Ontology Overview Motorola Labs

Myriam Ribière
Patricia Charlton
Networking and Applications Lab
Centre de Recherche de Motorola-Paris
Espace Technologique – Commune de Saint Aubin
Email: myriam.ribiere@crm.mot.com
Email: patricia.charlton@crm.mot.com

Content

1 Existing languages for ontology representation....3
1.1 KIF....3
1.2 OKBC....5
1.3 XML....5
1.4 RDF(S)....5
1.5 XML-based languages: XOL....8
1.6 OIL....9
1.7 DAML + OIL:....11
1.8 Conclusion about languages....13
2 Comparison on existing ontology developments....15
2.1 Protégé-2000....15
2.1.1 Presentation....15
2.1.2 Knowledge model....17
2.1.3 Buiding ontologies and general use of the tool....18
2.1.4 Cooperation....18
2.1.5 Evaluation of the API....18
2.2 OntoEdit....19
2.2.1 Knowledge model....19
2.2.2 Building ontologies and general use of the tool....21
2.2.3 Cooperation....21
2.2.4 Evaluation of the API....21

1 Existing languages for ontology representation

1.1 KIF

KIF (Knowledge Interchange format) [12] was one of the first knowledge representation language. It is a computer-oriented language for the interchange of knowledge among disparate programs. It has declarative semantics (i.e. the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions); it is logically comprehensive (i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides for the representation of knowledge about the representation of knowledge; it provides for the representation of non monotonic reasoning rules; and it provides for the definition of objects, functions, and relations. This language was defined around the Ontolingua tool that provides a cooperative ontology builder and server (http://www.ksl.stanford.edu/software/ontolingua/).

KIF representation with the frame-based meta ontology proposed on the Ontolingua server means that the frame-based representation elements such as classes, slots, facets and instances are described using KIF and the domain knowledge is defined with the frame-based representation. Since the Frame Ontology is less expressive than KIF, ontolingua allows to include KIF expressions inside of definitions based on the Frame-Ontology. So the Ontolingua language allows building ontologies in the following three manners: 1) Using exclusively the frame Ontology vocabulary, 2) using KIF expressions; 3) using both languages simultaneously.
We used the ontolingua server (http://www-ksl-svc.stanford.edu:5915/) to produce this example:

Declaration of the ontolingua server

```
(In-Package "ONTOLINGUA-USER")

Section for the meta information about the ontology
(Define-Ontology
     Servicedescription
     (Frame-Ontology)
   "Not supplied yet."
   :Referenced-Ontologies
   (Agents)
   :Io-Package
   "ONTOLINGUA-USER"
   :Intern-In
   ((Agents Person)))


(In-Ontology (Quote Servicedescription))
```

Section for the declaration of concepts like classes

```
(Define-Class Business_Operation
              (?X)
              "Declaration of the business Operation concept"
              :Def
              (And (Operation ?X)))

(Define-Class Business_Process
              (?X)
              "Not supplied yet."
              :Def
              (And (Process ?X)))

(Define-Class Communication_Agent
              (?X)
              "Not supplied yet."
              :Def
              (And (Thing ?X)))

 (Define-Class Manufacturing_Operation
              (?X)
```

```
             "Not supplied yet."
             :Def
             (And (Operation ?X)))

(Define-Class Manufacturing_Process
             (?X)
             "Not supplied yet."
             :Def
             (And (Process ?X)))

(Define-Class Operation
             (?X)
             "Define the class of all operations"
             :Def
             (And (Thing ?X)))

(Define-Class Process
             (?X)
             "Define the class of all processes"
             :Def
             (And (Thing ?X)))

(Define-Class Serviceprovider
             (?X)
             "Define the class of all service providers"
             :Def
             (And (Thing ?X)))

(Define-Class Transportation_Operation
             (?X)
             "Not supplied yet."
             :Def
             (And (Operation ?X)))
```

Section for the declaration of relations and properties like slots

```
(Define-Relation Has_Capability
                 (?Frame ?Value)
                 "Not supplied yet."
                 :Def
                 (And (Serviceprovider ?Frame) (Process ?Value)))

(Define-Relation Internet_Service_Contact
                 (?Frame ?Value)
                 "Not supplied yet."
                 :Def
                 (And (Serviceprovider ?Frame) (Communication_Agent ?Value)))

(Define-Relation Personal_Contact_Information
                 (?Frame ?Value)
                 "Not supplied yet."
                 :Def
                 (And (Serviceprovider ?Frame) (Person@Agents ?Value)))

(Define-Relation Provide_Service
                 (?Frame ?Value)
                 "Not supplied yet."
                 :Def
                 (And (Serviceprovider ?Frame) (Operation ?Value)))

(Define-Relation Serviceprovideridentification
                 (?Frame ?Value)
                 "Identify the Service provider"
                 :Def
                 (And (Serviceprovider ?Frame) (String ?Value)))
```

Axioms are represented like lisp-like expressions and can express quantifiers and basic logic operators:

- Existence: `(exists(?x))`
- Universal: `(Forall (?x))`

- Conjunction `(And ?x ?y)`
- Disjonction `(Or ?x ?y)`
- Negation `(Not ?x)`

There are available operators for defining expressions, sentences and objects are available (see [12]). KIF is not a language of representation, it is an interchange format, hence, classic knowledge representation language were implemented into KIF such as frame-based representation in KIF. When the knowledge model of the representation language does not allow to represent axioms or rules, the choice is made to mix the representation and use a theorem prover based on KIF to perform inferences. The HPKB (high Performance Knowledge Base) project at SRI (Stanford Research Institute) has used this solution with OKBC and SNARK (http://www.ai.sri.com/~stickel/snark.html).

1.2 OKBC

Several frame-based languages were proposed such as Ontolingua, Ocelot, LOOM, etc.; **OKBC** (Open Knowledge base Connectivity) provides a uniform model of Knowledge Representation Systems (KRSs) based on a common conceptualisation of classes, individuals, slots, facets, and inheritance. The GFP (Generic Frame Protocol) Knowledge Model is the implicit representation formalism underlying OKBC. OKBC is defined in a programming language independent fashion, and has existing implementations in Common Lisp, Java, and C. The protocol transparently supports networked as well as direct access to KRSs and knowledge bases.

FIPA ontology specification uses the OKBC representation. For representation of axioms and rules for agents and also for the declaration of some meta-ontologies, the OKBC representation in any OKBC compliant language is not sufficient.

1.3 XML

XML (extensible Markup Language) is a meta-language designed and developed by the XML working group of the W3C. It derives from SGML (Standard General Markup Language). XML allows users to define their own tags and attributes, define data structure, and extract data from documents.
The advantages of XML are the followings:

- *Extensible*: You can define any languages by just defining a DTD.
- *Simple*: XML documents are human readable and easy to understand/create.
- *Separation of syntax and semantics*: XML defines rules for well-formed documents; the semantics depend on the application that processes the document.
- *Separation of content and presentation*: XML does not imply the way of visualizing the information. This separation gives the possibility of applying different visual presentations to the same XML content.
- *Distribute knowledge over WWW*: It can be used to represent distributed knowledge across several web-pages, as it can be embedded in them.

The introduction of tags and attributes on web-pages help in adding semantics but it doesn’t provide a way to represent an entire ontology and to perform inferences. As it has some advantages, several languages were derived from XML to represent ontologies. The first one RDF(S) was propose by the Web consortium itself to help in introducing meta-information on web-pages.

1.4 RDF(S)

RDF was designed to describe metadata for the resources on the web in means of “statements”, “resources” and “properties”. Statements in RDF describe resources that can be web pages or real world object like publications, persons or institutions. Resources and properties are described with RDFS (RDF Schema). RDF Schema further extends RDF by adding more modelling primitives often found in ontology languages like classes, class inheritance, property inheritance, domain, range restriction.

RDF Schema enables the representation of classes, slots and facets and RDF allows the representation of instances and facts. There is not possibility to represent axioms in such language and the only inference mechanism is based on the subsumption relation existing between classes. The advantage of RDF is the use of XML namespace and URI to identify entities already defined on the WWW. This implies the followings possibilities:

- Statements could refer to different ontologies distributed on the WWW.
- An ontology could be refined according to an existing ontology available on the WWW.

It helps in sharing knowledge trough the Web and reuse knowledge to define new ontologies.
Figure 1 describes the knowledge model of RDF Schema, it helps to understand the possibilities in term of knowledge representation enabled by RDFS:

- Constraint properties or facets available are: range, domain
- Predefined slots on classes are: type, subclassOf, subPropertyOf, seeAlso, isDefinedBy, comments, labels.

Here the same example is used and represented in RDF and RDF Schema.

Header of the RDF file:

```
<rdf:RDF xml:lang="en"
        xmlns : rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns: rdfs = http://www.w3.org/2000/01/rdf-schema#>
```

An RDF file must declare namespace to describe statements and description of resources.

Declaration of the concept ServiceProvider and Operation as a subclass of the built-in upper class RESOURCE:

```
<rdf:Description ID="ServiceProvider">
        <rdf:type resource=htMtipr:i//www.w3.aomrg/2000/01/rdf-schema#Class/>
        <rdfs:subClassOf rdf:resource=http://www.w3.org/2000/01/rdf-schema#Resource/>
</rdf:Description>
<rdf:Description ID="Operation">
        <rdf:type resource=http://www.w3.org/2000/01/rdf-schema#Class/>
        <rdfs:subClassOf rdf:resource=http://www.w3.org/2000/01/rdf-schema#Resource/>
</rdf:Description>
```

Declaration of the concept Business_Operation, Transportation_Operation and Manufacturing_Operation as subclass of Operation:

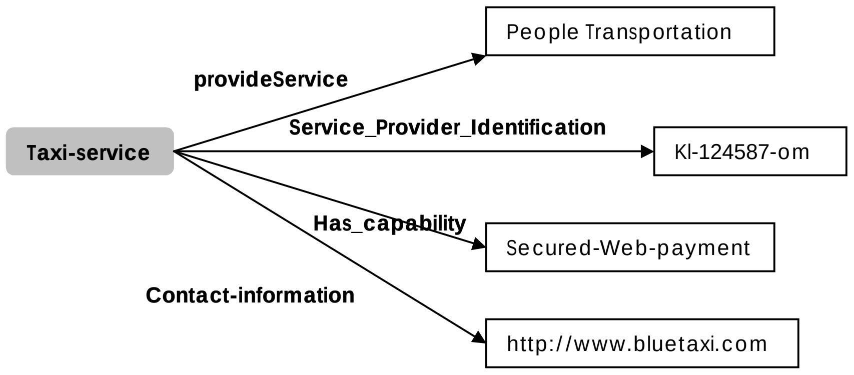
```
<rdf:Description ID="Business_Operation">
        <rdf:type resource=http://www.w3.org/2000/01/rdf-schema#Class/>
        <rdfs:subClassOf rdf:resource="#Operation"/>
</rdf:Description>
<rdf:Description ID="Transportation_Operation">
        <rdf:type resource=http://www.w3.org/2000/01/rdf-schema#Class/>
        <rdfs:subClassOf rdf:resource="#Operation"/>
</rdf:Description>
<rdf:Description ID="Manufacturing_Operation">
        <rdf:type resource=http://www.w3.org/2000/01/rdf-schema#Class/>
        <rdfs:subClassOf rdf:resource="#Operation"/>
</rdf:Description>
```

The declaration of properties on ServiceProvider class:

```
<rdf:Description ID=”provide_service”>
       <rdf:type resource=http://www.w3.org/1999/02/22-rdf-syntax-ns#property/>
       <rdfs:domain rdf:resource=”#Service_Provider”/>
       <rdfs:range rdf:resource=”http://www.myhome.com/Ontology_Operation#Operation”/>
</rdf:Description>
<rdf:Description ID=”service_provider_Identification”>
       <rdf:type resource=http://www.w3.org/1999/02/22-rdf-syntax-ns#property/>
       <rdfs:domain rdf:resource=”#Service_Provider”/>
       <rdfs:range rdfs:Literal”/>
</rdf:Description>
```

```
<rdf:Description ID=”has_capability”>
       <rdf:type resource=http://www.w3.org/1999/02/22-rdf-syntax-ns#property/>
       <rdfs:domain rdf:resource=”#Service_Provider”/>
       <rdfs:range rdf: resource=”http://www.myhome.com/Ontology_Process#Process”/>
</rdf:Description>
<rdf:Description ID=”has_manufacturing_capability”>
       <rdf:type resource=http://www.w3.org/1999/02/22-rdf-syntax-ns#property/>
       <rdfs:subPropertyOf rdf:resource=”#has_capability”/>
</rdf:Description>
```

This is an example of statement/instance to be represented:



The following statement could be expressed in RDF as:

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#>
       <rdf:Description ID=”TaxiService”>
              <rdf:type rdf:ID=”#Service_Provider”/>
              <provide_service rdf:resource=”#People_Transportation”/>
              <Service_Provider_Identification>kl-124587-
om</Service_Provider_Identification >
              <has_capability rdf:resource=”#Secured-Web-Payment”/>
              <Contact_information>http://wwww.bluetaxi.com</Contact_information>
       </rdf:description>
       <rdf:Description ID=”People_Tranportation”>
              <rdf:type rdf:ID=”#Tranportation_Operation”/>
       </rdf:description>
       <rdf:Description ID=” Secured-Web-Payment”>
              <rdf:type rdf:ID=”#Business_Process”/>
       </rdf:description>
</rdf:RDF>
```

1.5 XML-based languages: XOL

XOL was developed by Peter Karp of Pangea Systems, Vinay Chaudhri of SRI International, and Jerome Thomere of SRI International (http://www.ai.sri.com/~pkarp/xol/xol.html)

XOL stands for XML-based Ontology exchange Language. It was designed for the exchange of bioinformatics ontologies but it could be applied for different domains. The language was dedicated to the exchange of ontology definitions among different systems like database systems, ontology development tools or application programs. The definition of XOL comes from two important needs:

- the need for a language with the semantics of object-oriented knowledge representation systems: use a subset of OKBC called OKBC-lite.
- the need of an XML syntax for the interoperability and parser facility.

The subset of OKBC used covers only classes, slots and facets; frames are excluded from this language so it's impossible to define meta-ontologies through this language. We cannot produce axioms or rules with this language.

Every XOL document has the followings parts:

- Module section
- Class section
- Slot section
- Individual section

Description of the service example:

Header of the file with the declaration of meta-dat about the ontology:

```
<module>
       <name>Service description Ontology</name>
       <kb-type> some famous KBS </kb-type>
       <version>1.0</version>
       <documentation>Ontology for description of services</documentation>
</module>
```

Class section:

```
<class>
       <name>Service_Provider</name>
       <documentation>The class of all service providers</documentation>
       <subclass-of>THING</subclass-of>
</class>
<class>
       <name>Operation</name>
       <documentation>The class of all operations</documentation>
       <subclass-of>Thing</subclass-of>
</class>
<class>
       <name>Business_Operation</name>
       <documentation>The class of all business operations</documentation>
       <subclass-of>Operation</subclass-of>
</class>
<class>
       <name>Transportation_Operation</name>
       <documentation>The class of all transportation operations</documentation>
       <subclass-of>Operation</subclass-of>
</class>
```

Slot section:

```
<slot type = “own”>
       <name>Service_provider_Identification</name>
       <documentation>Identification of the service, this slot must be
unique</documentation>
       <domain>Service_Provider</domain>
       <slot-value-type>String</slot-value-type>
       <slot-cardinality>1</slot-cardinality>
</slot>

<slot type = “template”>
       <name>provide_service</name>
       <documentation>identify the service provided</documentation>
       <domain>Service_Provider</domain>
       <slot-value-type>Operation</slot-value-type>
       <slot-cardinality>1</slot-cardinality>
</slot>
```

The individual section:

```
<individual>
       <name>People Transportation</name>
       <documentation>Service of transportation</documentation>
       <instance-of>Tranportation</instance-of>
</individual>

<individual>
       <name> Secured-Web-payment</name>
       <documentation>Secured-Web-payment is a Business Process</documentation>
       <instance-of>Business_Process</instance-of>
</individual>

<individual>
       <name></name>
       <documentation>TaxiService</documentation>
       <instance-of>Service_Provider</instance-of>

       <slot-values>
              <name>provide_service</Name>
              <value>People_Transportation</value>
       </slot-values>
       <slot-values>
              <name>Service_Provider_identification</Name>
              <value>kl-124587-om</value>
       </slot-values>
       <slot-values>
              <name>has_capability</Name>
              <value>Secure-Web-Payement</value>
       </slot-values>
       <slot-values>
              <name>Contact_information</Name>
              <value>http://wwww.bluetaxi.com</value>
       </slot-values>
</individual>
```

1.6 OIL

OIL stands for Ontology Inference Layer (http://www.ontoknowledge.org/oil/index.shtml). It’s a proposal for a joint standard for specifying and exchanging ontologies. Its definition is based on existing frame-based language such as OKBC, XOL and RDF. The project is sponsored by the European Community via the IST projects Ibrow (An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web, http://www.swi.psy.uva.nl/projects/ibrow/home.html) and On-to-knowledge (http://www.ontoknowledge.org/).

“Oil unifies three important aspects provided by different communities: Formal semantics and efficient reasoning support as provided by description logics, epistemological rich modelling primitives as provided by Frame community and a standard proposal for syntactical exchange notations as provided by the Web community” [5], [6]. In this language, an ontology is described in three layers:

- Object level where concrete instances of ontology are described, not yet available.
- First meta level where the ontology can be defined.
- The second meta level or ontology container where meta-information on the ontology are described like author name, subject etc.

This language (see http://www.ontoknowledge.org/oil/syntax/Standard-OIL/ for the complete description of the standard) proposes a description of an ontology into the basic elements of a frame-based description: classes, slots, facets and offers the possibility to declare predefined generic axioms like disjointed classes, covered classes, disjointed covered classes and equivalent classes. It also enables to add particular properties on slots such as transitive, symmetric or functional properties.

First the container with the ontology produced by the Dublin Core Metadata Initiative (http://dublincore.org/documents/2001/04/12/usageguide/#whatismetadata) is described. It helps to describe the ontology content and intellectual property.

```
<ontology-container>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
         xmlns:dc=http://purl.oclc.org/dc#>
      <rdf:Description about="">
            <dc:Title>Service Description Ontology</dc:Title>
            <dc:Creator>RIOS</dc:Creator>
            <dc:Subject>Service providers, processes and
Operations</dc:Subject>
            <dc:description>Ontology for the description of
services</dc:Description>
            <dc:Type>Ontology</dc:Type>
            <dc:Format>Text</dc:Format>
            <dc:language>OIL</dc:Language>
      </rdf:Description>
</rdf:RDF>
</ontology-containers>
```

Then the first meta level by defining the service ontology

```
<ontology-definitions>
      <slot-def>
            <slot name="provide_service"/>
            <documentation>the service provide by the service
provider</documentation>
            <domain>
                  <class name="Service_provider"/>
            </domain>
      </slot-def>
      <slot-def>
            <slot name="Service_provider-identification"/>
            <documentation>Identification of the service
provider</documentation>
            <domain>
                  <class name="Service_provider"/>
            </domain>
      </slot-def>
      <slot-def>
            <slot name="has_capability"/>
            <documentation>The capability in term of process of the service
            provider </documentation>
            <domain>
                  <class name="Service_provider"/>
            </domain>
      </slot-def>
      <class-def>
            <class name="Service_Provider"/>
                  <documentation>The class of all service
provider</documentation>
      </class-def>
      <class-def>
            <class name="Operation"/>
                  <documentation> The class of all operations</documentation>
            </subclass-of>
      </class-def>
      <class-def>
            <class name="Transportation"/>
```

```
                    <documentation> The class of all
transportation</documentation>
             <subclass-of>
                          <class name="Operation">
             </subclass-of>
      </class-def>

      <class-def>
                    <class name="Woman"/>
                    <documentation> The class of all female
humans</documentation>
                    <subclass-of>
                    <AND>
                          <class name="person">
                          <NOT>
                                 <class name="Man"/>
                    </NOT>
             </AND>
             <subclass-of>
      </class-def>

      <Class-def>
             <class "ServiceProvider">
             <documentation> The class of all Service Provider</documentation>
                <subclass-Of>
                           <slot-constraint>
                                  <slot name="provide_Service"/>
                                  <value-type>
                                         <class name="Operation"/>
                                  </value-type>
                                  <max-cardinality>
                                         <number>1</number>
                                         <class name="Operation"/>
                                  </max-cardinality>
                           </slot-constraint>
                           <slot-constraint>
                                  <slot name="Contact_information"/>
                                  <max-cardinality>
                                         <number>1</number>
                                         <class name="Operation"/>
                                  </max-cardinality>
                                  <min-cardinality>
                                         <number>1</number>
                                         <class name="Operation"/>
                                  </min-cardinality>
                           </slot-constraint>
                </subclass-Of>
        </Class-def>
</ontology-definitions>
```

1.7 DAML + OIL:

The DAML project (Darpa Agent Markup Language) proposed a first release of an ontology language called DAML-Ont. After discussions about differencies between this language and the OIL proposition, the two project merged to propose DAML + OIL: a language based on RDF and RDF Schema with richer modelling primitives. DAML+OIL provides modelling primitives commonly found in frame-based languages. The language has a clean and well defined semantics based on description logics.
A DAML+OIL ontology consists of zero or more headers, followed by zero or more class elements, property elements, axioms and instances.
It has the advantages of the RDF model such as URI, XML for interchange and the expressivity of the OIL proposition.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
xmlns:op="http://cim4.ie.psu.edu:12/daml/RIOS/2001/05/RIOS_Operation#"
xmlns:="http://cim4.ie.psu.edu:12/daml/RIOS/2001/05/ServiceDescription#">
        <daml:Ontology about="">
             <daml:versionInfo>$Id: Operation_service, v 0.1 2001/05/01 09:50:40
      K. Boonserm $</daml:versionInfo>
                <daml:imports
rdf:resource="http://www.daml.org/2001/03/daml+oil#"/>
                <daml:imports
rdf:resource="http://cim4.ie.psu.edu:12/daml/rios/2001/05/RIOS_Operation#"/>
                <daml:imports
rdf:resource="http://cim4.ie.psu.edu:12/daml/rios/2001/05/RIOS_Process#"/>
                <!--daml:imports
rdf:resource="http://orl01.drc.com/daml/Ontology/POC/1.X/POC-ont-1.3v.daml#"/-->
                <daml:imports
rdf:resource="http://www.cs.yale.edu/homes/dvm/daml/agent-ont.daml#"/>
        </daml:Ontology>
        <daml:Class rdf:ID="ServiceProvider">
                <rdfs:subClassOf>
                        <daml:Restriction daml:minCardinality="1">
                                <daml:onProperty
rdf:resource="#providesService"/>
                                <daml:onProperty
rdf:resource="#contactInformation"/>
                        </daml:Restriction>
                </rdfs:subClassOf>
        </daml:Class>
        <daml:UniqueProperty rdf:ID="serviceProviderIdentification">
                <rdfs:domain rdf:resource="#ServiceProvider"/>
        </daml:UniqueProperty>
        <daml:ObjectProperty rdf:ID="providesService">
                <rdfs:domain rdf:resource="#ServiceProvider"/>
                <rdfs:range
rdf:resource="http://cim4.ie.psu.edu:12/daml/rios/2001/05/RIOS_Operation#Operatio
n"/>
        </daml:ObjectProperty>
        <daml:ObjectProperty rdf:ID="hasCapability">
                <rdfs:domain rdf:resource="#ServiceProvider"/>
        </daml:ObjectProperty>
        <daml:ObjectProperty rdf:ID="hasManufacturingCapability">
                <rdfs:subPropertyOf rdf:resource="hasCapability"/>
                <rdfs:range
rdf:resource="http://cim4.ie.psu.edu:12/daml/rios/2001/05/RIOS_Process#Manufactur
ing_process"/>
        </daml:ObjectProperty>
        <daml:ObjectProperty rdf:ID="contactInformation"/>
        <daml:ObjectProperty rdf:ID="personalContactInformation">
                <rdfs:subPropertyOf rdf:resource="#contactInformation"/>
                <rdfs:domain rdf:resource="#ServiceProvider"/>
                <rdfs:range
rdf:resource="http://orl01.drc.com/daml/Ontology/POC/1.X/POC-ont-1.3v#Person"/>
        </daml:ObjectProperty>
        <daml:Class rdf:about="http://orl01.drc.com/daml/Ontology/POC/1.X/POC-
ont-1.3v#Person">
                <daml:sameClassAs
rdf:resource="http://www.cs.yale.edu/homes/dvm/daml/agent-ont#HumanAgent"/>
        </daml:Class>
        <daml:ObjectProperty rdf:ID="internetServiceContact">
                <daml:subPropertyOf rdf:resource="#contactInformation"/>
                <rdfs:domain rdf:resource="#ServiceProvider"/>
                <rdfs:range
rdf:resource="http://www.cs.yale.edu/homes/dvm/daml/agent-ont#CompAgent"/>
        </daml:ObjectProperty>
</rdf:RDF>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:="http://cim4.ie.psu.edu:12/daml/rios/2001/05/RIOS_Operation#">
             <daml:Ontology rdf:about="">
                <daml:versionInfo>$Id: RIOS_Operation, v 0.1 2001/05/01 09:50:40
K. Boonserm $</daml:versionInfo>
                <rdfs:comment>Operation taxonomy</rdfs:comment>
                <daml:imports
rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
```

```
             </daml:Ontology>
             <daml:Class rdf:ID="Operation">
                <rdfs:label>Operation</rdfs:label>
                <rdfs:comment>Any operation</rdfs:comment>
             </daml:Class>
             <daml:Class rdf:ID="Business_operation">
                <rdfs:label>Business operation</rdfs:label>
                <rdfs:subClassOf rdf:resource="#Operation"/>
             </daml:Class>
             <daml:Class rdf:ID="Transportation_operation">
                <rdfs:label>Transportation operation</rdfs:label>
                <rdfs:subClassOf rdf:resource="#Operation"/>
             </daml:Class>
</rdf:RDF>
```

Drawback of such language is that it's become more and more complex to read and so it's more and more difficult to interpret and know if an ontology can be reused or not.

1.8 Conclusion about languages

The KIF interchange format could allow us to implement any language and add KIF sentences to generate axioms. XML-based languages like RDF seems to be interesting because it allow to share ontologies on the web by using URI and namespace but it is not expressive enough. RDF-based languages like DAML+OIL are interesting because of the rich expressiveness to represent concepts and their relationships and also most common used axioms. The drawback of this language is just the readability of this language.
Figure 3 presents the first evaluation of languages and their possibilities in term of expressivity

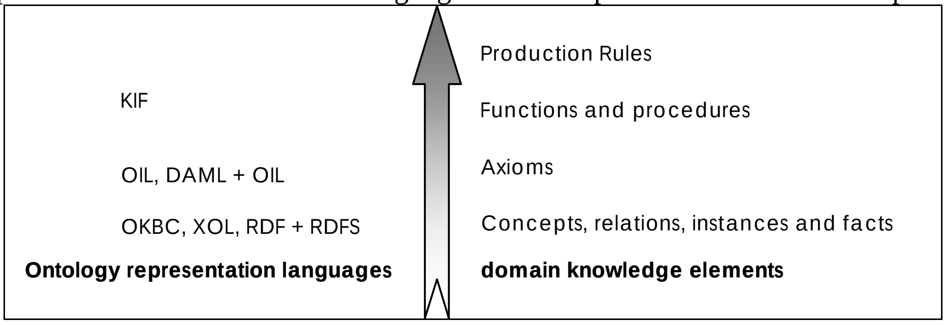

Production Rules
KIF
Functions and procedures
Axioms
OIL, DAML + OIL
OKBC, XOL, RDF + RDFS
Concepts, relations, instances and facts
Ontology representation languages
domain knowledge elements

Fig. 3: Degree of expressivity for the different languages or formats

The comparison table lists elements fundamental to be represented and how the different languages support them.

	KIF	OKBC	XOL	RDF(S)	OIL	OIL + DAML
Main elements						
Concepts	+	+	+	+	+	+
Relations	+	+	+	+	+	+
Functions	+	-	-	-	+	+
Instances	+	+	+	+	-	+
Axioms	+	-	-	-	+	+
Axiom possibilities						
Negation	+	-	-	-	+	+
Conjonction	+	-	-	-	+	+
Disjonction	+	-	-	-	+	+
Concept Taxonomy						
Subclass-of	+	+	+	+	+	+
Multiple-inheritance	+	+	+	+	-	+
Meta-classes	+	+	-	+	-	+
Slots						
Multi-valued slots	+	+	+	+	-	+
Slot hierarchy (subslot-of)	+	+	-	+	-	+
Slot-inverse	+	+	+	-	+	+
facets						

Default slot value	-	+	+	-	-	-
Type constraint	+	+	+	+	+	+
Cardinality constraint	+	+	+	-	+/-	+
Other slot constraints	+	+	+	-	+	+
Other features						
Meta-information on ontology	+	-	+	-	+	+
Ontology to be included	-	-	-	-	+	+
Primitive datatypes	+	+	+	-	-	-

2 Comparison on existing ontology developments

The first interest is to compare the expressiveness of the underlying knowledge model supported by the API. Second is to focus on the implementation part to evaluate the API possibilities of evolution.

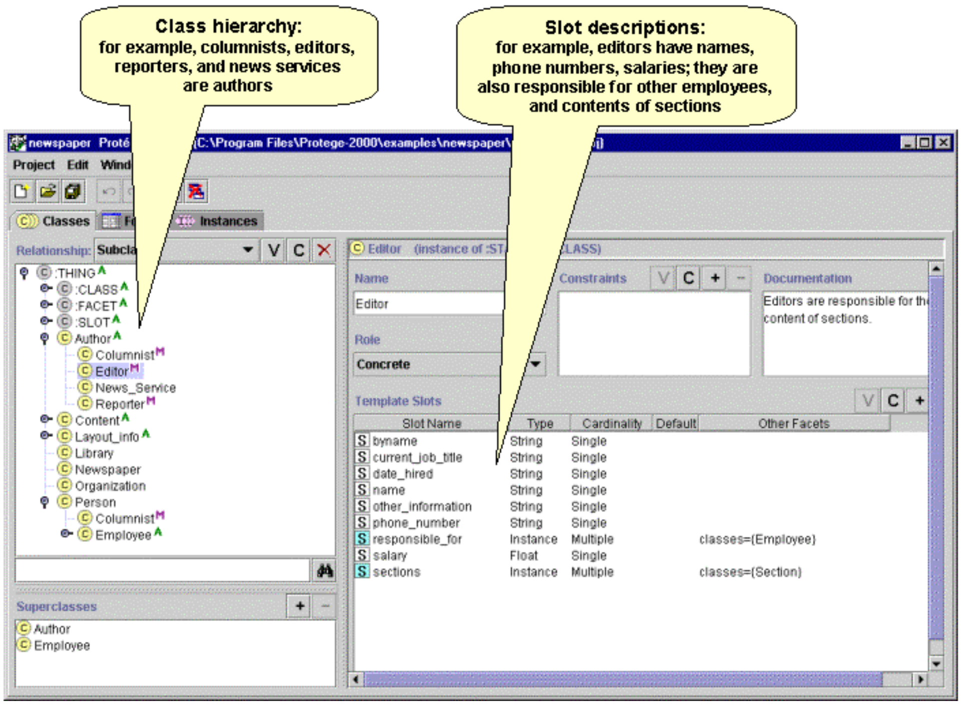
Knowledge model
1- basic elements for ontology representation (classes, slots, facets, instances)
2- advanced elements for ontology representation (functions, axioms, rules)
3- is it possible to use multi-inheritance ?
4- is it possible to represent meta-ontologies (high-level primitives) ?
5- Does the tool check new data for consistency with the ontology ?
6- Is it possible to re-use an existing ontology via an operation of inclusion/union ?
Building ontologies
1- Are there example-ontologies available in the tool ?
2- Does the tool provide librairies of ontologies that can be re-used ?
3- Is help sufficient and well organized ?
Cooperation
1- Does the tool allow synchronous editing of the same ontology by different users ?
2- are there ways to lock an ontology ?
3- are the changes made by other user easy to recognize ?
4- Is it possible to export the ontology's code in various format ?
5- Is it possible to import an ontology-description from another tool ?
General use of the tool
1- Evaluation of interface for different steps in ontology building
2- Is the meaning of the command clear ?
3- Evaluation of the stability of the tool and maintenance ?
Evaluation of the API
1- is it possible to easy extend the API ?
2- is it possible to customize the knowledge acquisition interface?

2.1 Protégé-2000

2.1.1 Presentation

Protégé provides an integrated knowledge-base editing environment and an extensible architecture for the creation of customized knowledge-based tools [15], [16]. It's a computer program, which should be installed locally and also an extendable platform (API). It's available on different platform like Windows, Mac OS, Solaris , Linux ,Unix. Protégé has been designed by Stanford's Medical Informatics Section (http://protege.stanford.edu/). You can also download plugins to improve the capabilities of the tool.
We have used the screen-shots of the protégé web site to present the different part of the tool.
The tool is dedicated to knowledge engineer for the following purposes:

- construct a domain ontology



Class hierarchy: for example, columnists, editors, reporters, and news services are authors

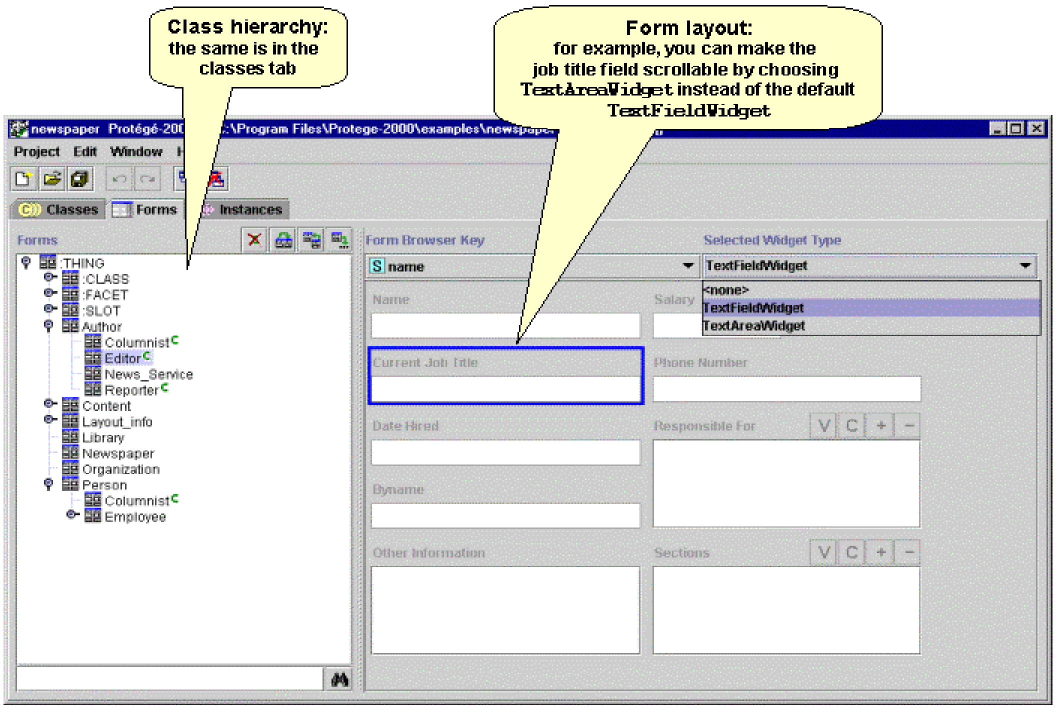
Slot descriptions: for example, editors have names, phone numbers, salaries; they are also responsible for other employees, and contents of sections

Editors are responsible for th content of sections.

Slot Name	Type	Cardinality	Default	Other Facets
byname	String	Single		
current_job_title	String	Single		
date_hired	String	Single		
name	String	Single		
other_information	String	Single		
phone_number	String	Single		
responsible_for	Instance	Multiple		classes={Employee}
salary	Float	Single		
sections	Instance	Multiple		classes={Section}

Fig. 4: Screenshot of the protégé-2000 interface for buiding concepts

o customize knowledge acquisition user interface

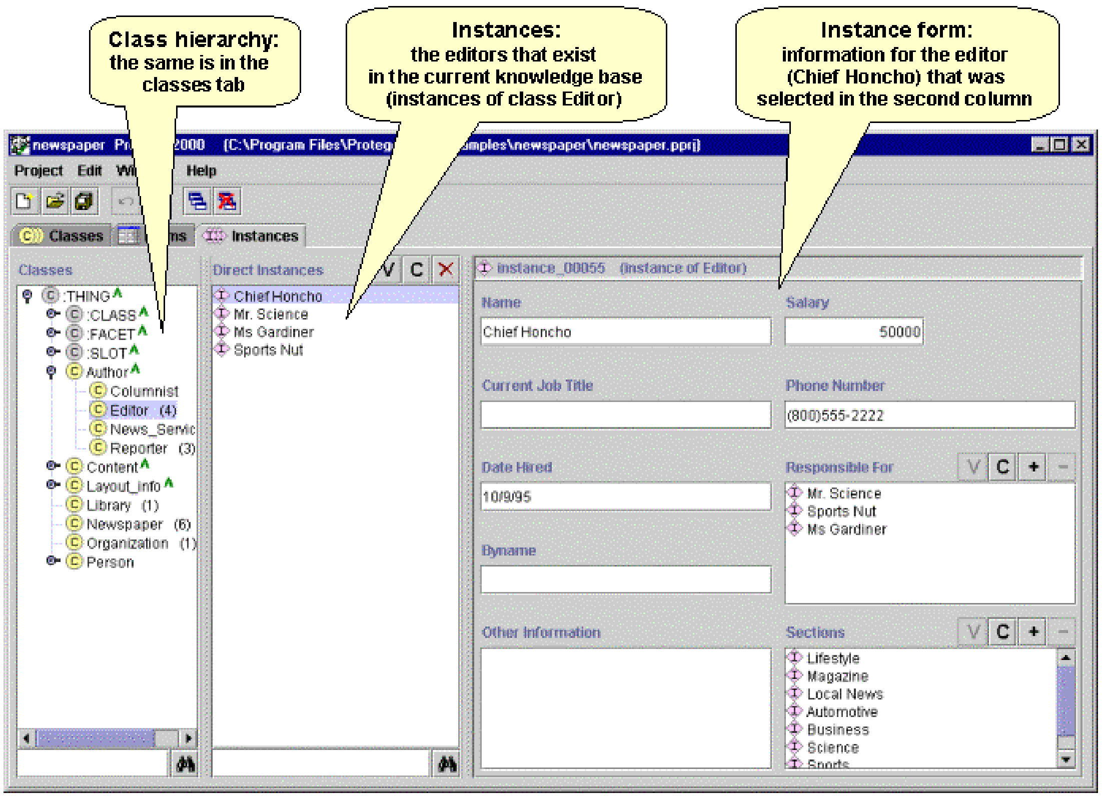


Class hierarchy: the same is in the classes tab

Form layout: for example, you can make the job title field scrollable by choosing TextAreaWidget instead of the default TextFieldWidget

Fig. 5: Screenshot of the protégé-2000 interface for customizing knowledge acquisition interface

o enter domain knowledge


Class hierarchy: the same is in the classes tab
Instances: the editors that exist in the current knowledge base (instances of class Editor)
Instance form: information for the editor (Chief Honcho) that was selected in the second column
newspaper Pr 2000 (C:\Program Files\Proteg mples\newspaper\newspaper.pprj)
Project Edit Wi Help
Classes ms Instances
Classes
:THING
:CLASS
:FACET
:SLOT
Author
Columnist
Editor (4)
News_Servic
Reporter (3)
Content
Layout_info
Library (1)
Newspaper (6)
Organization (1)
Person
Direct Instances
V C X
Chief Honcho
Mr. Science
Ms Gardiner
Sports Nut
instance_00055 (instance of Editor)

Name: Chief Honcho
Salary: 50000
Current Job Title:
Phone Number: (800)555-2222
Date Hired: 10/9/95
Responsible For: Mr. Science, Sports Nut, Ms Gardiner
Byname:
Other Information:
Sections: Lifestyle, Magazine, Local News, Automotive, Business, Science, Sports

Fig. 6: Screenshot of protégé-2000 for building instances

The API is dedicated to software developers to implement new languages and new features that they would like to support in their applications.

2.1.2 Knowledge model

Classes in protégé-2000 constitute a taxonomic hierarchy. Protégé-2000 visualizes the subclasses relation in a tree, it supports multiple inheritance and the root of the class hierarchy is the built-in class :THING. As protégé implements the use of metaclasses, both individuals and classes themselves can be instance of classes. A metaclass is a template that is used to define new classes in an ontology, such feature brought flexibility to the API because it enables developers to use Protégé as an editor for different knowledge representation systems, so different frame-based languages (such as DAML + OIL, RDF) are plugins available in the download area of protégé-2000). For example the class A in Figure 7 is a metaclass and the class B is an instance of this metaclass.

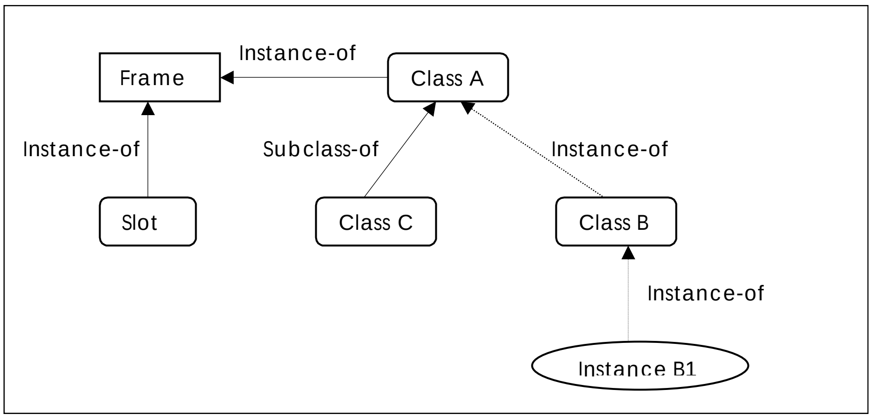


Figure 7. Protege Knowledge model

Slots describe properties of classes and instances; a slot itself is a frame. Like in OKBC slots are first class objects, they are defined independently of any class. You can't have two slots with different signatures and same name. Protégé implements two sorts of slots: template and own slots. An own slot attached to a frame describes properties of an object represented by a frame. Own slots attached to a class, do not inherited to its subclasses or propagated to its instances.
Facets are one way to specify constraints on allowed slot values
As Protégé is based on an OKBC representation, relations are binary relations and you can't represent functions through slots. The only way to represent functions is to reify relations as classes (see the plugin Relations: http://protege.stanford.edu/plugins.html). Axioms and rules cannot explicitly be represented in Protégé, you have to use plugins of ontology representation languages that enables the representation of axioms and rules like DAML + OIL, PAL tabs (Protégé Axiom Language).
A complete user interface is customisable to allow the creation of instances. This feature helps developers to implement a dedicated user interface for the new language (plugin) they have implemented from the Protégé API.
It is also possible to import an existing ontology via an include operation, but it's more an union operation because it does not merge an ontology into another. To merge ontologies, Henrik Eriksson proposed a plugin for protégé called "The PROMPT Tab".It guides you through the merging process making suggestions, determining conflicts, and proposing conflict-resolution strategies.

2.1.3 Buiding ontologies and general use of the tool

Protégé provides some generic ontologies (Dublin Core Ontology, GLIF Ontology, Ontology of SCIENCE) for reuse and examples, but it does not constitute a real basis for building a new ontology. An example-ontology is available in the help facility in HTML. It helps to understand how to build an ontology in a general sense and with the protégé-2000 tool. It proposes some guidelines to design an ontology. There exists several mailing lists (protege-users, protégé-discussion, protégé-beta) on the tool that are really active and helpful for knowledge engineer and developers. A workshop is also organized each year: The *International Protégé Workshop* brings together researchers developing or using Protégé development methodologies and tools.

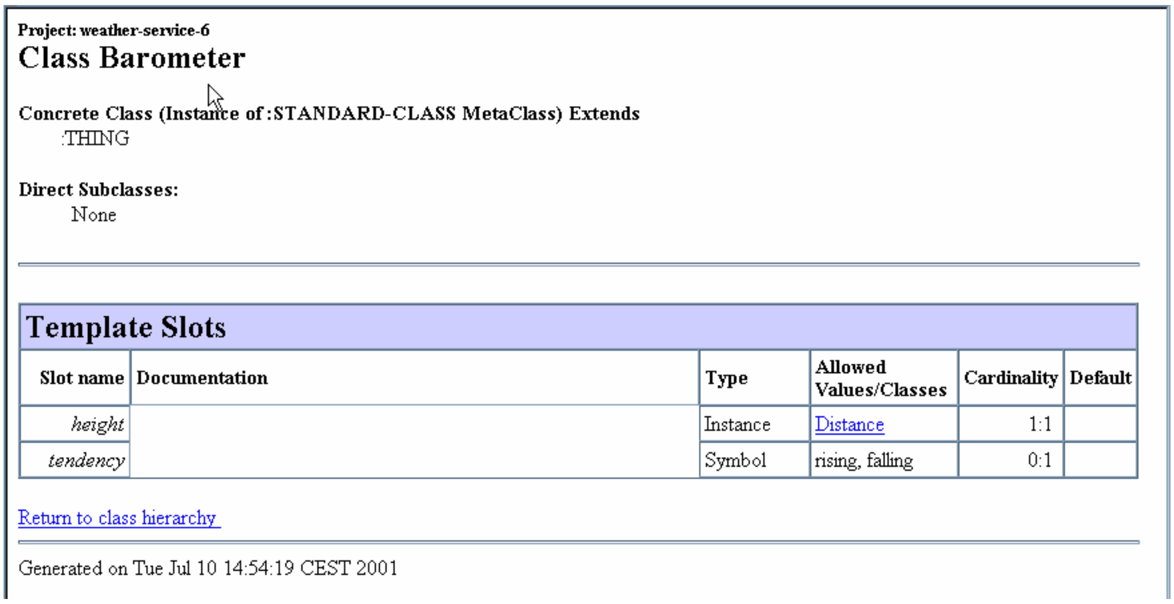
2.1.4 Cooperation

Protégé is a tool, which should be installed locally in your computer. It doesn't enable the synchronous editing of an ontology by different users. However it's possible to import/export ontologies in different format:

- *Text Files*: You can import a project from two text files describing the classes/slots and instances information. Importing a text project can be used, for example, for updating from Protégé/Win to Protégé-2000.
- *Database Table*: You can import a project from a table in a JDBC database.
- *Resource Description Framework (RDF) Files*: You can import a project from two RDF files that describe the classes/slots and instances information.

2.1.5 Evaluation of the API

The API is well structured and scalable. The knowledge model is implemented as several java interfaces, so it's very easy to extend the tool to fulfil different goals. It's also easy to implement a new knowledge representation language. As a proof of those possibilities, several people on the mailing list have already proposed plug-ins for principally Jess, OKBC, OIL, Topic maps etc. They are all accessible on the following address: http://protege.stanford.edu/plugins.html.
A big advantage of protégé tool is the possibility to customize the user interface for knowledge acquisition. You could define your own interface to help people who will use the tool to enter knowledge. This solution could open the tool to non-expert people who needs to enter assertional knowledge in the knowledge base. Another advantage is the automatic generation of documentation on ontology concepts (see fig 8)


Project: weather-service-6

Class Barometer

Concrete Class (Instance of :STANDARD-CLASS MetaClass) Extends
:THING

Direct Subclasses:
None

Template Slots

Slot name	Documentation	Type	Allowed Values/Classes	Cardinality	Default
height		Instance	Distance	1:1	
tendency		Symbol	rising, falling	0:1	

Return to class hierarchy

Generated on Tue Jul 10 14:54:19 CEST 2001

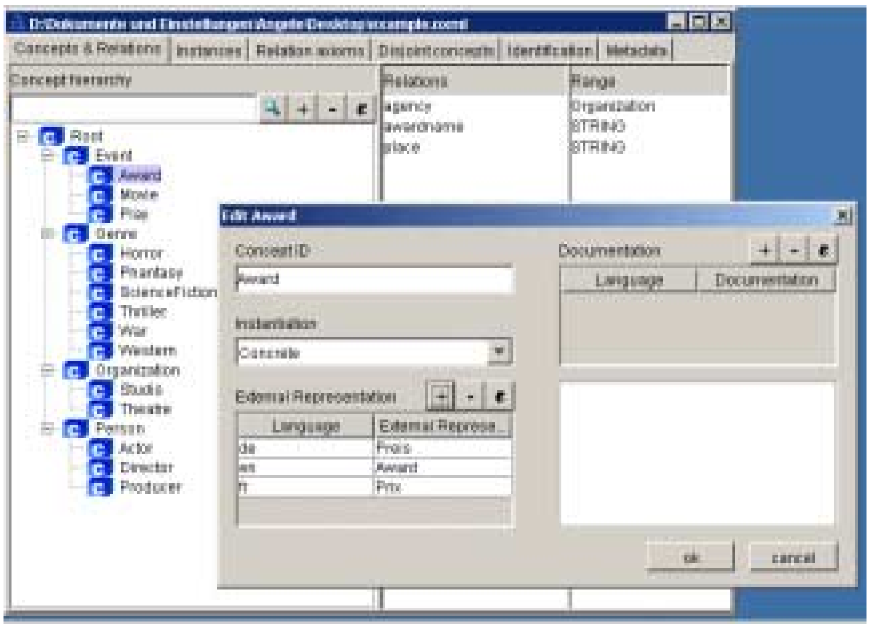
Fig 8: Documentation of a concept built with protégé-2000

2.2 OntoEdit

OntoEdit is a free product proposed by Ontoprise. Ontoprise (http://www.ontoprise.de) was co-founded by Prof. JAngele, Hans-Peter Schnurr, Prof. Rudi Studer and Dr. Steffen Staab. Ontoprise develops semantic tools and middleware, semantic platforms and semantic applications for the Semantic Web. OntoEdit (downloadable at http://www.ontoprise.de/start_products.htm) is a development environment for design, adaptation and import of knowledge models for application systems. OntoEdit supports multilingual development of ontologies and multiple inheritance. OntoEdit relies on W3C standards and offers several export interfaces.
OntoEdit comes with two other applications that are not free: Ontobroker™ and OntoAnnotate. OntoBroker™ enables the processing of knowledge described with OntoEdit and help in validating this knowledge. OntoAnnotate help in annotating Web Pages based on the ontologies described in OntoEdit. OntoEdit is dedicated to the OIL language, that's why the knowledge model of this tool is closed to the knowledge model of OIL.

2.2.1 Knowledge model

Classes in Ontoprise constitute a taxonomic hierarchy. Ontoprise visualizes the subclasses relation in a tree, it supports multiple inheritance and the root of the class hierarchy is the built-in class :ROOT (see fig. 9). Each concept can correspond to different words in different languages, so OntoEdit propose to associate to a concept a list of external representation each corresponding to a language.



Relations	Range
agency	Organization
awardname	STRING
place	STRING

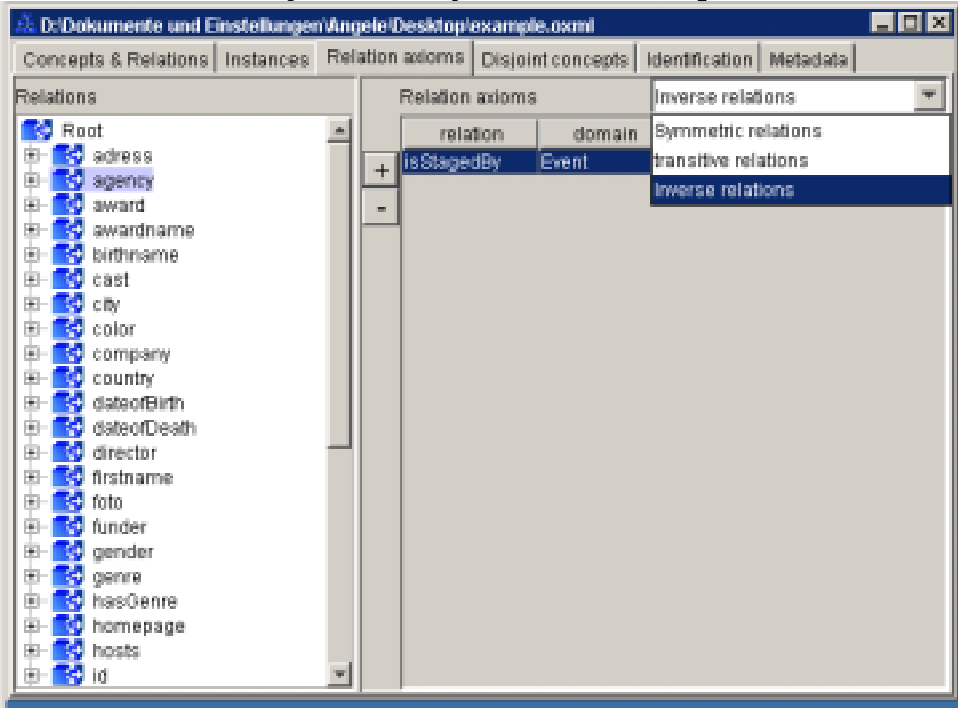
Concept ID: Award

Instantiation: Concrete

Language	External Represe...
de	Preis
en	Award
fr	Prix

Fig. 9: Concept hierarchy and multilingual representation of a concept

Slots or relations describe properties of classes and instances. As OIL comes from frame-based language such as OKBC, slots are first class objects. They are defined independently of any class. You can't have two slots with different signatures and same name. Ontoedit allows the creation of instances and so must deal with multiple inheritance issues. OntoEdit implements local and global slots, they are like own and template slots of OKBC knowledge model or Protégé knowledge model .As you can see in fig 10, OntoEdit enables to create relation axioms by defining generic properties on slots like symmetric, transitive. This feature is related to the OIL knowledge model and can be found in Protégé through the OIL plugin, but is not included in the knowledge model of the tool like in OntoEdit. It's not possible to implement meta-ontologies with OntoEdit.



relation	domain
isStagedBy	Event

Fig 10: list of relation with their signature and associated axioms.

Facets are one way to specify constraints on allowed slot values: in Ontoedit you can just specify constraints on type of values (range, domain) and constraints on cardinality. The only built-in range values are STRING, BOOLEAN, INTEGER and any class derived from :ROOT.

Some predefined axioms can be expressed like disjunction of concepts. Other predefined axioms are available in the OIL knowledge model but not implemented in the free version of OntoEdit.

One of the particularities of OIL is to define an ontology container that describes some meta-data about the ontology. OntoEdit implements this container through two different tabs (fig. 11): an identification tab that provide information on the ontology: URI, Title, Domain of the ontology, application area, related ontology used, etc. and a metadata tab providing information on the development part of the ontology: documentation about developers, list of documentations, language and some statistics about the ontology.

It's possible to import an ontology but there is no operation to aggregate or merge two ontologies. Consistency is checked during the building process.

2.2.2 Building ontologies and general use of the tool

Ontoprise doesn't provide any generic ontologies for reuse. Some examples are available to understand OntoEdit features. The help and tutorial guide is sufficient to understand the different functionalities and their use. Each tab for the different steps of ontology building are easy to use and menus are easy to understand. The tool is maintained by Ontoprise and new features are now available only with the non-free version.

2.2.3 Cooperation

Ontoprise is a tool, which should be installed locally in your computer. It doesn't enable the synchronous editing of an ontology by different users. It's only possible to import DAML + OIL ontology (RDF also available on the non-free version) but you could export ontologies in different format: Flogic, DAML + OIL, DTD, SQL-2 and RDF, XML-Schema on the non-free version.

2.2.4 Evaluation of the API

The documentation of the API is not available. OntoEdit seems to be difficult to extend. It does not implement the notion of meta-ontologies. A lot of features are available only on the non free version.

References:

[1] Oscar Corcho and Asuncion Gomez-Perez. In proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods at the 4th European Conference on Artificial Intelligence, ECAI'00, Berlin, Germany, August 20-25, 2000.
Internet: http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/accepted-papers.html

[2] Marin Dimitrov. XML Standards for Ontology Exchange. In Proceedings of "OntoLex 2000: Ontologies and Lexical Knowledge Bases", Sozopol, Sept. 8-10, 2000.

[3]A. Duineveld and R. Studer and M. Weiden and B. Kenepa and R. Benjamis. WonderTools? A comparative study of ontological engineering tools. Proceedings of KAW99. Banff, Canada. 1999.

[4] F. van Harmelen, I. Horrocks. Reference description of the DAML+OIL ontology markup language. Internet: http://www.daml.org/2000/12/reference.html, http://www.daml.org/2000/12/daml+oil-walkthru.html

[5] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M.Erdmann and M. Klein. OIL in a nutshell In: Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), R. Dieng et al. (eds.), Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag, October 2000.

[6] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann and M. Klein. OIL in a nutshell. In Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), R. Dieng et al. (eds.), Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag, October 2000. http://www.cs.vu.nl/~ontoknow/oil/downl/oilnutshell.pdf

[7] Nicola Guarino. Understanding, Building and Using Ontologies, International Journal of Human and Computer Studies vol. 46 n. 2/3, 1997, pp. 293-310,
Internet: http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/vanHeijst.pdf

[8] Nicola Guarino. Formal Ontology in Information Systems. Proceedings of the International Conference on Formal Ontology in Information Systems, FOIS’98, Amsterdam, IOS Press, pp.3-15. Trento, Italy, 6-8 1998.

[9] I. Horrocks & al.: The Ontology Inference Layer OIL
Internet: http://www.cs.vu.nl/~dieter/oil/Tr/oil.pdf

[10] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. The Ontology Inference Layer OIL, June 2000.
http://www.cs.vu.nl/~dieter/oil/Tr/oil.pdf

[11] R. Jasper, M. Uschold. A framework for Understanding and Classifying Ontology Applications. In proceedings of IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, Stockholm, Sweden, august 1999.
Internet: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf

[12] Knowledge Interchange Format draft proposed American National Standard:
http://logic.stanford.edu/kif/dpans.html

[13] Marian H. Nodine, Amy Unruh. Facilitating Open Communication in Agent Systems: the InfoSleuth Infrastructure. Technical Report MCC-INSL-056-97, Microelectronics and Computer Technology Corporation, Austin, Texas 78759, April.1997.

[14] M. Nodine, J. Flower, T. Ksiezyk, B. Perry, M. Taylor and A. Unruh. Active Information Gathering in Infosleuth. International Journal of Cooperative Information Systems. Vol. 9, Nos. 1 & 2 (2000) 3-27.

[15] N. F. Noy, R. W. Fergerson, & M. A. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France,2000.

[16] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web Contents with Protégé-2000.M.A In IEEE Intelligent Systems, Vol. 16, No. 2, March/April 2001, special issue on Semantic Web, pp. 60-71. Available as SMI technical report SMI-2001-0872 (2001).

[17] An informal description of Standard OIL and Instance OIL.
http://networkinference.semanticweb.org/downloads/OIL.pdf

[18] Resource Description Framework, model and syntax specification, W3C recommendation, February 1999. http://www.w3.org/TR/REC-rdf-syntax/ .

[19] Resource Description Framework Schema Specification 1.0. W3C candidate recommendation march 2000. http://www.w3.org/TR/rdf-schema/

[20] Steffen Staab, Michael Erdmann, Alexander Maedche, Stefan Decker. An extensible Approach for modelling Ontologies in RDF(S). In proceedings of ECDL Workshop on Semantic Web. Lisbon, Portugal, 21 September 2000. http://www.ics.forth.gr/proj/isst/SemWeb/proceedings/session2-1/html_version/.