Table 3. Complexity of reasoning in basic DLs

DL \ TBox	acyclic	general
$\mathcal{FL}^-$	PTime	PTime
$\mathcal{AL}$	coNP	PSpace
$\mathcal{ALE}$	coNP	PSpace
$\mathcal{ALU}$	PSpace	ExpTime
$\mathcal{ALC}$	PSpace	ExpTime

Supposing that O^C and O^D do not add further complexity, it is possible to reduce the complexity of the CDR model by restricting the allowed constructors for C_i and D_j, moving consequently to a less expressive logic. Restricting negation to atomic concepts and disallowing union concepts would enclose the CDR ontology to $\mathcal{ALE}$, which has PSpace complexity for general reasoning. Other alternative consists on using only acyclic TBoxes, which would give complexities of PSpace for $\mathcal{ALC}$ and coNP for $\mathcal{ALE}$.

Other choices are not appropriate, however. Moving from $\mathcal{ALC}$ to $\mathcal{ALU}$ does not reduce the complexity, neither in the general case nor with acyclic TBoxes. Moving to $\mathcal{AL}$ is not possible, because existential quantification can not be restricted. Similarly, expressivity of $\mathcal{FL}^-$ is too limited.

According to this formulation, role hierarchies are not necessary in the CDR model. Nevertheless, they may be considered for convenience, in such a way that sub-roles of R_1 and R_2 can be defined with particular semantics and handled consequently. This will increase the complexity to $\mathcal{ALCH}$, but with the advantage that reasoning for the general case still remains ExpTime.

In any case, $\mathcal{ALCH}$ is less expressive than $\mathcal{SHIF}(\mathcal{D})$ (equivalent to OWL-Lite), so reasoning in practice with available DL engines (e.g. Pellet[6]) will be quite efficient, as they are highly optimized and worst-case inferences are infrequent. Hence, more complex logics with extended semantics could be as well considered to extend the basic formulation without significant performance impact.

7 Conclusions and future work

In this work we have presented the CDR formalism, a design pattern for the representation and management of context-relevant knowledge in OWL ontologies. This pattern eases the representation of knowledge when facing the problem of information overload in KBSs, which is critical in Knowledge Mobilization. We also provide a plug-in for the Protégé ontology-development platform which simplifies constructing, editing and consulting the relevance model; currently this software is being tested and new features are being suggested to be implemented. Finally, we have discussed the main features of the pattern, remarking reusability and standardization as the more important, and studied computational complexity of the resulting ontology.

Looking into the future, we strongly believe that describing and promoting best practices for Semantic Web ontologies is not only useful but also necessary to boost semantic applications. More design recommendations and patterns as the produced by the OEP Task Force should be publicly available for use and discussion. It is interesting to note that in turn an eventual pattern repository could be described using an ontology.

Concerning our design pattern, in Sect. 6.1 we have remarked that OWL imports can be problematic when more than one context or profile model is involved. This issue has been pointed out in some current works and some solutions have been proposed [17]. Evolution of the relevance model is also important and temporal and non-monotonic reasoning formalism may be further considered. In fact, representing validity (as in temporal and non-monotonic logics) depending on time or new knowledge (both can be assimilated to context) and relevance (as in our model) might be regarded as similar ideas. It would be interesting to compare both approaches and to study to which extent one can be reduced to the other.

A fuzzy and probabilistic/possibilistic extension to the crisp ontology generated by the CDR pattern is also being considered. Such fuzzy ontology would allow to define weighted relevance relations between context and domains and, which is more interesting, partial matching of similar contexts. For instance, a context could be asserted to be subsumed by another with a certain degree. This would make the CDR ontology no longer compliant with OWL-DL, so we are as well studying procedures to reduce a fuzzy ontology to a crisp one [18].

[6] http://pellet.owldl.com/

Acknowledgement. This research has been partially supported by the project TIN2006-15041-C04-01 (Ministerio de Educación y Ciencia). F. Bobillo holds a FPU scholarship from Ministerio de Educación y Ciencia. J. Gómez-Romero holds a FD scholarship from Consejería de Innovación, Ciencia y Empresa (Junta de Andalucía).

References

1. Svátek, V.: Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In: Proceedings of BIS2004. (2004)
2. Eppler, M., Mengis, J.: The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines. The Information Society **20**(5) (2004)
3. Iivari, J.: Information Systems as a Design Science. In: Information Systems Development: Advances in Theory, Practice and Education. Springer (2005)
4. Reich, J.R.: Ontological Design Patterns: Metadata of Molecular Biological Ontologies, Information and Knowledge. In: Database and Expert Systems Applications: 11th International Conference, DEXA 2000. (2000)
5. Staab, S., Erdmann, M., Maedche, A.: Engineering ontologies using semantic patterns. In: Workshop on E-Business and Intelligent Web. (2001)
6. Noy, N., McGuinness, D.: Ontology Development 101: A Guide to Creating Your First Ontology. Technical report, Stanford Knowledge Systems Laboratory (2001)
7. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: Proceedings of the 4th International Semantic Web Conference. (2005)
8. Guha, R., McCool, R., Fikes, R.: Contexts for the Semantic Web. In: Proceedings of the 3rd International Semantic Web Conference. (2004)
9. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. Web Semantics: Science, Services and Agents on the World Wide Web **1**(4) (2004)
10. Stuckenschmidt, H.: Toward Multi-viewpoint Reasoning with OWL Ontologies. In: The Semantic Web: Research and Applications. (2006)
11. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. **3**(3) (1999)
12. Chen, H., Finin, T., Joshi, A.: The SOUPA Ontology for Pervasive Computing. In: Ontologies for Agents: Theory and Experiences. Birkhuser Basel (2005)
13. Gu, T., Pung, H., Zhang, D.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications **28**(1) (2005)
14. Khedr, M., Karmouch, A.: ACAI: agent-based context-aware infrastructure for spontaneous applications. Journal of Network and Computer Applications **28**(1) (2005)
15. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel -Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
16. Calvanese, D.: Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity. In: European Conference in Artificial Intelligence. (1996)
17. Bao, J., Honavar, V.: Divide and Conquer Semantic Web with Modular Ontologies - A Brief Review of Modular Ontology Language Formalisms. In: First International Workshop on Modular Ontologies (WoMo2006). (2006)
18. Bobillo, F., Delgado, M., Gómez-Romero, J.: A crisp representation for fuzzy $\mathcal{SHOIN}$ with fuzzy nominals and general concept inclusions. In: Proceedings of the 2nd Int. Workshop on Uncertainty Reasoning for the Semantic Web. (2006)