# EOL Guide

25/04/2002 07:06

A guide to the Exteca Ontology Language

**Version History**
25 April 2002First Draft
11 Aug 2002 Brought up to date with current implementation

**Authors**
Llewelyn Fernandes
Mauro Talevi
Neetu Jain

# Contents

# Introduction

This document provides an introduction to knowledge engineering using the Exteca Ontology Language. It is not intended as a complete definition of EOL – for that please see the EOL Specification.

# EOL Step-by-step

In this section we will see how the Exteca Ontology Language can be used to model knowledge through some worked examples.

## *Simple Community Example*

In this example we will build a simple ontology that represents a community of people living together and sharing some interests.

### Ontology declaration

The first thing we need to do is to declare what the ontology will be called and how we can reference it in the future:

```
<ontology id=”community” uri=”www.exteca.com/community” version=”1”>
```

### Metadata declaration

Next we can say a fair amount about the ontology itself. The metadata section follows the Dublin Core set. Here we have added just a few elements.

```
<metadata>
        <metadata-title>An ontology for my community</metadata-title>
        <metadata-creator>Exteca</metadata-creator>
        <metadata-subject>Community, People, Interests</metadata-subject>
        <metadata-description>This ontology represents a virtual community of
        people sharing common interests</metadata-description>
        <metadata-publisher>Exteca</metadata-publisher>
        <metadata-date>9 April 2002</metadata-date>
</metadata>
```

### Define some primary concepts

One of the first things you will want to do is to think about the types of things you are modelling. These are the concepts. You can think of concepts as classes or sets which define the characteristics and positioning of individuals within the world being modelled. We want to model people and the places they live in and the hobbies they have.

```
<concept id=”Person>
</concept>

<concept id=”Place”>
</concept>

<concept id=”Hobby>
</concept>
```

### Define some associations

Next we think about how we want to relate the concepts in the ontology. Associations are binary relations and are always defined with a name for both directions.

We want to say where people live

```
<association id="resident-of inverse-id="has-resident"/>
```

and what hobbies they enjoy

```
<association id="enjoys" inverse-id="enjoyed-by"/>
```

## Define some attributes

We may also want to model certain characteristics of concepts. In EOL we define these attributes first without tying them to particular concepts. This allows us to freely add attributes to concepts later on. (Computer programmers note that this is equivalent to defining the data type of *all* properties of all classes before the class definition).

So we can say that we want to keep information about the internet connection speeds that people have access to

```
<attribute id="internet-connection-speed" data-type type="decimal"/>
```

and the date the concept was entered on the system

```
<attribute id="date-entered" data-type type="date"/>
```

## Define some instances of people

Now we define some actual people in our community. For this we use the special pre-defined **instance-of** association.

```
<concept id="Alice">
        <instance-of concept="Person"/>
</concept>

<concept id="Bob">
        <instance-of concept="Person"/>
</concept>
```

## Define some hobbies

Now we'll add in some hobbies. We will create a hierarchy of hobbies that represents the following:

- Arts
  - Cinema
  - Theatre
- Homemaking
  - DIY
  - Gardening
- Sport
  - Fishing
  - Golf
  - Tennis

Here are the entries in the ontology. We use the special pre-defined association **subclass-of** to show that once concept is a subclass (or subset) of another.

```
<concept id="Arts">
```

```
        <subclass-of concept="Hobby"/>
</concept>

<concept id="Cinema">
        <subclass-of concept="Arts"/>
</concept>
<concept id="Theatre">
        <subclass-of concept="Arts"/>
</concept>

<concept id="Homemaking">
        <subclass-of concept="Hobby"/>
</concept>

<concept id="DIY">
        <subclass-of concept="Homemaking"/>
</concept>

<concept id="Gardening">
        <subclass-of concept="Homemaking"/>
</concept>

<concept id="Sport">
        <subclass-of concept="Hobby"/>
</concept>

<concept id="Fishing">
        <subclass-of concept="Sport"/>
</concept>

<concept id="Golf">
        <subclass-of concept="Sport"/>
</concept>

<concept id="Tennis">
        <subclass-of concept="Sport"/>
</concept>
```

## Define some places

As we did for hobbies we want to define a hierarchy of places. The way we do this illustrates some interesting techniques for modelling hierarchies, using the specially defined associations **instance-of** and **class-of** and an association we will define called **part-of**.

First let's define part-of. We declare part-of to be a subassociation of the special predefined **hierarchy** association.

```
<association id="part-of" inverse-id="has-part">
        <subassociation-of association="hierarchy"/>
</association>
```

Now we can create our hierarchy.

Places can be countries, cities or districts

```
<concept id="Country">
        <subclass-of concept="Place"/>
</concept>
<concept id="City">
        <subclass-of concept="Place"/>
```

```
</concept>
<concept id="District">
        <subclass-of concept="Place"/>
</concept>
```

England is a country

```
<concept id="England">
        <instance-of concept="Country"/>
</concept>
```

London is a city, and part of England

```
<concept id="London">
        <instance-of concept="City"/>
        <link association="part-of" concept="England"/>
</concept>
```

Peckham and Battersea are districts of London

```
<concept id="Peckham">
        <instance-of concept="District"/>
        <link association="part-of" concept="London"/>
</concept>
<concept id="Battersea">
        <instance-of concept="District"/>
        <link association="part-of" concept="London"/>
</concept>
```

This gives us the following hierarchy of concepts

```
Place
        Country
                England
                        London
                                Peckham
                                Battersea
        City
                London
                        Peckham
                        Battersea
        District
                Peckham
                Battersea
```

The interesting point here is that concepts appear at multiple places in the hierarchy! This is entirely intentional and gives us a very rich and natural way to navigate the ontology.

## Classes, Instances and hierarchical modelling

In the above definition we have introduced some techniques that could do with some further clarification.

In EOL concepts are either classes or instances. Classes are collections of concepts, whilst instances are individuals that belong to one or more classes. Classes can be futher broken down into subclasses. A special predefined class called Top is the super class of all classes. The path from Top through classes and subclasses and through to instances gives a convenient hierarchical representation of the domain we are modelling:

```
Top
```

ClassA
ClassB
InstanceX

This hierarchical structure can be generated by examining the way concepts are linked using the special pre-defined associations subclass-of and instance-of. EOL formalises this by considering subclass-of and instance-of to be subassociations of an association called **hierarchy**.

A concept linked to another concept with subclass-of is itself a class. A concept linked to another concept with instance-of is itself an instance. A concept cannot be both a class and a instance! A concept that is not linked with subclass-of or instance-of is considered to be a subclass-of Top.

If we briefly revisit the definition of places in our ontology we will see an example of this.

Place is a subclass of Top:

```
<concept id="Place">
</concept>
```

Country is a subclass of Place:

```
<concept id="Country">
    <subclass-of concept="Place"/>
</concept>
```

and England is an instance of Country:

```
<concept id="England">
    <instance-of concept="Country"/>
</concept>
```

Sometimes we want to take a hierarchy further than just classes and instances. We can do this by defining a new association as a subassociation of hierarchy and using this to extend the hierarchy.

Again we can revisit the definition of places to see this working.

First we defined part-of:

```
<association id="part-of" inverse-id="has-part">
    <subassociation-of association="hierarchy"/>
</association>
```

then we used part-of to extend our hierarchy:

```
<concept id="London">
    <instance-of concept="City"/>
    <link association="part-of" concept="England"/>
</concept>
```

It is important to use the correct association to model each hierarchical relation:

**subclass-of** should be used where we are creating a subclass or subset of another class. Reading the association out loud should make sense, so "City is a subclass of Place" makes sense, but "London is a subclass of England" does not.

**instance-of** should be used where we have a single uniquely defined member of a class. So, London is a unique place – there is only one place that is London. (Having other places called London in other coutries does not change this, as they are different Londons.)

**part-of** should be used where we are naming a physical part of another thing. Again reading out the association should make sense: "London is a part of England".

Care must be taken not to use subclass-of when you mean instance-of or vice-versa. It is tempting to say that "London is a subclass of City", but the distinction is that there is only one London, not a class of Londons. Similarly "City is an instance of Place" would be wrong because there are many cities.

## Add some properties and links

Now we have a framework defined and we can start adding specific information about the concepts to the ontology. We will redefine Alice and Bob to include this information.

Alice lives in Battersea and enjoys cinema and tennis. She joined the community on 12 Jan 2002 and has a high-speed internet access.

```
<concept id="Alice">
        <instance-of concept="Person"/>
        <link association="enjoys" concept="Cinema"/>
        <link association="enjoys" concept="Tennis"/>
        <link association="resident-of" concept="Battersea"/>
        <property attribute="date-entered" value="12/01/2002"/>
        <property attribute="internet-connection-speed" value="512kbs"/>
</concept>
```

Bob lives in Peckham and enjoys golf, fishing and cinema. He entered the community on 15 April 1996 and only has modem access.

```
<concept id="Bob">
        <instance-of concept="Person"/>
        <link association="enjoys" concept="Golf"/>
        <link association="enjoys" concept="Fishing"/>
        <link association="enjoys" concept="Cinema"/>
        <link association="resident-of" concept="Peckham"/>
        <property attribute="date-entered" value="15/04/1996"/>
        <property attribute="internet-connection-speed" value="56kbs"/>
</concept>
```

# *Advanced Community Example*

We will now augment the simple community ontology with some advanced features.

## Global Restrictions

The simple community ontology relates concepts but leaves the way that concepts are related completely open. We now introduce restrictions on concepts which serve the purpose of providing added knowledge to the ontology and also as a validation mechanism to make sure that concepts are only related in a meaningful way.

Let's remind ourselves of the definintion of resident-of:

```
<association id="resident-of inverse-id="has-resident"/>
```

When we defined this association we had in mind the notion of people living in places. The way it is defined at the moment we can also define hobbies as being residents of people! We

can restrict the way the resident-of association is used by applying global restrictions. Global restrictions apply to every use of an association.

Let's restrict resident-of to relate only people to places

```
<association id="resident-of" inverse-id="has-resident"/>
        <mapping domain="Person" range="Place"/>
</association>
```

We may also want to say that a person can only live in one place and a place can have 0 or more people. This can be achieved using cardinality restrictions.

```
<association id="resident-of" inverse-id="has-resident"/>
        <mapping domain="Person" range="Place"/>
        <cardinality count="1" inverse-count=">=0"/>
</association>
```

## Local Restrictions

Sometimes we want to restrict an association in particular ways for particular concepts. A district can only belong to a single city

```
<concept id="District">
        <exists-link association="part-of" concept="City" cardinality="1"/>
</concept>
```

Also we can define enirely new concepts using restrictions. Sporty people are the type of people who enjoy sport.

```
<concept id="Sporty People">
        <subclass-of concept="People"/>
        <exists-link association="enjoys" concept="Sport"/>
</concept>
```

Here we have said that for any concept within People, they can be classed as a sporty person if there exists at least one Sport that they enjoy. We may want to make the restriction stronger and say that you are not really sporty unless you play 3 or more sports:

```
<concept id="Sporty People">
        <subclass-of concept="People"/>
        <exists-link association="enjoys" concept="Sport" cardinality="3+"/>
</concept>
```

Now let's define the concept of people who are dedicated art lovers and like nothing else:

```
<concept id="Art Lovers">
        <subclass-of concept="People"/>
        <forall-links association="enjoys" concept="Art"/>
</concept>
```

Here we have said that with art lovers all the things they enjoy are concepts within Art.

## Indicating preferences with link strengths

Up to now everything we have defined is defined with certainty. Someone either enjoys tennis or they don't. It is useful to be able to express the degree to which a link is made. This can be achieved with the strength attribute.

```
<concept id="Bob">
```

```
        <instance-of concept="Person"/>
        <link association="enjoys" concept="Golf" strength="80"/>
        <link association="enjoys" concept="Fishing" strength="10"/>
        <link association="enjoys" concept="Cinema" strength="40"/>
</concept>
```

The strengths are values between 0 and 100. So in the above we have stated that Bob like golf twice as much as he likes cinema, and he only has a little interest in fishing.

## Signposts as a navigational aid

A common aim of an ontology is the creation of a "controlled vocabulary" where there is a preferred name for each concept. For example we might say that "Cinema" is the preferred name for "Movies" and "Films". It is always useful to add these non-preferred concepts to the ontology too as they can be used to point the user of the ontology towards the controlled vocabulary concepts. They also provide useful semantic information for automated processes.

Here's how we define these "signposts".

```
<concept id="Cinema">
        <subclass-of concept="Arts"/>
</concept>
<concept id="Movies">
        <signpost-of concept="Cinema"/>
</concept>
<concept id="Films">
        <signpost-of concept="Cinema"/>
</concept>
```

## Extending ontologies with imports and namespaces

Ontologies can be based on an existing ontology. This is a very useful mechanism for

- extending an existing ontology by adding new concepts
- providing further restrictions on concepts
- providing domain-specific variations on an ontology
- organising your work by separating parts of the model

Let's see how this works. Imagine we want to split our ontology above into a general framework ontology and numerous community-specific ontologies.

We can define the framework incorporating all the basic definitions of primary concepts and associations:

```
<ontology id="community" uri="www.exteca.com/community" version="1">
        …basic definitions…
</ontology>
```

and then create an extension that imports this ontology:

```
<ontology id="my-community" uri="www.exteca.com/my-community" version="1">
        <import url="www.exteca.com/community" prefix="cty"/>
        …futher definitions…
</ontology>
```

The prefix is used to reference elements from the base ontology, so for example we can define a new person, Peter, as:

```
<concept id="Peter">
        <instance-of concept="cty:Person"/>
</concept>
```

## Assigning Content

EOL provides some extensions to the modelling language to allow content and content descriptions to be incorporated in to the model. This is not always a requirement and sometimes it is useful to model content completely outside of the semantic ontology model, but on other occasions this approach has benefits.

The EOL content description consists of resources and rules. Resources are actual references to content, such as documents or web pages, that are relevant to a concept. For example we can add a reference to Bob's web site about fishing:

```
<concept-content concept="Bob">
        <resources content="web">
                <resource url="www.bobsfishingpage.com"/>
        </resources>
</resources>
```

Rules define how particular types of content map to concepts. This information is used by automated content classification solutions to automatically assign concepts to documents. Here is a rule that classifies any web page containing the phrase "Bob Jones" to be about Bob:

```
<concept-content concept="Bob">
        <rules content="web">
                <rule logic="Bob Jones"/>
        </rules>
</resources>
```

This is a very simple example of a rule. Rules can be built to provide very sophisticated and highly accurate categorisation. Further details can be found in separate documentation.