CHAPTER 4

OWL, VISUAL AND KNOWLEDGE REPRESENTATIONS OF THE SRMONTO

4.1 INTRODUCTION

One of the most important characteristics of knowledge management is representation of knowledge using formal representation mechanisms. This chapter deals with various representations of SRMONTO which has been described in the previous chapter, to make even the common user to understand the domain easily. Through this, the domain knowledge representation, which is one of the applications of ontology, is achieved. Visualization is becoming increasingly important in Semantic Web tools. OWLViz is designed to be used with the Protege OWL plugin. OWLViz integrates with the Protege-OWL plugin, using the same colour scheme, so that primitive and defined classes can be distinguished, computed changes to the class hierarchy may be clearly seen and inconsistent concepts highlighted separately. OWLViz has the facility to save both the asserted and inferred views of the class hierarchy to various concrete graphics formats including png, jpeg and svg. It is used to visualize light-weight ontologies that describe a domain through a set of classes (concepts) and their hierarchical relationships. Also known as taxonomies, such ontologies are frequently used in several domains as classification systems. The types of visualizations are highly configurable and include picking a set of classes or instances to visualize part of an ontology, displaying slots and slot edges, specifying
colors for nodes and edges, and when picking only a few classes or instances, the user can apply various closure operators (e.g., subclasses, superclasses) to visualize their vicinity.

The SRMONTO is represented using the following three representational mechanisms.

1. OWL Representation
2. Visual Representation
3. Knowledge Representation

Figure 4.1 demonstrates the five top levels of the SRMONTO, which is explained in Chapter 3. It is an integrated ontology of Software Risk Identification Ontology (SRIONTO), Software Risk Analysis Ontology (SRAONTO), Software Risk Planning Ontology (SRPONTO), Software Risk Control Ontology (SRCONTO) and Software Risk Tracking Ontology (SRTONTO) where N(C) is the number of concepts and N(P) is the number of relationships each ontology has.

![Figure 4.1 The Structure of the SRMONTO](image-url)
4.2 OWL REPRESENTATION OF THE SRMONTO

In order to make the constructed ontology machine processable, it must be represented in a formal representation language format. Among various Ontology Representation Languages discussed in the literature survey such as the XML, RDF, REFS and OWL, the author found that OWL has a lot of advantages over the others. Hence, the following sub topics describe the owl representation of the SRMONTO.

The OWL has been built, using the RDF to remedy the weaknesses in the RDF/S and DAML+OIL. It provides a richer integration and interoperability of data among communities and domains. It can be said that there is a similarity between the OWL and the RDF, but the former is a stronger syntax with more machine interpretability and vocabulary language than the RDF. Obviously, the RDF is generally limited to binary ground predicates, and the RDF Schema also has the limitation that it represents a subclass hierarchy and a property hierarchy, with the domain and range definitions of these properties. In other words, the language of the OWL is more expressive than that of the RDF and RDF Schema.

As discussed in Chapter 3, though the SRMONTO is an integrated work of five sub ontologies, each sub ontology works individually; all classes in the sub ontologies are subclasses of the class ‘Thing’. Its notation description is the same with an ontology class notation. OWL documents are usually called OWL Ontologies, the elements of which are Namespaces, Housekeeping, classes, properties, property restrictions, enumerations and instances.

Because the OWL is written in the RDF, and the RDF is written in the XML, the OWL documents start with several namespace declarations using the RDF, XML Namespace, and URIs. rdf:RDF is the root element of
an OWL Ontology, and also specifies a number of namespaces. Figure 4.2 shows the representation of namespace declaration.

```xml
<rdf:RDF
xmlns:owl =http://www.w3.org/2002/07/owl#
xmlns:rdf =http://www.w3.org/1999/02/22-rdfsyntax-ns#
xmlns:rdfs=http://www.w3.org/2000/01/rdfschema#
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#”>

Figure 4.2 Representation of Namespace Declaration

After rdf:RDF, some declarations to identify namespaces associated with this Ontology could be added. The effect of all of these namespaces is that such prefixes as the OWL and the RDF should be understood as referring to things drawn from following namespaces, as for example http://www.w3.org/2002/07/owl#.

In the OWL, classes are defined by using an owl:Class element, that is a subclass of the rdfs:Class. Figure 4.3 shows the representation of the SuperClass and SubClass relationship.

```xml
<owl:Class rdf:about="#Design">
  <rdfs:subClassOf rdf:resource="#Product_Engineering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string" >The translation of requirements into an effective design within project and operational constraints</rdfs:comment>
</owl:Class>

Figure 4.3 Sample Representation for SuperClass - SubClass relationship

One of the power elements of the OWL is an “owl:disjointWith”, which is missing from the RDFS, and is used to disjoint one class from the
others. “owl:equivalentClass” is another element that could be used to establish the equivalence between the classes. Last but not the least, there are two predefined classes, the owl:Thing (which defines everything) and the owl:Nothing, which is an empty set. A fragment of software risk identification ontology is shown in Figure 4.4.

```xml
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#">
   <owl:Ontology rdf:about=""/>
   <owl:Class rdf:ID="Dependencies">
     <rdfs:subClassOf>
       <owl:Class rdf:ID="Contract"/>
     </rdfs:subClassOf>
     <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
       This attribute refers to the possible contractual dependencies on outside contractors or vendors, customer-furnished equipment or software, or other outside products and services.
     </rdfs:comment>
   </owl:Class>
   ...
   <owl:FunctionalProperty rdf:about="#Risk_Factors">
     <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
       ...
     </rdfs:label>
   </owl:FunctionalProperty>
</rdf:RDF>
```

*Figure 4.4 (Continued)*
Figure 4.4 A Portion of OWL Representation of SRIONTO

In the OWL it is possible to have Boolean combinations such as a union, intersection, or complement of classes. For example, in the SRIONTO, it can be said that “Stability”, “Completeness”, and “Validity” are a collection of the “Risk_Factors”.

```xml
<!-- Created with Protege (with OWL Plugin 3.4, Build 125) -->
```

http://protege.stanford.edu -->
Figure 4.5 shows a portion of the OWL representation of the SRAONTO.

<?xml version="1.0"?>
<rdf:RDF
  xmlns="http://www.owl-ontologies.com/riskan.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-ontologies.com/riskan.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="External_Factors">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:ID="Factors"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:ID="is_influenced"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:ID="Micro_Factors"/>
                  <owl:Class rdf:ID="Macro_Factors"/>
                </owl:unionOf>
              </owl:Class>
            </owl:someValuesFrom>
          </owl:Restriction>
        </owl:Class>
      </owl:intersectionOf>
    </owl:equivalentClass>
  </owl:Class>
</rdf:RDF>

Figure 4.5 (Continued)
Figure 4.5 A Portion of the OWL Representation of the SRAONTO

In the OWL, classes are defined by using an owl:Class element that is a subclass of the rdfs:Class. The representation of a single class named “Factor” is shown in Figure 4.6:

```xml
<owl:Class rdf:ID="Factor">
  <rdfs:label>Factor</rdfs:label>
  <rdfs:comment>
    Factors that govern the process/project
  </rdfs:comment>
</owl:Class>
```

Figure 4.6 OWL Representation of a single class “Factor”

Figure 4.7 shows the OWL representation of the owl:disjointWith.
<owl:Class rdf:about="#others">
  <owl:disjointWith rdf:resource="#Markov_Modelling"/>
  <owl:disjointWith rdf:resource="#Go_Method"/>
  <owl:disjointWith rdf:resource="#Dynamic_event_tree_analysis_methodology"/>
  <owl:disjointWith rdf:resource="#Dynamic_Event_Logic_Analytical_Methodology"/>
  <owl:disjointWith rdf:resource="#Digraph"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Technique_for_Dynamic_Systems"/>
  </rdfs:subClassOf>
</owl:Class>

**Figure 4.7 OWL Representation of the owl:disjointWith**

The above owl representation says that, the concept “others” is a subclass of “Technique_for_Dynamic_Systems” and it is disjoint (“owl:disjointWith”) from “Markov_Modelling”, “Go_Method”, “Dynamic_event_tree_analysis_methodology”, ”Dynamic_Event_Logic_Analytical_Methodology" and “Digraph”.

SRAONTO also uses Boolean combinations such as a union, intersection, or complement of classes in its OWL representation. For example it can be said that “External_Factor” is a collection of the “Micro_Factor” and “Macro_Factors”. Figure 4.8 shows the OWL representation of the Boolean combinations “union”.

Another important Boolean relation supported by the OWL is its inverse properties to inherit domain and range. Figure 4.9 shows the OWL representation of the Boolean combinations “inverse”.

4.3 VISUAL REPRESENTATION OF THE SRMONTO USING THE ONTOVIZ

The OntoViz Tab configured in the protégé tool is used to visualize the software risk identification ontology with the help of a highly sophisticated graph visualization software called Graphviz.
4.3.1 Visualization of SRIONTO

The snapshot of visualization of the SRIONTO using the ontoviz in the protégé editor is presented in Figure 4.10.

Figure 4.10  The SRIONTO visualized using the ontoviz in the protégé editor

The visualizations are captured by the OntoViz embedded in the Protégé. The entire portion of the visualized ontology can be stored as image files by specifying the required path. Figure 4.11 shows the visual representation of a portion of SRIONTO.
Figure 4.11 Visualization of SRIONTO
4.3.2 Visualization of SRAONTO

The OntoViz Tab configured in the protégé tool is used to visualize the developed ontology with the help of the highly sophisticated graph visualization software called the Ontoviz. Figure 4.12 shows the snapshot of the visualization of the SRAONTO using the ontoviz in the protégé editor.

Figure 4.12 Visualization of the SRAONTO using the ontoviz in the protégé editor

Figure 4.13 shows the entire visual representation of the software risk analysis ontology.
Figure 4.13 Visual Representation of the SRAONTO
The complex, abstract, and interrelated contents so far represented verbally, are transformed into visual representations through the work of the visual representation of the SRMONTO.

4.4 KNOWLEDGE REPRESENTATION

Since ontologies provide semantic web agents with the background knowledge about domain concepts and their relationships, using it in knowledge representation will become more effective. A total of 234 source files have been automatically generated by the OWL document generator and each represents a concept in the SRMONTO. Figure 4.14 shows the snapshot of the main page used to represent the software risk management knowledge. On the left side, all the major areas of software risk management are displayed. The desired knowledge can be obtained by selecting the area. Here, the concept ‘Risk Analysis’ has been selected and the knowledge about the concept has been effectively represented. Figures 4.15 and 4.16 show the knowledge representations of the SRTONTO and SRCONTO respectively. This approach not only gives the hierarchical structure but gives four different types of knowledge. After selecting a particular concept, the name of the concept is displayed at the top as the ‘class’ name, followed by the semantic description of the concept. Then, the taxonomical arrangement of the domain starting from the top level concept to the selected concept, is displayed from the SRMONTO followed by the parent concept and then the descendents of the selected concept. The system will find out all the sub concepts of the selected concepts by selecting any one of the sub classes until no more sub concepts can be found.
Figure 4.14 Main page contains five major categories of the domain

Class: Risk_Analysis

The technique or process of assessing risks which provides insight into risks that an organization may face into future. It helps making decisions on the risks to lessen or risk analysis.

- Risk_Analysis

Usage

Class Description/Definition (Necessary Conditions)

Class Description/Definition (Necessary Conditions)

Class Description/Definition (Necessary Conditions)

Figure 4.15 Representation of the SRTONTO
**4.5 SUMMARY**

As a knowledge representation mechanism, this SRMonto facilitates visualizing domain knowledge based on widely available sources, and augments knowledge visualization approaches that focus on documents and concepts. Users can use this visual representation to discover patterns and make valuable connections between data. The hierarchical structure of the ontology and the visualization of the SRMonto make it natural to create a navigable hypermedia interface on its basis.

It is represented in OWL to codify the knowledge of the domain for agents’ processing of messages. The representation in the OWL for run-time communications between agents achieved a flexibility of modification and extendibility very well. However, it is realized that the OWL representation is
at a rather low level of abstraction. It is not very readable for domain experts to validate the ontology. Consequently the developed ontology has been processed again, semantically represented using the OWL document generator and visualized using the OntoViz.

The SRMONTO is evaluated both quantitatively and qualitatively using various metrics and parameters. The entire evaluation results are given in Chapter 8. The next chapter describes an ontology based e-learning system for software risk management, which is one of the applications of the SRMONTO.