Representing the Component Library into Ontology Design Patterns

Aldo Gangemi[1] and Vinay K. Chaudhri[2]

[1] STLab, ISTC-CNR, Roma, Italy
aldo.gangemi@cnr.it
[2] SRI International, Menlo Park, US
vinay.chaudhri@sri.com

Abstract. For Ontology Design Patterns (OP) to be widely adopted by conceptual modelers, we need a critical amount of them, and that amount should be larger for patterns that describe good practices for modeling content (ie, Content Patterns or CP), e.g. about time, space, events, biological entitites, medical cases, legal norms, etc. It is possible and desirable to reuse existing repositories that contain modeling solutions, and to represent them as OPs. This paper analyzes some components from the Component Library (CLIB), proposing some solutions to represent them into OWL2 CPs. Some constructs from CLIB components need the expressivity of rule languages in order to fully represent them, but these extra-features can be separated from the basic ontological content. Additionally, CLIB components are shown to be enrichable with the pattern annotation schema defined for OPs, which also allows a quick upload and publication on ontologydesignpatterns.org.

1 Introduction

Ontology Design Patterns (OP) [20] are reusable solutions for modeling ontologies, based on good practices. In order to be widely adopted by ontology modelers, we need a large amount of them, especially for patterns that describe good practices for modeling content (Content Patterns, CP), either general (time, space, events, etc.) or specific (biological entities, medical cases, legal norms, etc.).

The ontologydesignpatterns.org initiative [19] aims to collect the OPs collaboratively, and several of them are being uploaded on its semantic wiki-based site. On the other hand, it is also possible to reuse existing repository of resources, which contain modeling solutions that can be represented as CPs. One of them is the Component Library (CLIB) [8, 10],[3] which contains hundreds of solutions, and explicitly builds on the idea of a *system of concepts* [8], as well as to that of *knowledge patterns* [10]. The original idea of the authors of CLIB [8] is indeed very close to that of CPs:

[3] http://www.cs.utexas.edu/ mfkb/RKF/tree/

... our goal is to identify repeated patterns of axioms in a large theory, and then abstract and reify those patterns as components in their own right (analogous to the notion of "design patterns" in object-oriented programming ...

... in contrast to [a] DL algorithm which exhaustively constructs concept representations without regard to task, our algorithm is goal-driven, constructing only those parts of the concept representation required to answer questions. Our trade-off is to sacrifice completeness for a language sufficiently expressive for our purposes. An interesting consequence of our approach is that the concept description which is built is question-specific, containing just that information required to answer the question(s) which were posed ...

In practice, the main requirements of CPs: small, task-based models that fit one or more competency questions by following good practices [16, 20], are shared by CLIB components. The main differences with current OWL CPs include the encoding of CLIB is done in the KM language. The statements in KM have straightforward and well-defined semantics in First-order logic [9]. KM components also represent dynamic aspects of actions the representation for which has not been studied in the context of OWL.

This paper analyzes some components from CLIB, showing how to represent them into OWL CPs. OWL2 [21] is employed in order to take advantage of the maximal expressivity currently available for the Semantic Web. Some constructs from CLIB components need the expressivity of rule languages in order to be fully represented. These extra-features can be separated from the basic ontological content, leaving intact the value of CLIB as a resource for CPs.

Additionally, CLIB components are shown to be enrichable with the pattern annotation schema defined for OPs, which also allows a quick upload and publication on the `ontologydesignpatterns.org` wiki.

The paper is organized as follows: in Section 2 we introduce CLIB and its main features; in Section 3 some uses of the CLIB are described; in Section 4 we represent some CLIB components as CPs on the Semantic Web; in Section 5 we present the publishing plans for CLIB components as CPs.

2 Component Library

The Component Library or CLIB was created with the goal of enabling users with little experience in knowledge engineering to represent knowledge from their domain of expertise by instantiating and composing generic components from a small, hierarchical library. Components are coherent collections of axioms that can be given an intuitive label, usually a common English word. The components should be general enough that their axiomatization is relatively uncontroversial. Composition consists of specifying relationships between instantiated components so that additional implications can be computed. The current CLIB contains a few hundred components, and less than one hundred relations.

Each component of CLIB is formally expressed in KM [9], which in turn is defined in first-order logic. KM includes a situation mechanism for representation and reasoning with actions and the changes they cause.

The main division in CLIB is between entities (things that are) and events (things that happen). Events are states and actions. States represent relatively static situations brought about or changed by actions.

2.1 Actions and States

The actions are grouped into fifteen top-level clusters, each having several more specific subclasses. These clusters are: Add, Remove, Communicate, Create, Break, Repair, Move, Transfer, Make-Contact, Break-Contact, Make-Accessible, Make-Inaccessible, Perceive, Shape, and Orient.

The list was developed by consulting linguistic resources such as WordNet [15], the defining vocabulary of the Longman's Dictionary of Contemporary English and the Roget's theasaurus. We consider here a few examples of the formal axioms that define the actions in the component library. (In the following, the words shown in all caps are the concepts drawn from the CLIB, and the words shown in italics are the relations drawn from the CLIB.)

Conditional rules: If the raw material of a PRODUCE is a SUBSTANCE, then the *product* is composed of that SUBSTANCE. If the *raw materials* are OBJECTs, then the *product* has those OBJECTs as *parts*.

Definitions: An instance of MOVE whose destination is *inside* a CONTAINER is automatically reclassified as an instance of ENTER.

Simulation: If the *destination* of a MOVE is a SPATIAL-ENTITY then the *location* of the OBJECT of the MOVE after the MOVE is that SPATIAL-ENTITY.

States are coherent collections of axioms that represent situations brought about or changed by actions. Many of the CLIB actions are defined in terms of the change in state they cause. This relationship between actions and states is made explicit in the library: there are actions that put objects into states, actions that take objects out of states and actions whose behavior is affected by objects being in states. For example, the BREAK action puts an object into a BE-BROKEN state. The REPAIR action takes an object in a BE-BROKEN state out of that State.

2.2 Entity and Roles

The entity hierarchy in CLIB is less developed than the hierarchy of actions. An important sub-division of entities contains role concepts. A role can be thought of as a temporally unstable entity. It is what an entity is in the context of some event. For example, PERSON is an entity while EMPLOYEE is a role. A PERSON remains a PERSON independent of the events in which she participates. Conversely, someone is an EMPLOYEE only by virtue of participation in an EMPLOYMENT event. A more detailed discussion on the representation of roles and the work related to them is available elsewhere [14].

2.3 Relations

The CLIB contains a small set of relations to connect Entities and Events. The design of the relations between events and entities was inspired by the case roles in linguistics [1], the design of relations between entities was based on the semantics of English noun phrases, and the choice of relationships between events followed from studies in discourse analysis, and process planning.

Examples of event-entity relations are: *agent, object, instrument, etc.* Examples of entity-to-entity relations are *content, has-part, location, material, etc.* Examples of event-to-event relationships are *causes, defeats, enables, prevents, etc.*

While the current CLIB contains domain and range constraints for all these relations, it does not yet contain their complete axiomatization. For example, the CLIB does not yet contain axioms for what it means for an action A to *prevent* another action B.

2.4 Properties

The CLIB has a small number of properties. Properties link entities to values. For example, the *size* of an entity is a property that takes a value. The value can be a cardinal (25 kilograms), a scalar (big relative to housecats) or a categorical (brown). The current CLIB has about 25 general categories. This final list of properties includes such properties as *age, area, capacity, color, length, shape, size, smell* and *wetness.*

3 Uses of the Component Library

The Component Library has been used in several projects, but most notably, in Vulcan's Project Halo (See `http://www.projecthalo.com`) , and DARPA's Project CALO (See `http://caloproject.sri.com`). More detailed description of these uses are available elsewhere [3, 4], but for the present paper, we only focus on its use in a system called AURA that has been developed under Vulcan's Project Halo [4].

The short-term goal of AURA is to enable domain experts to construct declarative knowledge bases (KBs) from a science textbook in the domains of Physics, Chemistry, and Biology in a way that it can answer questions similar to those in a college level exam. The overall concept of operation for AURA is as follows: a knowledge formulation engineer (KFE) with at least a graduate degree in the discipline of interest undergoes 20 hours of training to enter knowledge into AURA; a different person, called a question formulation engineer (QFE), with a high school level education undergoes 4 hours of training and asks questions of the system. Knowledge entry is inherently a skill-intensive task, and therefore, requires more advanced training in the subject as well as in using the system. A QFE is a potential user of the system, and the training requirement was kept lower because we wanted the barrier to using the system to be as low as possible.

For this section, we primarily focus on the knowledge formulation component of AURA because that highlights the use of CLIB more clearly.

The KFEs build their KBs by starting from CLIB. AURA implements a way to convert the axioms in CLIB into a graphical form, to allow a KFE to search for required components, and to graphically assemble them into a domain-specific representation. As a concrete example, we show how a KFE would represent the concept of Virus Infection in Figure 1.

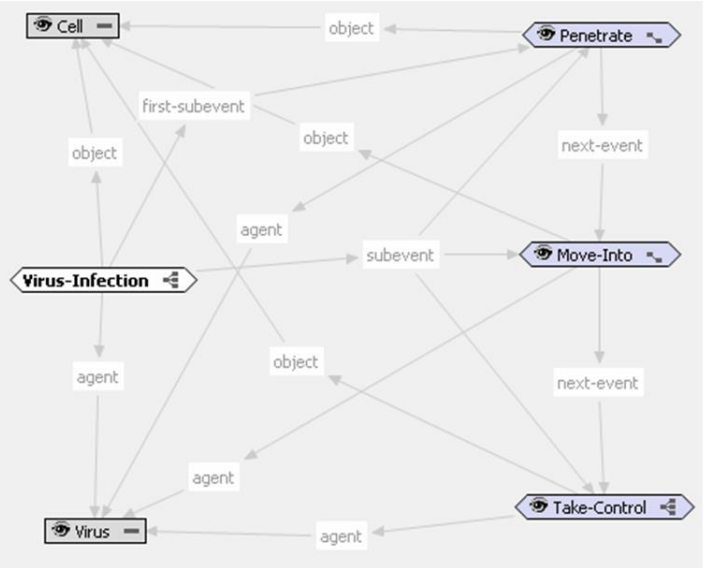

Fig. 1. The Representation of Virus Infection using the CLIB

In this Figure, a KFE has connected three generic CLIB actions: PENETRATE, MOVE-INTO and TAKE-CONTROL, together to first define a sequence amongst those events using the relation *next-event*, and *first-event*, and then specialized those events using relations to VIRUS and CELL.

AURA has undergone substantial testing in its ability to allow KFEs to formulate knowledge in Physics, Chemistry, and Biology showing the effectiveness of this approach for knowledge representation and acquisition [7].

4 Representing CLIB for the Semantic Web

Given the prior success in exploiting CLIB for knowledge representation and acquisition, the time is now ripe to broaden its usage especially in the semantic web community. Doing so will require at least the following steps. First, we need to represent the content of CLIB in a format that is widely used in the semantic web community. OWL is an obvious starting point, but we expect that for fully representing the content of CLIB a language more expressive than OWL will be needed. Second, we need to subject CLIB to community review so that its representations are generally agreed upon and accepted. Such an exercise is consistent with the original goal of CLIB to ensure that the component definitions are non-controversial and represent the consenus view on the definition of the concepts they represent. Such a goal is also consistent with the goal of the ontology patterns portal `ontologydesignpatterns.org`. Finally, we need to start constructing use cases that are of relevance to semantic web that demonstrate how the CLIB representations can be useful for modeling problems other than what it has been used for.

For the rest of the section, we focus on the problem of representing the content of CLIB using semantic web languages. We will first take an example concept from CLIB, and first explain its formal semantics as they are represented in its native representation language KM. Then we show the representation of the same knowledge using OWL, and a rule language SILK.

4.1 CLIB constructs and formal semantics

In order to exemplify what CLIB constructs can be translated into OWL, by preserving as much semantics as possible, we show here the KM axioms for the `Attach` component in CLIB. The `Attach` is an action that "*causes two things to be attached to each other*". We have included only inferentially significant axioms, and omitted the ones that are aimed at natural language generation, or for controlling how they are displayed to the user.

```
(Attach has
  (superclasses (Action))
  (required-slot (object base))
  (primary-slot (agent))
)

(every Attach has
  (object ((exactly 1 Tangible-Entity) (a Tangible-Entity)))
  (base ((exactly 1 Tangible-Entity) (a Tangible-Entity)))

  ;; SOFT PCS:
  (soft-pcs-list (
      (:triple (the base of Self)
                  object-of (mustnt-be-a Be-Inaccessible))))
```

```
  (resulting-state ((a Be-Attached-To)))
  (add-list ((:set
        (:triple (the resulting-state of Self)
            object (the object of Self)
            [Attach-add-1])
        (:triple (the resulting-state of Self)
            object (the base of Self)
            [Attach-add-2])))))

(every Attach has
  (preparatory-event ((:default
      (a Make-Contact with
            (object ((the object of Self)))
            (base   ((the base of Self))))
      (a Detach with
            (object ((the object of Self)))
            (base   ((the base of Self))))
))))
```

Informally, the KM code says that `Attach`:

- is subsumed by the more general component `Action`
- is always related to exactly one `object` and one `base`, both of type `TangibleEntity`
- can be related to an `agent`
- has a `resulting-state` of type `Be-Attached-To`, which has *coreferential* links to the `object` and `base` of `Attach`, and (operationally) generates these links when instantiated
- has two *defeasible* `preparatoryEvents` (`(preparatory-event ((:default)`: `MakeContact` and `Detach`, which have coreferential links to the `object` and `base` of `Attach`
- has a "soft constraint" (`soft-pcs-list`): the base of `Attach` (coreferential link) cannot be in a state of type: `BeInaccessible`

Although not explicit in the above code, `Action` inherits other properties from its subsuming components (`Action` and `Event`):

- is always related to at least one `instrument` of type `Entity`
- is always related to at least one `subevent` of type `Event` (including itself: a *non-proper* subevent relation)
- can be related to a `nextEvent` of type `Event`
- can be related to a `timeDuring` of type `TimeInterval`

Finally, it has properties that derive from hardcoded functionalities of KM. For example, the following code collects the subevents of any `Event` (then including `Attach` events) into a single list, which can then be simulated by KM (there are several such operational statements that apply to temporal, spatial, agentive, etc. aspects of events.[4]:

[4] The (operational) semantics of the `actions` slot is built into KM

```
(actions ((:set (forall (the subevent of Self)
                        (the actions of It))
                Self)))
```

The expressivity of KM largely exceeds OWL model-theoretical semantics. For example, as explained in 4.2, in OWL (either 1 or 2), coreferential links cannot be declared in general, but only in special cases (transitive properties, property chains). Also, defeasible axioms and soft constraints are not expressible in regular OWL.[5]

Table 1 shows some *reengineering patterns*, presented as correspondences between constructs that are used in CLIB, and their equivalents in OWL1 or OWL2.[6] More correspondences are exemplified in Section 4.2.

KM vs SW Semantics		
KM Construct	SWL	SWL Construct
`superclasses`	*OWL1, OWL2*	`SubClassOf`
`required-slot`	*OWL1, OWL2*	`SubClassOf ObjectSomeValuesFrom`
`primary-slot`	*OWL1, OWL2*	`SubClassOf ObjectMinCardinality 0`
`exactly` n	*OWL1, OWL2*	`ObjectExactCardinality` n
`exactly` n `A`	*OWL2*	`ObjectExactCardinality` n `A`
`superslots`	*OWL1, OWL2*	`SubObjectPropertyOf`
`domain`	*OWL1, OWL2*	`ObjectPropertyDomain`
`instance-of`	*OWL2*	`ClassAssertion`

Table 1. A list of some KM constructs used in CLIB, and their approximation as constructs from Semantic Web languages.

4.2 Representing CLIB using OWL

A translator to convert portions of CLIB into OWL is already available [6]. This translator has been used extensively as part of both the Halo and CALO projects. In the CALO project, the OWL ontology generated by this translator was used to integrate several diverse programming languages and reasoning modules [3]. In the AURA system, the OWL export of CLIB and the content authored by the KFEs was used to define a mapping between the CLIB and an ontology generated through Semantic Media Wiki [5]. But, as expected, this translation is incomplete. As a concrete example, we show below an OWL representation of the `Attach` component that we had shown earlier.

```
SubClassOf(Attach Action)
```

[5] There exist proposals and prototype implementatioms for extending OWL towards non-standard description logics.

[6] We present all OWL formulas in OWL2 Functional Syntax [18].

```
SubClassOf(Attach ObjectSomeValuesFrom(base Tangible-Entity))
SubClassOf(Attach ObjectSomeValuesFrom(object Tangible-Entity))
```

In this OWL representation, there is no representation for the slots that capture qualified cardinality restrictions, meta-level assertions, and the dynamic aspects of `Action`: for example, its `add-list` and `del-list` as from the `Attach` example above. OWL1 does not provide any support for those constructs. OWL2 has enough expressivity: for example, the following constructs can then be added, by means of more reengineering patterns that help finding correspondences to the semantics assumed in KM:

- qualified cardinality restrictions can be expressed natively in OWL2 (see table 1), e.g.
  `SubClassOf(Attach ObjectExactCardinality (1 base TangibleEntity))`;
- meta-level assertions like those used in CLIB for classifying slots (e.g. `(causes (instance-of (CausalRelation)))`, can be represented by means of OWL2 "punning", which provides different interpretations for the constants of an ontology. For example, `ClassAssertion` axioms can be asserted of classes, individuals, or properties without violating the formal semantics of OWL2, e.g. `ClassAssertion(causes CausalRelation)`;
- formal axioms for actions can be approximated by using OWL2 property chains. Conditional rules like the one presented in Section 2.1 can be schematized as follows:

  ```
  (if [A R1 B] then (forall [A R2 C] (:triple [It R3 B] [A-add-1])))
  ```

  and can be reengineered by firstly declaring some OWL object properties with appropriate domains and ranges for the antecedent part of the conditional rule:

  ```
  ObjectPropertyDomain(R1 A)
  ObjectPropertyDomain(R1 B)
  ObjectPropertyDomain(R2 A)
  ObjectPropertyDomain(R2 C)
  ObjectPropertyDomain(R3 C)
  ObjectPropertyDomain(R3 B)
  ```

  and then declaring an object property chain axiom:

  ```
  SubObjectPropertyOf(SubObjectPropertyChain(R2- R1) R3)
  ```

  Similarly, definitional rules can be represented by means of property chains, used in restrictions within equivalence axioms. The only rule type that seems completely outside OWL2 is simulation. In that case, it's the dynamics of the process that needs to be simulated in the language, not just represented, and this requires not only an `add-list`, but also a `del-list`, which could only be added programmatically to OWL;

– axioms including coreference are also very difficult to represent in OWL2 (coreference is prevented for complexity reasons). They can be partly represented by property chains, but the actual semantics has a substantial gap. For example, the `add-list` construct for the `Attach` component requires that the resulting state of `Attach` has the same object as the object and base of `Attach`, i.e. the tangible entities that result to be attached after the process. In OWL2 we can assert e.g.:

```
SubClassOf(Attach ObjectExactCardinality(1 base
   ObjectIntersectionOf(ObjectExactCardinality(1 baseOf
      BeAttachedTo) TangibleEntity)))
```

but the semantics does not catch the coreference, so that the BeAttachedTo states of `Attach` and of its base and object can be different. Therefore, given the complexity of this OWL2 construct, and its inability to catch the important part of the original axiom, it seems pretty inadequate to be incorporated into an OWL CP proposal;
– soft constraints and defeasible axioms are not covered by OWL2, but extensions of OWL exist that deal with probabilistic, possibilistic, and other varieties of soft reasoning. We have not yet decided if soft axioms should be provided in knowledge patterns as a general guideline, but we are exploring more evidence of its advantages, and community feedback. Defeasible axioms within universal axioms can be approximated by cutting the defeasible constraint on type, for example:
`SubClassOf(Attach ObjectSomeValuesFrom(preparatoryEvent Event))`
(since Event is the range of the slot `preparatoryEvent`).

As a wrap-up, after running all reengineering procedures described above, the `Attach` component gets represented in OWL2 as follows:

```
SubClassOf(Attach Action)
SubClassOf(Attach ObjectExactCardinality (1 base TangibleEntity))
SubClassOf(Attach ObjectExactCardinality (1 object TangibleEntity))
SubClassOf(Attach ObjectMinCardinality (0 agent Entity))
SubClassOf(Attach ObjectSomeValuesFrom(resultingState BeAttachedTo))
SubClassOf(Attach ObjectSomeValuesFrom(preparatoryEvent Event))
```

We do not include the inherited axioms from parent components (`Action` and `Event`).

We hope to investigate the use of OWL2 profiles[7] and OWL-based rule languages [17] to see how more of the information in CLIB could be represented.

4.3 Representing CLIB using Rule Languages

The taxonomic subset of CLIB can be captured to a great extent using the OWL family of languages. Property chains seem to help somehow to harvest

[7] `http://www.w3.org/TR/owl2-profiles/`

more. However, for a complete representation, a rule language is necessary. For example, the Virus Infection example shown earlier is represented in KM as follows:

```
(Virus-Infection has
   (subclass-of (Move)))

(every Virus-Infection has
    (agent ((a Virus)))
    (object ((a Cell)))
    (first-subevent ((the Penetrate sub-event of Self)))
    (subevent ((a Penetrate with
                 (agent ((the agent of Self)))
                 (object ((the object of Self))))
               (a Move-Into with
                 (agent ((the agent of Self)))
                 (object ((the object of Self))))
               (Take-Control with
                 (agent ((the agent of Self)))
                 (object ((the object of Self)))))))
```

Fully representing this axiom will require a rule language. We already have an effort underway to represent such rules using a new semantic web language called SILK (See `http://silk.semwebcentral.org`). SILK is a successor of SWSL and is more expressive than OWL. The above rule can be represented in SILK as follows:

```
?x[agent -> _#1(?x):Virus],
?x[object -> _#2(?x):Cell],
?x[subevent -> _#3(?x):Penetrate [next-event -> _#4,
                                  agent -> _#1,
                                  object -> _#2]]
               _#4(?x):Move-Into [next-event -> _#5,
                                  agent -> _#1,
                                  object -> _#2]]
               _#6(?x):Take-Control [agent -> _#1,
                                     object -> _#2]]
?x[first-subevent -> _#3(?x)]
:- ?x: Virus-Infection.
```

We believe that for fully representing the ontology design patterns in CLIB, an expressive representation language such as SILK is indispensable.

For the editing, manipulation and storage of the knowledge created by the KFEs, AURA uses a representation that is based on a network of individuals - also known as prototypes [9]. A prototype captures a rule whose antecedent is existentially quantified, and its consequent represents a network of existentially quantified individuals. We show the representation of the above rule using prototypes below:

```
(_Virus-Infection2117 has
  (prototype-scope (Virus-Infection))
  (prototype-participants (_Cell2168
                           _Virus2167
                           _Penetrate2166
                           _Move-Into2165
                           _Take-Control2164
                           _Virus-Infection2117))
  (object (_Cell2168))
  (agent (_Virus2167))
  (subevent (_Penetrate2166
             _Move-Into2165
             _Take-Control2164))
  (first-subevent (_Penetrate2166)))

(_Cell2168 has
  (instance-of (Cell)))

(_Virus2167 has
  (instance-of (Virus)))

(_Penetrate2166 has
  (object (_Cell2168))
  (agent (_Virus2167))
  (instance-of (Penetrate))
  (next-event (_Move-Into2165)))

(_Move-Into2165 has
  (object (_Cell2168))
  (agent (_Virus2167))
  (instance-of (Move-Into))
  (next-event (_Take-Control2164)))

(_Take-Control2164 has
  (object (_Cell2168))
  (agent (_Virus2167)))

(Virus-Infection has (superclasses (Move)))
```

It may be possible to represent prototype version of the above rule directly into OWL, but it would require further research to determine if the inferences that are expected of a prototype can also be supported in OWL directly. We are including the prototype representation in this paper to illustrate a possible approach of dealing with the expressiveness limitations of OWL.

It will also be interesting to explore the potential of integrating RIF[8] [2], also including its production rule dialect [13] in order to extend the ontology patterns representation with full-fledged expressivity, while remaining within the W3C standards.

5 Publishing CLIB-CP

We are planning to represent the generic fom CLIB and their most useful axioms into OWL2. We are also translating the metadata that annotate the components. They consist mainly of natural language processing-related code for generating friendly renderings of the components, and of some short comments that summarize the intentional content of each component, for example: "A conceal causes something to be concealed by something else.". However, in the `ontologydesignpatterns.org` initiative, we are interested in a rich annotation of patterns, which provides information about intent, consequences, related patterns, scenarios, competency questions, etc. In the automatic reengineering, we are translating the short comments as strings for the `Intent` value in the pattern annotation schema, and then we are adding the URIs of patterns mentioned within others as related ones. All the other metadata will be possibly introduced by engaging the open communities interested in reusing elements from our portal, or in asking for clarifications.

Several systems to discuss and evaluate patterns have been developed for the ODP portal [12]: they allow to automatically upload an OWL version of a pattern, to ask for reviews and public comments, to provide feedback and reviews, to discuss, to get consensus, and to recommend the patterns as best practices eventually. For the CLIB-based CPs, a script is being implemented to translate and import all at once, but adjusting some parameters.

Once we have a good fraction of CLIB published as ontology patterns on our portal, we will use the review mechanism of the portal to evalute the ontology patterns. For example, each component that is represented as an ontology pattern will be reviewed from the point of view of completeness, accuracy, and to what extent it represents the consensus meaning of that component. The review process will be moderated by a scientific committee overseen by an ODP editorial board.

6 Conclusions

The representation work on CLIB is related to the larger initiative of populating an online, collaboratively-maintained library of ontology design patterns. Since reuse of content patterns requires a critical amount of them, the effort concentrates on one hand on building the community that submits, discusses, and validates pattern proposals, and on the other hand on representing existing resources that have substantial affinity with content patterns. Representation

[8] `http://www.w3.org/TR/rif-rdf-owl/`

practices have been devised until now on FrameNet [11] and the CLIB, which is the contribution of this paper. CLIB is probably the most developed repository of *knowledge* patterns, and has a scope that goes beyond that of content ontology patterns in trying to represent also event dynamics, simulation, etc. However, the part translatable to OWL and rule languages like RIF or SILK proves to be very inline with the ODP initiative, and will constitute an important inventory and a rock-solid resource for the community.

Acknowledgements

This work has been partly supported by the EU projects NeOn, funded within the 6th IST Framework Programme, and IKS, funded within the 7th IST Framework Programme, and by Vulcan Inc. as part of their Project Halo.

References

1. K. Barker, T. Copeck, S.Delisle, and S. Szpakowicz. Systematic Construction of a Versatile Case System. *Journal of Natural Language Engineering*, 3(4):279–315, 1997.
2. H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds. RIF Core. W3C Working Draft 18 December 2008. http://www.w3.org/TR/2008/WD-rif-core-20081218/, 2008.
3. V. K. Chaudhri, A. Cheyer, R. Guili, B. Jarrold, K. Myers, and J. Niekarasz. A case study in engineering a knowledge base for an intelligent personal assistant. In *International Workshop on Semantic Desktops held during ISWC-2006*, 2006.
4. V. K. Chaudhri, P. E. Clark, S. Mishra, J. Pacheco, A. Spaulding, and J. Tien. AURA: Capturing Knowledge and Answering Questions on Science Textbooks. Technical Report, SRI International, 2009.
5. V. K. Chaudhri, M. Greaves, D. Hansch, A. Jameson, and F. Pfisterer. Using a semantic wiki as a knowledge source for question answering. In *AAAI Spring Symposium on Symbiosis of Semantic Web and Knowledge Engineering*, Stanford, CA, 2008.
6. V. K. Chaudhri, B. Jarrold, and J. Pacheco. Exporting knowledge bases into OWL. In *OWL: Experiences and Directions*, Athens, Georgia, 2006.
7. V. K. Chaudhri, B. John, S. Mishra, J. Pacheco, B. Porter, and A. Spaulding. Enabling experts to build knowledge bases from science textbooks. In *Proceedings of the International Conference on Knowledge Capture Systems*, 2007.
8. P. Clark and B. Porter. Building concept representations from reusable components. In *Proceedings of AAAI'97*, pages 369–376. AAAI press, 1997.
9. P. Clark and B. Porter. Km – the knowledge machine: Users manual. Technical report, University of Texas at Austin, 1999.
10. P. Clark, J. Thompson, and B. Porter. Knowledge Patterns. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 591–600, San Francisco, 2000. Morgan Kaufmann.
11. B. Coppola, A. Gangemi, A. Gliozzo, D. Picca, and V. Presutti. Frame Detection over the Semantic Web. In *Proceedings of the Fifth European Semantic Web Conference*. Springer, 2009.

12. E. Daga, V. Presutti, and A. Salvati. http: //ontologydesignpatterns.org and evaluation wikiflow. In A. Gangemi, J. Keizer, V. Presutti, and H. Stoermer, editors, *SWAP*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
13. C. de Sainte Marie, A. Paschke, and G. Hallmark. RIF Production Rule Dialect. W3C Working Draft 18 December 2008. http://www.w3.org/TR/2008/WD-rif-prd-20081218/, 2008.
14. J. Fan, K. Barker, B. Porter, and P. Clark. Representing roles and purpose. In *Proceedings of the International Conference on Knowledge Capture Systems*, 2001.
15. C. Fellbaum, editor. *WordNet. An Electronic Lexical Database*. MIT Press, 1998.
16. A. Gangemi. Ontology design patterns for semantic web content. In *Proceedings of ISWC 2005, Galway, Ireland*, Berlin, 2005. Springer.
17. I. Horrocks and P. F. Patel-Schneider. A proposal for an owl rules language. In *Proceedings of the 13th International Conference on World Wide Web*, pages 723–731, New York, NY, 2004. ACM Press.
18. B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft 08 October 2008. http://www.w3.org/TR/2008/WD-owl2-syntax-20081008/, 2008.
19. Ontology Design Patterns Web Portal. http://www.ontologydesignpatterns.org.
20. V. Presutti and A. Gangemi. Content ontology design patterns as practical building blocks for web ontologies. In *Proceedings of the 27th International Conference on Conceptual Modeling - ER2008*, pages 128–141, 2008.
21. W3C OWL Working Group. OWL 2 Web Ontology Language. W3C Working Draft 11 June 2009, http://www.w3.org/TR/2009/WD-owl2-overview-20090611/, 2009.