A Semantic Web Approach to VTB Model Searching

By

Andrew Mark Finkbeiner

Bachelor of Science

University of South Carolina, 2000

Submitted in Partial Fulfillment of the

Requirements for the Degree of Master of Science in the

Department of Computer Science and Engineering

College of Engineering and Information Technology

University of South Carolina

2003

Department of Computer Science and Engineering Director of Thesis	Department of Computer Science and Engineering 2nd Reader
Department of Electrical Engineering 3rd Reader	Dean of the Graduate School

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr. Jose M. Vidal, for his patience, advice, and guidance. His insights helped motivate me and his direction kept me on track throughout this thesis. I would also like to thank Dr. Antonello Monti and Dr. Larry M. Stephens for being part of my thesis committee and providing guidance and suggestions to improve my work. I must also thank Dr. Roger Dougal, the director of the Virtual Test Bed project, for the opportunity to work as a research assistant.

Foremost, I would like to thank my family, whose constant love and support helped motivate and encourage me throughout my graduate school career.

ABSTRACT

The Semantic Web aims to encode semantics into web documents so that machines can meaningfully manipulate the data they contain. We have developed a system that uses concepts and tools from the Semantic Web to enable intelligent searching in our domain — a system-prototyping environment. We determine the semantic distance between objects from our domain and queries. This distance is used to determine what is the best match for a query. To determine the semantic-distance we use an instance-based learning method, specifically a K-Nearest neighbor approach is used. We place both the model and the query in N-dimensional space and compute the distance. To place the models in N-dimensional space we utilize a rule-engine that applies rules based on the semantics of an ontology language and rules specific to our domain. These rules add additional information to our knowledge base beyond that which can be extracted from the models. This additional information makes a more intelligent search possible.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Overview

The Virtual Test Bed (VTB) is a system for prototyping of large-scale, multi-technical dynamic systems. As VTB makes the transition from being a simulation environment to a design environment, adding intelligent behavior becomes increasingly important. The vision of the future VTB is an environment where users are able to develop large systems with varying levels of detail. In this situation, subsystems are independently developed, by different people, across an organization. In such a distributed work environment it is important that users be able to share the work they have done, in order to save time and money. It is also important that the design environment be able to help streamline the design process, by intelligently reasoning on the data. We believe that one of the first steps towards the development of an intelligent and efficient design environment is the ability to intelligently search for models.

The Semantic Web is a vision of what the Web might become. The idea behind the Semantic web is to encode semantics into web documents so that machines can

meaningfully manipulate its contents. Enabling computers to understand the content of the Web has many interesting benefits, one of which is improved searching capabilities.

We have developed a system that uses concepts and tools from the Semantic Web to enable intelligent searching in our domain — a system prototyping environment. It works by determining the semantic-distance between the models and a query. To determine the semantic-distance we use an instance based learning method, specifically a K-Nearest neighbor approach is used. We place both the model and the query in N-dimensional space and compute the distance. To place the models in N-dimensional space we utilize a rule-engine that applies rules based on DAML semantics, and rules specific to our domain. These rules add additional information to our knowledge base beyond that which can be extracted from the models. This additional information makes a more intelligent search possible.

1.2 Roadmap

The details of design, implementation, and testing of our system are presented in the following chapters. Chapter 2 presents some background to this work. It describes both technologies used and concepts that are important to understanding this work. Chapter 3 describes the system that was created. Chapter 4 explains the details of the matching algorithm and Chapter 5 discusses the testing of our system. Chapter 6 describes the analysis of the work and the future direction of this research.

CHAPTER 2

Background

2.1 Related Work

The work presented synthesizes techniques from disparate fields. It is important that a basic understanding of these tools and ideas are understood so that when their application in this work is described, the reader will have an understanding of their purpose. These techniques and tools are summarized in the following sections.

2.2 VTB

The Virtual Test Bed (VTB) is a research project primarily based at the University of South Carolina. VTB is a system for prototyping of large-scale multi-technical dynamic systems. It is a collection of software intended to enable users to simulate designs prior to hardware construction. The application driving the development of VTB is advanced power systems for navy platforms. The simulation environment has three major components: schematic editor, solvers, and models.

2.2.1 Schematic Editor

The Schematic Editor is a user interface for VTB that allows the creation and control of schematic diagrams. Figure 2.1 is a screenshot of Schematic Editor.

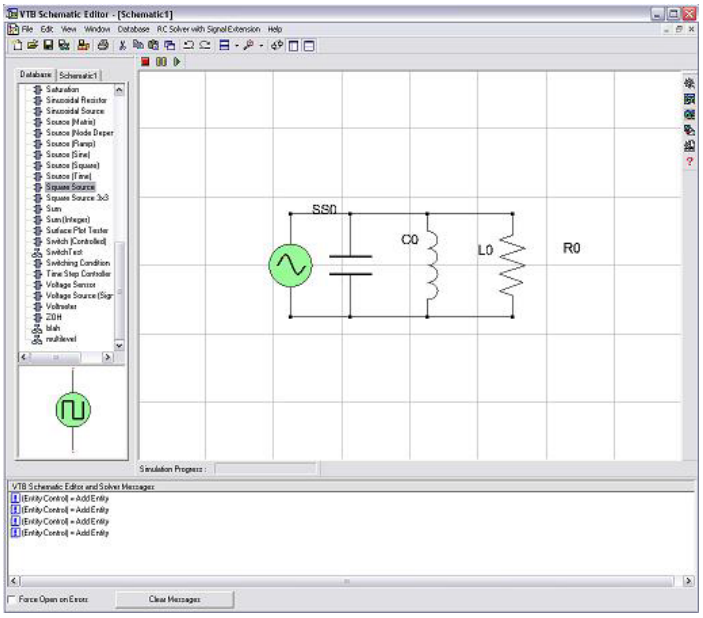

Figure 2.1. Schematic Editor.

This is the interface through which users manipulate models and solvers. Schematic editor is designed to be a generic simulation environment. It has a plug-in architecture that allows users to create their own solvers and models. Currently we have only one solver that is widely used. It is a mixed signal solver that uses both natural coupling and signal coupling. This solver is based on the principle of conservation of energy and uses matrix techniques in its solving engine. All models and solvers are individual dynamic link libraries that are loaded at runtime. Schematic editor will load a selected solver then

act as a communicator between the user and the solver. It tells the solver what to load, how entities (models) are connected, what their parameters are, and when to run and stop simulations.

2.2.2 Models

Models are the most important component to VTB because they are the building blocks of systems. There are three different types of models. There are core models, which are actual dll's coded in c++ that inherit from that VtbModel base-class. There are interpreted models, which are developed in other environments such as ACSL or MATLAB, and use a wrapper that communicates with VTB. And there are hierarchical-devices, which are models that are made up of other models. Figure 2.2 illustrates the concept of a hierarchical-device.

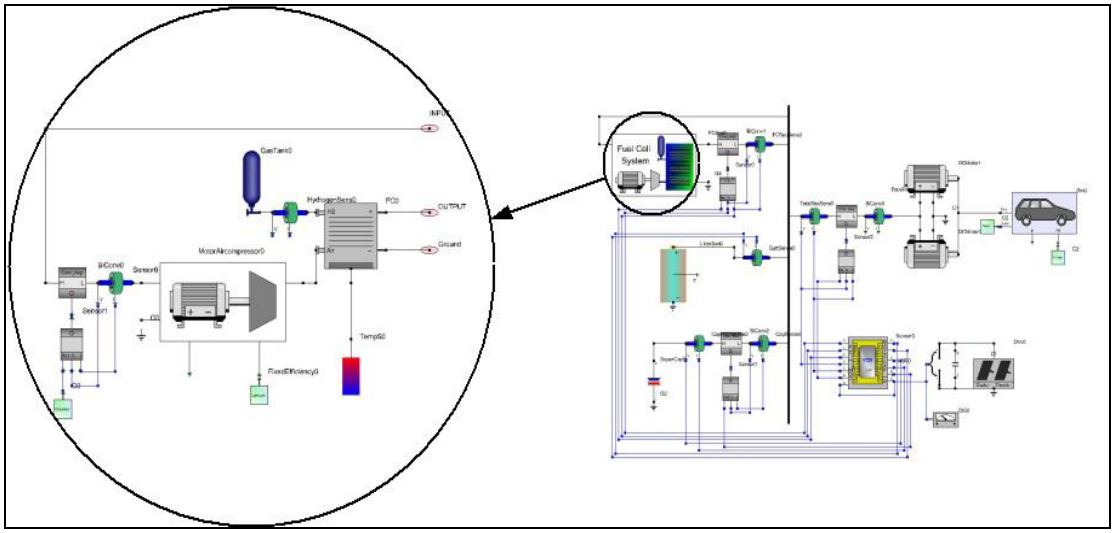

Figure 2.2. An Example of a Hierarchical-Device.

We are only concerned with only core models and hierarchical-devices for this work.

All core VTB models have the common base class VtbModel, which contains all of the necessary information that can be attributed to all VTB entities. The most important part of this class is the function GetEntityInformation(). This function returns a class that gives most of the important characteristics of the model such as ports, parameters, states, name, and description. Appendix A contains the VTBModel base class and associated classes. Hierarchical-devices are XML files based on a defined schema. The XML file contains basically the same information that a core model contains, along with information about the configuration of the models contained within it. An example hierarchical-device is listed in Appendix B.

2.3 The Semantic Web

The Semantic Web provides a vision of what the web might become. It hopes to bring structure to the meaningful content of the web. Today, most content on the web is designed for humans to read, not for computers to meaningfully manipulate. The Semantic Web will hopefully enable both humans and computers to process web content by encoding semantics into web documents [1].

Encoding semantics into web documents will enable computers to understand the meaning of the data in that document. This will create an environment where software agents will roam from page to page on the web and carry out sophisticated tasks for users [1]. The Semantic Web is not a separate web, it is an extension of the current web in

which information is combined with meaning. The proposed project will instantiate a part of the Semantic Web vision, as such, we view this prototype as a vehicle for analyzing the strengths and weaknesses of the various technologies proposed as part of the Semantic Web vision.

2.4 XML, RDF, RDFS, and DAML

For web pages to be encoded with semantics there must be languages that support this operation. This is where the eXtensible Markup Language (XML) and the Resource Description Framework come into play. XML is a markup language for documents containing structured information. XML provides a facility to define tags and the structural relationships between them. Programs or scripts can make use of these tags to process the information in these documents, but the programmer must have fore knowledge of what the tags mean, as the tags do not give meaning to the information [1].

RDF encodes semantic information as sets of triples. These sets of triples resemble the subject, verb, and object triples of an elementary sentence [2]. The basic RDF data model consists of three object types: resources, properties, and statements. Resources can be a collection of web pages, a single web page, a part of a web page, or an object that is not accessible on the web, such as a book. Resources are always named by Uniform Resource Identifiers (URIs), which are short strings that identify resources in the Web [3]. A property is a specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, defines its permitted values,

the types of resources it can describe, and its relationship with other properties [2, 4]. A statement is a specific resource together with a named property plus the value of that property for that resource. These three individual parts of a statement are called, respectively, the subject, the predicate, and the object. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML [2].

An example sentence:

Andy is the author of this document.

Which has the following parts:

Subject (resource)	this document
Predicate (property)	author
Object (literal)	Andy

Would be encoded in RDF/XML as:

```
<rdf:RDF>
        <rdf:Description about=”this document”>
                <author>Andy</author>
        </rdf:Description>
</rdf:RDF>
```

The triples of RDF can be used to describe the majority of data processed by machines. Because URI’s are used to encode the information anyone connected to the web has access to the definition.

RDF Schema (RDFS) builds on RDF by providing the mechanisms needed to specify classes and properties as part of a vocabulary. RDFS can be used to build a hierarchy of

classes and properties and then indicate which classes and properties are expected to be used together [5, 6].

The US Defense Advanced Research Projects Agency (DARPA) launched the Darpa Agent Markup Language (DAML) to support Semantic Web development [7, 8]. Like RDFS, DAML has the notion of classes and properties. But it extends RDFS with things such as equivalence ("childOf" on an English genealogy site is the same as "filsDe" on a French site), enumeration, and restrictions (cardinality) [9]. If you tell a computer something in DAML, it can use inference to give you new information based on that input. For example, if you tell your application:

```
(motherOf subProperty parentOf)
(Mary motherOf Bill)
```

Your application will conclude that:

```
(Mary parentOf Bill)
```

Being able to use inference to gain more knowledge about the data you're working with is very powerful. It will allow applications to do reasoning with web data that would otherwise not by possible.

2.5 Jena

Jena is a Semantic Web toolkit developed by Hewlett-Packard Laboratories. Jena provides an API for developers of Semantic Web applications. The main part of Jena is the RDF API. Figure 2.3 displays the basic architecture of Jena.

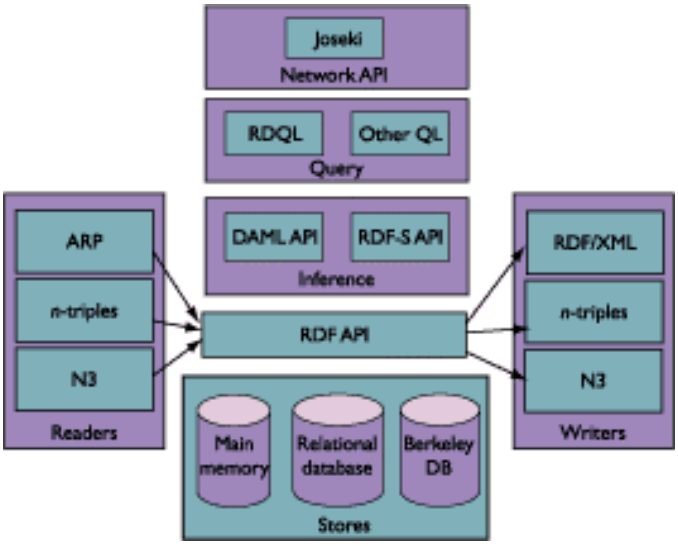

Figure 2.3. Jena Architecture

The RDF API supports the creation, manipulation, and query of RDF graphs. It also supports different storage technologies such as main memory and relational databases. Both the readers and writers are accessed through the RDF API. ARP is an RDF/XML parser that is included in Jena. The ARP parser is generated from the RDF/XML grammar by a compiler [10].

Three layers sit on top of the RDF API: inference, query, and networking. The inference layer is used to add information to the RDF graph. For example, RDF-S and DAML both implement the subclass-of relationship, which would add information to the existing RDF

graph. The current version of Jena only supports limited inference, but a more sophisticated reasoner is in development [10]. Jena also has a query language for RDF graphs called RDQL. RDQL is an easily understood language that is a great model to follow as it is closely related to SquishQL, which is based on an open source database language RdfQL [10, 11]. The network layer allows an application to query and update a remote RDF database [10]. Jena provides a great API for developing applications for the semantic web, but it will not be complete until it fully supports inference.

2.6 Jess

Jess is a tool for building Expert Systems. An Expert System is a set of rules that can be repeatedly applied to a collection of facts about the world. Rules that apply are fired, or executed. Jess uses a special algorithm called Rete to match the rules to the facts [12]. The Jess language is based on CLIPS, which is a highly specialized form of LISP [13]. Jess has some important features that are used in this work. They are briefly described in the following sections.

2.6.1 Facts

A rule-based system such as Jess maintains a collection of facts. These facts, or knowledge base, are a core concept in Jess. The following is an example of a fact contained in the Jess knowledge base.

```
(PropertyValue  rdf:type  rdf:Literal  rdfs:Class)
```

For this work we either assert a fact or use the *deffact* construct. The difference between the two is that a fact declared with *deffact* is asserted whenever a *reset* command is called, a fact that is just asserted is lost when reset is called.

2.6.2 Defrules

A Jess rule is something like an if…then statement. Jess rules are executed whenever their if parts, or left-hand-sides (LHS) are satisfied. An example of a rule is shown below.

```
(defrule do-get-gas
   "If gas is low, go to gas station."
   (gas-is-low)
=>
   (go-to-gas-station))
```

This rule has two parts, separated by the "=>" symbol, which you can be read as *then*. The first part consists of the LHS pattern, `gas-is-low`. The second part consists of the RHS *action* `(go-to-gas-station)`. The LHS of a rule consists of patterns, which are used

to match facts in the knowledge base, while the RHS contains function calls. When the fact 'gas-is-low' is asserted into the knowledge base the RHS of the rule will be executed.

2.6.3 Defquery

A defquery is a special Jess rule that has no RHS. A defquery is how we find information in the Jess knowledge base. Lets assume that the following facts are the current knowledge base of Jess.

```
color jetta red
color mx6 blue
color pathfinder green
color civic green
make pathfinder nissan
make jetta vw
```

The following defquery can find cars based on their color.

```
(defquery find-color
    (declare (variables ?color))
    (color ?car ?color))
```

If we execute the query with a color argument of *green* Jess will match the following facts.

```
color pathfinder green
color civic green
```

We can extend our query to also include the make of the car.

```
(defquery find-color-and-make
    (declare (variables ?color ?make))
    (color ?car ?color)
    (make ?car ?make))
```

Now we can execute a query specifying both the make and color of the car we are looking for.

2.6.4 Deffunction and Userfunction

The deffunction construct and the Userfunction java interface allow us to define our own Jess functions. The Userfunction interface is the most interesting as it allows us to program functions for Jess in the Java programming language. Userfunction adds a lot of power to Jess as it enables programmers to extend Jess to accomplish their tasks. A simple example of a use of Userfunction is shown in figure 2.4.

```
import jess.*;

public class StringSearch implements Userfunction
{
        public String getName(){return "string-search"}

        public Value call(ValueVector vv, Context context)
        {
                String arg1 = vv.get(1).stringValue(context);
                String arg2 = vv.get(2).stringValue(context);
                if(arg.compareTo(match) == 0)
                        return new Value(true);
                else
                        return new Value(false);
        }
}
```

Figure 2.4. Example of Jess Userfunction interface.

This function simply checks to see if one string is equal to another. Even though this example is not useful, it does displays how simple it is to extend the functionality of Jess.

2.7 DAMLJessKB

DAMLJessKB is a tool for reasoning with the semantic web. It is a description logic reasoner for performing inference with DAML [14]. It combines both Jena and Jess to create a knowledge base from data encoded with DAML. First, it uses an RDF parser, from the Jena toolkit, to create a stream of triples. These triples are then asserted into Jess. Then, it runs rules derived from the semantics of DAML to populate the knowledge base with additional facts. Figure 2.5 displays an overview of the DAMLJessKB process [14].

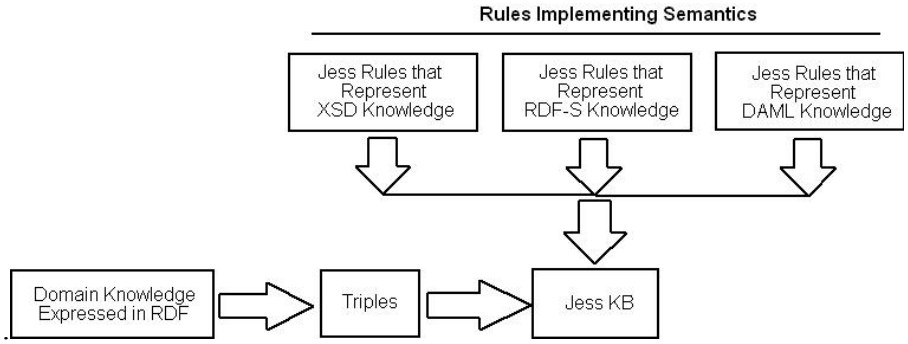

Figure 2.5. DAMLJessKB Process

An example of a Jess rule that defines the rdfs:subClassOf language construct is shown below. This rule states that an instance of a subclass is an instance of the parent class.

```
(defrule subclass-instances
    "This enforces and makes meaningful the rdfs:subClassOf relationship."
    (PropertyValue rdfs:subClassOf ?child ?parent)
    (PropertyValue rdf:type ?instance ?child)
=>
    (assert
        (PropertyValue http://www.w3.org/1999/02/22-rdf-syntax-ns#type
                       ?instance
                       ?parent)))
```

Using all of the additional facts that are created using rules such as those in Listing 1 the knowledge base can then be queried. The listing below displays what a simple query may look like. This query asks for the type of an object, X, specified by the user.

```
(defquery query-type-by-subject
    "Query the type by object"
    (declare (variables ?x))
    (PropertyValue http://www.w3.org/1999/02/22-rdf-syntax-ns#type ?x ?y))
```

DAMLJessKB makes excellent use of existing tools to create a useful tool for Semantic Web applications.

2.8 WSDL and UDDI

The Web Services Description Language (WSDL) provides a model and an XML format for describing Web services. WSDL defines what a web service does, how it does it, and how it can be used by clients. WSDL describes Web services starting with the messages that are exchanged between the service provider and requestor. The messages themselves are described abstractly and then bound to a concrete network protocol and message format. A message consists of a collection of typed data items. An exchange of messages between the service provider and requestor are described as an operation. A collection of operations is called a port type. A service contains a collection of ports, where each port is an implementation of a portType, which includes all the concrete details needed to interact with the service [15].

The Universal Description, Discovery and Integration (UDDI) project is an industry effort to define a registry where users can discover the services they need. The UDDI has a registry of all the web service's metadata and a a set of WSDK type information for searching the registry [16, 17]. Conceptually, the information provided in a UDDI business registration consists of three components: "white pages" including address, contact, and known identifiers; "yellow pages" including industrial categorizations based on standard taxonomies; and "green pages", the technical information about services that are exposed by the business. To use UDDI developers register their web service with the UDDI business registry. Logically the UDDI business registry is centralized, but physically it is distributed with multiple root nodes that replicate data with each other [17]. Once a service is registered other people or businesses can see that web service, and they can get all of the information that is needed to use it.

2.9 PartMiner

PartMiner is a global supplier of electronic components. They provide component information that is needed by engineers and purchasers in the electronics industry. PartMiner and Electronics Workbench, the makers of MultiSim, have partnered to create a way to generate models from the parts datasheet parameters retrieved from

PartMiner[20]. PartMiner has a subscription service that allows users to do complex searches for components. Users can search by part number, manufacturer name , technical characteristics, keyword and example parametrics(mathematical modeling). They also have cross reference searches such as: upgrade, downgrade, and similar [21]. PartMiner is a great example of the types of searches that are important to potential users of this work.

CHAPTER 3

Architecture

3.1 Overview

The Semantic Web aims to add semantics to the information available on the web. Information encoded with semantics enables machine processing of that information. When machines are able to process and 'understand' information intelligent programs can perform tasks independent of user input. This automation could make the Web a much more effective environment.

One area in which the Semantic Web could greatly increase Web functionality is in searching for information. Currently web searches are based on keyword matching and various statistical methods. It is likely that a search utilizing semantics will result in a much more accurate and useful response.

A wide range of languages such as RDF, DAML and OWL have been developed to enable the semantic markup of information. There are also RDF parsers, ontology

editors, and various other tools that have been developed in the hopes that the Semantic Web vision will become a reality.

This application integrates existing Semantic Web technologies to enable intelligent searching of VTB models. We use the DAML ontology language to describe the models used in the VTB simulation environment and then use DAML-enabled technologies to provide more intelligent model query results. A reasoning-engine is the 'brains' of our method; it uses both the semantics of the DAML language and domain information to facilitate this advanced search.

3.2 System Components

We have built a model searching toolkit based on Semantic Web ideas and tools. This application attempts to provide users of the VTB simulation environment with a model finding tool that will enable them to more efficiently design complex systems. The basic overview of our system is a displayed in figure 3.1.

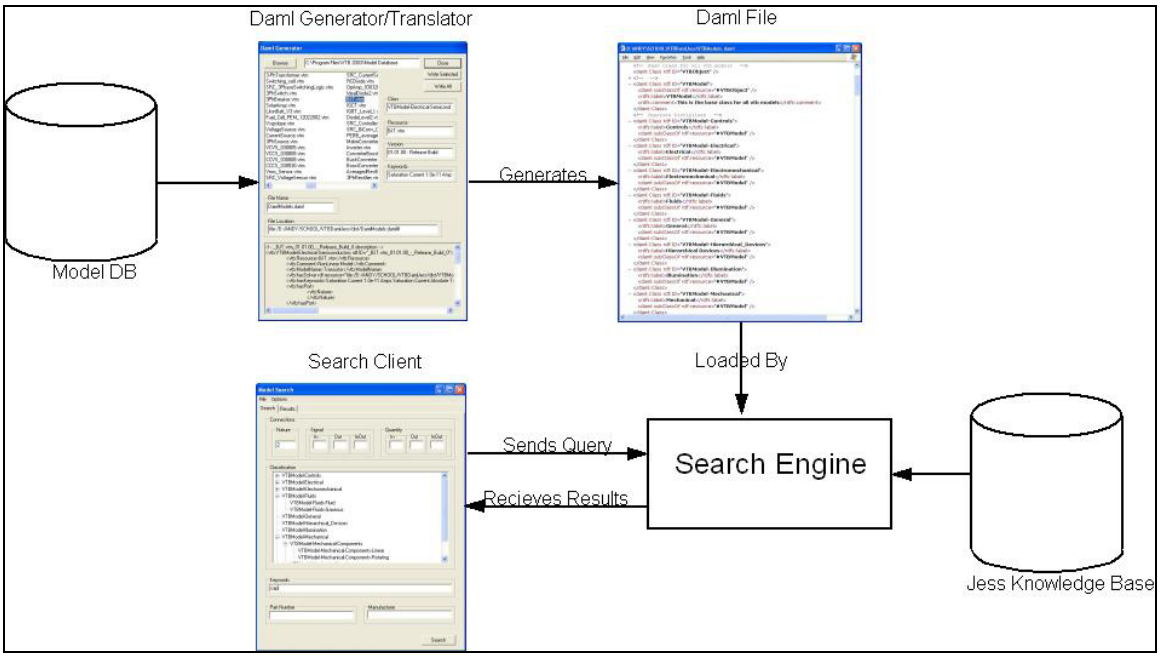

Figure 3.1. System Overview.

The three main components of out system are the DAML Generator/Translator, the Search Client, and the Search Engine.

3.2.1 DAML Generator/Translator

The DAML Generator transforms a VTB model database into a DAML file representing those models. It is the first step in the process since the DAML file containing the model information is what the search engine uses. This tool also gives the user the flexibility to edit and add information pertaining to the models.

3.2.2 Search Client

The Search Client is the interface through which the user communicates with the search engine. It is the visual part of the project and handles the communication between the user and the search engine. Its sole purpose is to send queries and receive results.

3.2.3 Search Server

This is the core component of the system. It contains Jess, which is the reasoning-engine of our system, and DAMLJessKB, which provides the Jess rules that define the DAML ontology language. The search server utilizes these tools along with the Jena ARP parser to create a knowledge base used in our search. Our search server uses the knowledge contained in Jess to do a semantic-distance search to determine the best match to a query. We use an instance based learning approach to match our query. Specifically we use a K-Nearest neighbor based algorithm with post processing to combine it with keyword matching.

3.3 Design Considerations

3.3.1 DAML

DAML is an ontology language that enables machines to intelligently process information by describing the semantics with which the information is encoded. DAML extends existing languages RDF and RDFS by adding additional semantics. For this work RDFS would have been sufficient, but DAML was chosen because it allows restrictions on properties and classes making it a more complete semantic language. The OWL web ontology language is the most recent language and will most likely replace DAML as the standard ontology language. The differences between OWL and DAML are minimal; the main difference is OWL has no ties to the department of defense, unlike DAML. Since DAML provides all the features needed for this work and has existing tools associated with it, it remained our choice as the markup language for this application.

3.3.2 DAMLJessKB

DAMLJessKB was chosen as our reasoning tool because of its support for DAML semantics and it's use of Jess as a reasoning engine. Although some changes were made to the source code of DAMLJessKB, the functionality and the core intelligence, specifically the Jess rules describing DAML, remained the same.

3.3.3 Jess

Jess is the intelligence in our system. It supports the development of rule-based expert systems, which enable many interesting behaviors. Jess is the inference engine behind DAMLJessKB, and we further utilized its capabilities by adding domain specific rules to enhance our searching ability. The Jena Semantic Web toolkit comes with an inference engine, but it is not as powerful as Jess and was therefore not considered an option [10]. The extendibility and power of Jess makes it an ideal tool for our work.

CHAPTER 4

Implementation

4.1 Overview

The core idea behind this work is that by utilizing semantics, a more intelligent search mechanism can be achieved. We use a rule engine so that we can infer new data, from existing data, based on expert knowledge and the semantics of the ontology language we use. Our method of searching uses the concept of *semantic distance.* For this work semantic distance is used as a measure to describe how similar one object is to another. We compute the semantic distance from a DAML query to a DAML VTB model. By using DAML to describe our models we can use semantics to give a more accurate search result.

4.2 VTB Model Ontology

The VTB Model ontology is the first step in our system. The ontology describes the relationships between models, and the structure of their data. A complete listing of the

VTB model ontology can be found in Appendix C, but throughout this section we will present pieces of the ontology as we explain it's significance.

One important aspect of the ontology is that it describes the class of a model. For example a model of a resistor is a passive, electrical component. Figure 4.1 displays a minimal DAML file that defines the classification of a resistor model along with the DAML instance of the model.

```
<daml:Class rdf:ID="VTBObject"/>
<daml:Class rdf:ID="VTBModel">
        <daml:subClassOf rdf:resource="#VTBObject"/>
        <rdfs:label>VTBModel</rdfs:label>
</daml:Class>
<daml:Class rdf:ID="Electrical">
        <rdfs:label>Electrical</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>
<daml:Class rdf:ID="Passives">
        <rdfs:label>Passives</rdfs:label>
        <daml:subClassOf rdf:resource="#Electrical"/>
</daml:Class>
<vtb:Passives rdf:ID="Resistor.vtm"/>
```

Figure 4.1. Example DAML file that defines an instance of a resistor model.

Before any rules are run with the information we put into our system, the only facts we have about that class are that Resistor.vtm is of type Passives. But once the rule that defines the daml:subClassOf construct fire, we get the additional information that Resistor.vtm is also of type Electrical, VTBModel, and VTBObject. While this is obvious to humans, a machine does not understand this concept until we tell it this relationship, which is exactly what an ontology allows us to do.

The class structure is the first and most basic inference that we use in this work. Figure 4.2 presents a condensed classification class structure of the VTB model ontology.

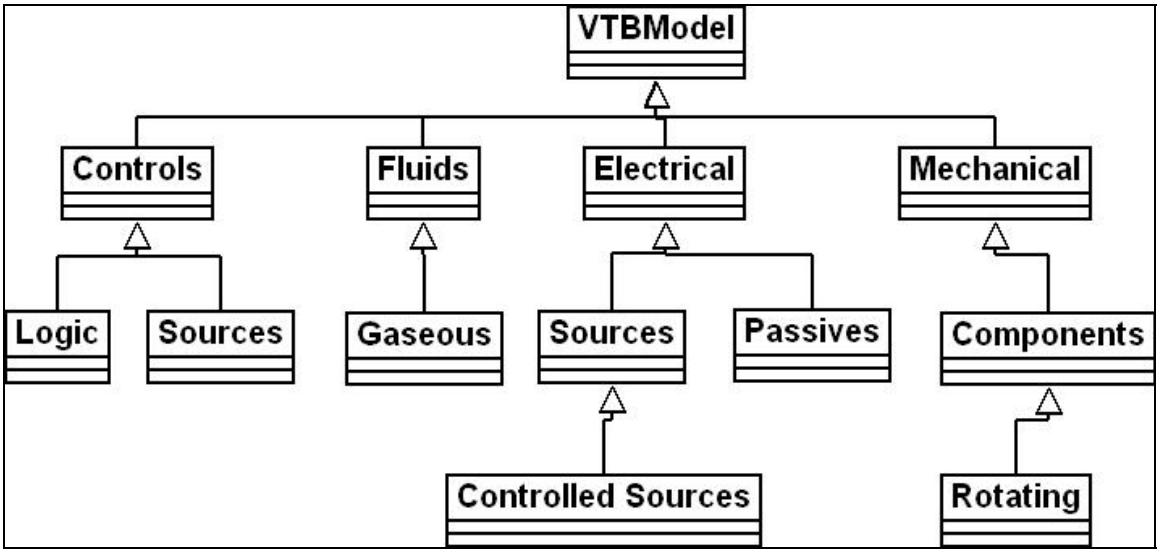

Figure 4.2. VTB Classification Structure.

The classification structure displayed in Figure 4.2 is only a partial diagram of the classification structure of the ontology. Figure 4.3 is the DAML representation of the classification structure shown in figure 4.2.

```
<daml:Class rdf:ID="VTBModel">
        <daml:subClassOf rdf:resource="#VTBObject"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Controls">
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Electrical">
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Fluids">
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Mechanical">
        <daml:subClassOf rdf:resource="#VTBModel"/>
```

```
</daml:Class>
<daml:Class rdf:ID="VTBModel-Controls-Logic">
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Controls-Sources">
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Electrical-Passives">
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Electrical-Sources">
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Fluids-Gaseous">
        <daml:subClassOf rdf:resource="#VTBModel-Fluids"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Mechanical-Components">
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Electrical-Sources-Controlled_Sources">
        <daml:subClassOf rdf:resource="#VTBModel-Electrical-Sources"/>
</daml:Class>
<daml:Class rdf:ID="VTBModel-Mechanical-Components-Rotating">
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical-Components"/>
</daml:Class>
```

Figure 4.3. Model Classification Structure.

Refer to Appendix C if you wish to see the complete ontology, as it is quite large and cannot all be listed here.

Besides describing the classification class structure the ontology gives form to other important data contained in models. It describes the properties of models and places restrictions on those properties. A prime example of this is the description of connections. Connections have a base class called Port, which has three subclasses; Nature, Signal, and Quantity. Restrictions are placed on both the signal and quantity

ports that they must have both a type and a direction. Figure 4.4 is the section of the ontology that describes the model connections.

```
<daml:Class rdf:ID="Port"/>
<daml:Class rdf:ID="Nature">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Signal"/>
        <daml:disjointWith rdf:resource="#Quantity"/>
</daml:Class>
<daml:Class rdf:ID="Signal">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Nature"/>
        <daml:disjointWith rdf:resource="#Quantity"/>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasDirection"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasType"/>
                </daml:Restriction>
        </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Quantity">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Signal"/>
        <daml:disjointWith rdf:resource="#Nature"/>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasDirection"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasType"/>
                </daml:Restriction>
        </daml:subClassOf>
</daml:Class>
```

Figure 4.4. Ontology Description of Connections.

The significance of this part of the ontology is that we now have a description of what model connections are. With this description and rules supplied by us we can both validate that we have a VTB model, and we can have intelligent software reason on information described in this way.

The VTB model ontology also describes other important information about models such as: parameters, keywords, part number, manufacturer, and in the special case of hierarchical devices, what models it contains. Now that we have a way to describe models using DAML, Jena can convert our ontology and instances of models into triples, which can be asserted into our rule-engine.

4.3 DAML Creator / Translator

Converting a VTB model database to a DAML file is the first step in our search process. Figure 4.5 is a screenshot of the program that does that handles the DAML generation/translation.

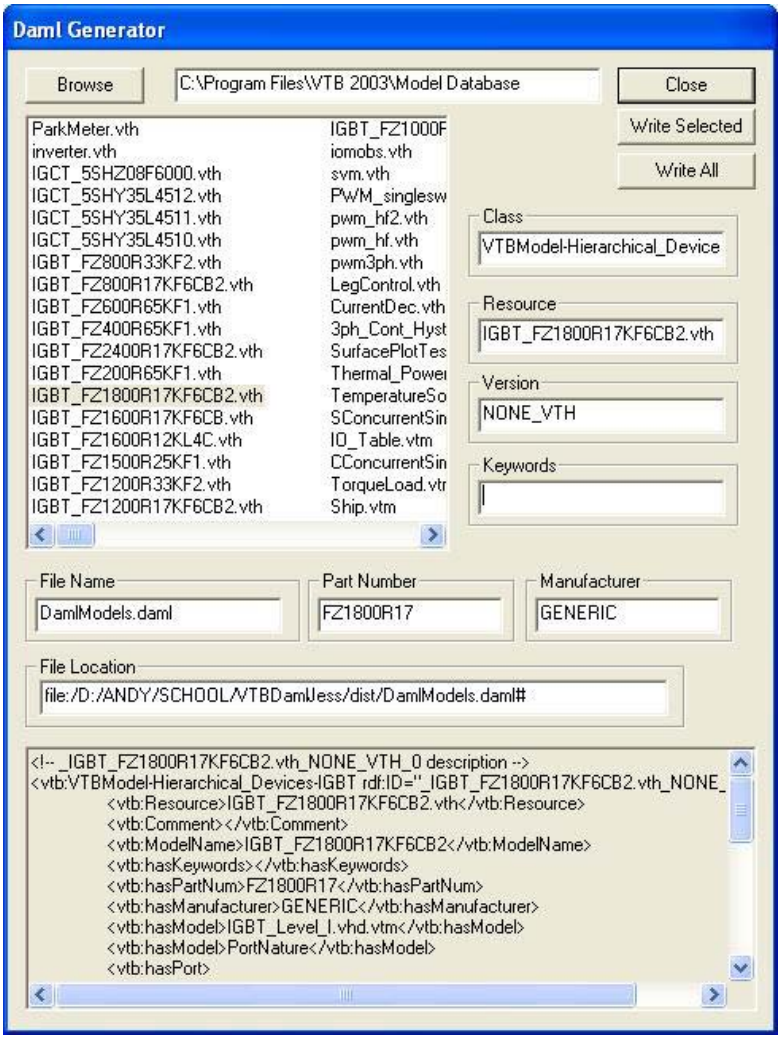

Figure 4.5. DAML Generator Screenshot.

The DAML generator takes as an argument a VTB model database, which happens to be a directory path. It uses the folder structure of the model database to determine model classifications, and it extracts information from the models, so the user has to input as little information as possible. Certain attributes such as the classification, resource name, version, keywords, part number, and manufacturer can be modified. The DAML Generator also gives the option of either writing selected models to a file, or writing the

entire database. There is also a preview of the DAML model that will be written, to ensure that the desired output is correct.

Because VTB models are built using Microsoft Visual C++ 6.0, the DAML Generator was built using the same compiler so that it would link with the proper libraries and be able to load the models. The DAML generator also makes use of the Xerces XML parser for the extraction of information from hierarchical devices. The following listing is an example the DAML instance generated for a model by this tool.

```
<!-- _SRC_IntegerDoubleConversion.vtm_01.01.00_-_Release_Build_1 description -->
<vtb:VTBModel-Controls-Transformations rdf:ID="IntegerDoubleConversion.vtm">
        <vtb:Resource>SRC_IntegerDoubleConversion.vtm</vtb:Resource>
        <vtb:Comment>Integer-Double Converter Model</vtb:Comment>
        <vtb:ModelName>Integer ->Double</vtb:ModelName>
        <vtb:hasSolver rdf:resource="#VtbRCSignalExSolver"/>
        <vtb:hasKeywords>Integer- Double true  Integer to Double OR Double To Integer Double-
Integer conversion mode 0  0 -  Round ; 1 -  Ceiling ; 2 -  Floor Integer - Double Integer - Double ID
Integer-Double Converter Model </vtb:hasKeywords>
        <vtb:hasPort>
                <vtb:Signal>
                        <vtb:hasDirection rdf:resource="#INPUT"/>
                    <vtb:hasType rdf:resource="#DOUBLE"/>
                </vtb:Signal>
        </vtb:hasPort>
        <vtb:hasPort>
                <vtb:Signal>
                        <vtb:hasDirection rdf:resource="#OUTPUT"/>
                    <vtb:hasType rdf:resource="#DOUBLE"/>
                </vtb:Signal>
        </vtb:hasPort>
        <vtb:hasParameter rdf:resource="#IntegerDoubleConversion.vtm_Param_1"/>
</vtb:VTBModel-Controls-Transformations>
vtb:Parameter rdf:ID=”IntegerDoubleConversion.vtm_Param_1">
        <vtb:ParameterName>Double->Integer conversion mode</vtb:ParameterName>
        <vtb:ParameterValue>0</vtb:ParameterValue>
        <vtb:ParameterUnits></vtb:ParameterUnits>
        <vtb:ParameterComment>1 -> Ceiling ; 2 -> Floor</vtb:ParameterComment>
        <vtb:hasType rdf:resource="#INTEGER"/>
</vtb:Parameter>
```

4.4 Query Interface

The query interface is a GUI application that allows a user to search for VTB models. Figure 4.6 displays a snapshot of this application.

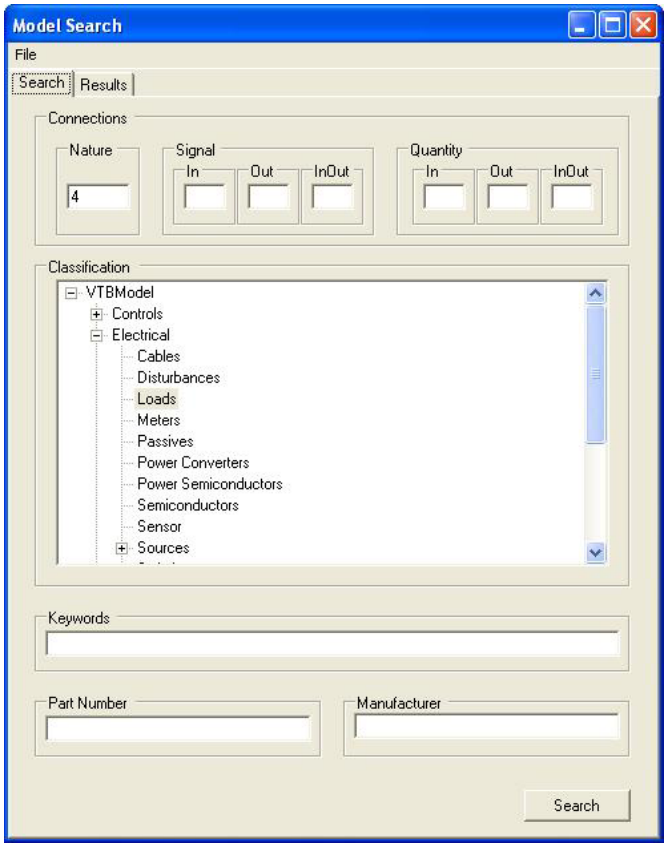

Figure 4.6. Model Search GUI.

The model search GUI is a client to the search server in our system. It sends queries to the search engine based on user input, and receives and displays the results. The communication with the search server is achieved through simple networking code using TCP/IP socket programming. The model search GUI was written in Microsoft C#

because it provides a simple GUI construction process, simple network coding, and built in XML serialization.

The query interface allows the user to specify model connections, classification, keywords, part number, and manufacturer in its search. This initial search capability allows simple searches based on partial instances of a DAML query. More interesting is the similarity search that can be done after an initial search is completed and a model selected. A similarity search uses a complete DAML instance when trying to find a semantic distance. This will be clarified in the following section, which discusses the search engine.

4.5 Search Engine

The search engine is the core piece of our system. It contains Jess, which is our rule engine and knowledge base. The search engine is written in Java and communicates with the query interface using TCP/IP sockets. The search engine loads our VTB model ontology along with any instances of VTB DAML models, initializes our search mechanism, and determines the semantic distance between models and queries.

Strategy 4.5.1

Our strategy for searching is to determine a semantic distance between VTB DAML model queries and instances of VTB DAML models. To accomplish this task we elected

to classify VTB models by placing them in N-Dimensional space. While this may be an intuitive approach to determining semantic distance, it is not an easy approach. To help us construct our feature vectors and determine their values we use a rule engine, Jess, along with rules that describe both DAML semantics and domain characteristics. These rules help us to generate facts, which enable us to achieve our goal of placing models in N-Dimensional space.

Our approach to determining semantic distance fits very nicely into the machine learning topic of instance based learning. Our specific approach to instance based learning is the K-Nearest neighbor algorithm, although, in our work we simply use K = 1. In K-Nearest neighbor classification you simply place the example instances in N-dimensional space, and then place the query in N-Dimensional space. The query is then classified as the average of its K-nearest neighbors. The two main problems with this approach are that some data does not naturally fit into a feature vector, and irrelevant attributes can affect distance [22]. Both of these problems can be overcome, the first with creativity and the second by weighting features to modify their effect on the distance.

4.5.2 DAML Rules

DAMLJessKB supplies the DAML rules and initial facts the Jess uses. These rules provide a general level of inference that can by used by any DAML ontology, regardless of its domain. These rules define classes and how they relate to one another, and what restrictions are placed on them. The number of rules and facts that implement the DAML

semantics are many so we will just cover a few of the most relevant rules. The entire listing of rules defined by DAMLJessKB can be found Appendix D. The listing below is one of the core semantic rules utilized by this work.

```
(defrule subclass-instances
  "An instance of a subclass is an instance of the parent class. This
   enforces and makes meaningful the daml:subClassOf relationship."

  (PropertyValue daml:subClassOf ?child ?parent)
  (PropertyValue rdf:type ?instance ?child)
=>
 (assert
  (PropertyValue rdf:type ?instance ?parent)
 )
)
```

The subclass-instances rule enforces and makes meaningful the daml:subClassOf relationship. The listing below is an example of this rule in action.

```
Facts in knowledge base:
    (PropertyValue daml:subClassOf Electrical VTBModel)
    (PropertyValue daml:subClassOf Electrical Sources)
    (PropertyValue daml:subClassOf Sources Generic_Source)
    (PropertyValue rdf:type voltageSource GenericSource)
Facts added to knowledge base as a result of 'defrule subclass-instances':
    (PropertyValue rdf:type voltageSource Sources)
    (PropertyValue rdf:type voltageSource VTBModel)
```

DAMLJessKB also defines important facts, for example:

```
  (PropertyValue rdf:type rdf:Resource rdfs:Class)
  (PropertyValue rdf:type rdf:type rdf:Property)
  (PropertyValue rdf:type rdf:Property rdf:Resource)
  (PropertyValue rdf:type rdf:Property rdfs:Class)
  (PropertyValue rdf:type rdfs:Class rdfs:Class)
  (PropertyValue rdf:type daml:subClassOf rdfs:subClassOf)
  (PropertyValue rdfs:subClassOf daml:subClassOf rdfs:subClassOf)
```

4.5.3 Domain Rules

The domain rules that we define give our application the ability to perform basic reasoning on our domain. Our rules utilize the information extracted from the models to infer some basic information that helps us to classify models. Most of the information may seem trivial to humans with a general knowledge of VTB or system design experience, however, a program must be told explicitly how to deal with the data it has, if anything resembling reasoning is to be done.

Our first rules deal with connections and their relationship to one another. What the first three rules do is generalize the type and number of ports used, to the supported back plane. For example, a model that has a signal in and a signal out port supports the signal backplane. This benefits our query mechanism by placing models with similar backplane support closer to each other. The three rules that generalize the connection information are listed below.

A model with signal ports supports the signal backplane A model with nature ports supports the nature backplane A model with quantity ports supports the quantity backplane

These rules will help give better results to queries when users know the functionality that they want from a model and the type of connections but not the specific number of ports.

Other domain rules make use of model parameters to draw correlations between model instances. An instance of a model parameter has parameter units, names, and values.

Using this information we are able to reason that models that have a frequency parameter are most likely somewhat similar to each other. And obviously there are many such units that can be defined, a few examples are: resistance, inductance, time, power, volume, impedance, etc… An example of a Jess rule that implements this is shown in Figure 4.7.

```
(defrule has-power
        (PropertyValue rdf:type
                        ?model
                       http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
         (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
         (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                        ?param
                        ?literal)
         (PropertyValue rdf:value
                        ?literal
                        ?unit)
         (test (or (string-search ?unit watt) (string-search ?unit watts)))
=>
         (assert
                (PropertyValue HAS_POWER ?model ?param)
         )
)
```

Figure 4.7. Jess Listing of the has-power Rule.

The domain rules although simple in concept greatly increase the sophistication of our search process. The rules put intelligence into our system and enable a more complete, and hopefully relevant, classification of models. The Complete listing of domain rules can be found in Appendix E.

4.5.4 Jess Queries

The Jess defquery construct is the method by which we see what facts are in the knowledge base. To accomplish our search we extensively use specific queries to determine where a model is placed in N-Dimensional space. We also use queries in conjunction with our domain rules to assert new facts. A complete listing of all queries used in this work can be found in Appendix F.

Queries in Jess are fast as long as they are written correctly. The number of partial matches generated, is the main factor that determines the performance of Jess. To ensure that Jess runs optimally the most specific patterns, those matching the fewest facts, should be at the top of a query. This is not always convenient as it forces us to break up our queries into two separate queries so that we can receive the information we are looking for. For example Figure 4.8 executes nearly twice as fast as Figure 4.9.

```
//execute some java code here
result = QUERY(get_param_unit_lit)
unit = QUERY( literal_value, result)
 //end java code
 (defquery get_param_unit_lit
          (declare (variables ?param))
          (PropertyValue    vtb:ParameterUnits
                                ?param
                                ?value))
(defquery literal_value
          (declare (variables ?literal))
          (PropertyValue  rdf:value
                              ?literal
                              ?comment))
```

Figure 4.8 An Efficient Parameter Unit Search.

```
//execute some java code here
unit = QUERY(get_param_unit)
//end java code
 (defquery get_param_unit
          (declare (variables ?param))
           (PropertyValue  rdf:value ?literal ?comment)
          (PropertyValue    vtb:ParameterUnits ?param ?literal))
```

Figure 4.9 An Inefficient Parameter Unit Search.

It's also important to note that if we are doing a query to find a certain piece of information; that information must be in the first fact of the RHS of the query, as that is the Fact that Jess returns. For Example in Figure 4.9; *(PropertyValue rdf:value ?literal ?comment),* is that fact that is returned so if you want to find the string value of a parameter unit that has to be on top, which in turn matches many more facts than *(PropertyValue vtb:ParameterUnits ?param ?literal)*, which has a drastic negative effect on the performance of Jess.

Listed below is a simple Jess query that returns all nature ports from a model.

```
(defquery check-nature-backplane
        (declare (variables ?model))
        (PropertyValue vtb:hasPort  ?model  ?Port)
        (PropertyValue rdf:type       ?Port     vtb:Nature)
)
```

This query takes as an argument a model name and returns a list of results. It's important to note that the interface through which we query Jess is in our Java source code. The basic process is you execute a run-query command in Jess and store the query result. After the query is run we then fetch an iterator to our query result. Below is a listing of the Java code that implements this.

```
public Iterator query(String queryAndArgs){
    rete.executeCommand(queryAndArgs); //rete is Jess
    Iterator q = (Iterator) rete.fetch("QUERY-RESULT").externalAddressValue(null);
    while (q.hasNext()){
        Fact f = ((jess.Token) res.next()).fact(1);
        ValueVector list = fact.get(0).listValue(null);
        String pred = list.get(0).stringValue(null);
        String subj = list.get(1).stringValue(null);
        String obj = list.get(2).stringValue(null);
    }
}
```

As you can see in the code above, a Jess query can store the resulting lists of facts, which can easily be retrieved through our java code.

4.5.5 Model Classification

Our method of searching hinges on our ability to use semantics and a rule engine to help us classify models. A challenge that we face is using the data in the model to place it in N-Dimensional space. From the model we can extract connection information, parameters and their properties, and keywords. We use this information along with the database hierarchy in which the model is located to classify the models.

To classify our models we place them in N-Dimensional space. Their distance from one another is a measure of their similarity, and hence is our form of classification. We use feature-vectors to describe our models and place them in N-Dimensional space. There are three main sections of our feature-vector. The first describes the connections of the model. Figure 4.10 illustrates the process from model to DAML instance to the connection feature-vector.

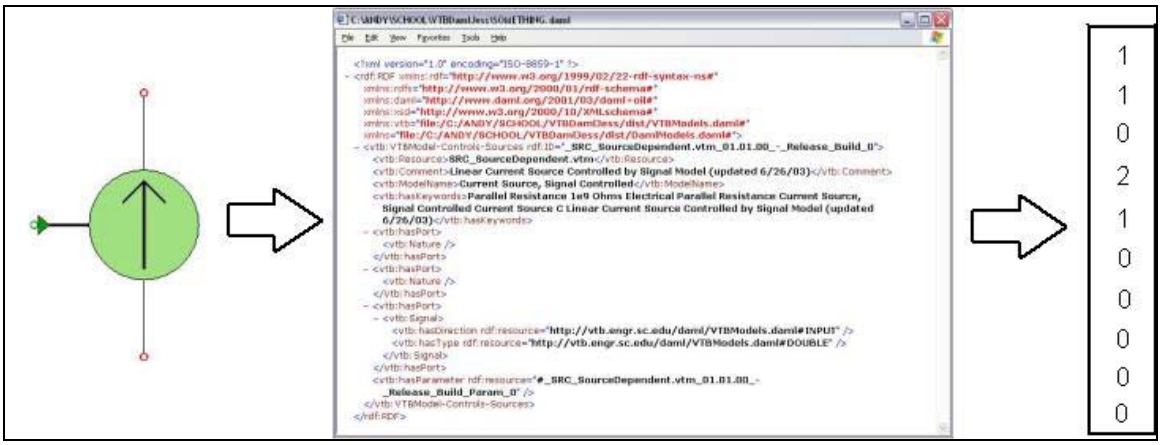

Figure 4.10. Model to Feature Vector Process.

The first three places in the vector specify the supported backplanes of the model, and the remaining 7 places indicate the number of each specific type of port. All values of the feature vector are determined by querying the Jess Knowledge base.

The next section of the classification vector deals with the model classification with respect to the model database. For example a resistor is a Passive model, which is a subclass of an electrical model, which is a subclass of a VTB model. The classification feature-vector is basically a Boolean vector the length of all subclasses of VTBObject defined by our ontology. To determine the value of a models classification feature vector a Jess query is run for each classification to determine if a model is of that specific type. When a model has a specific type it's value for that type in the feature vector is set to one.

The last section of the feature vector is composed of features relating to the parameters of models. This part of the feature vector, like the classification feature-vector, is also a Boolean vector. To construct this feature-vector we use the facts asserted by our domain specific rules. Each domain rule corresponds to an index into the feature vector; this is illustrated in Figure 4.11.

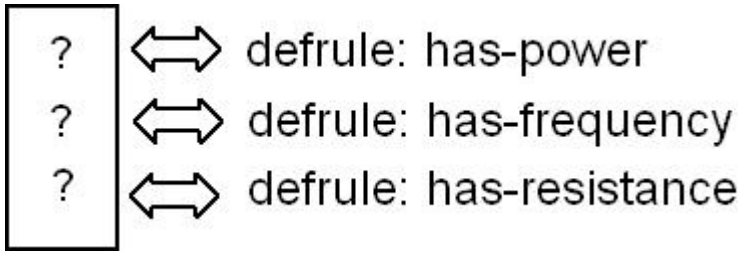

Figure 4.11. Domain Rules Feature-Vector.

For this application we have ten different rules that are used for ten indices into our vector.

Another important point is how H-devices are placed in N-dimensional space. Each h-device has models that are contained within it. To take advantage of this knowledge in our classification we take an average of all of the models feature vectors that that h-devices has. We then add a percentage of that average to the h-devices actual feature vector. This way we preserve some of the information of the models that it contains but reduce their affect on the classification of the h-device.

4.5.6 Semantic Distance Computation

To determine the semantic-distance between a model and a query we use a K-Nearest neighbor approach. The basic process is first; all relevant DAML files are loaded into Jess. Jess is then run, generating new facts about the models. The facts in the knowledge base are then used to place the models in N-dimensional space. The placing of models in N-dimensional space is the most time consuming part of our search process, it happens only once, when the search server is initialized.

Once the models have all been a given a place in N-dimensional space they are ready to be compared to queries. Queries and model instances are both of the same form. From a simple query string, sent by the query interface tool, a DAML query instance is

constructed. The query is asserted into Jess so that inferences can be made on the information. Then it goes through the same initialization process as the DAML models, so that it's location in N-dimensional space can be calculated. Finally we apply any weights we have to the vectors and calculate the final distance between our query and all of the model instances. Figure 4.12 illustrates the search process.

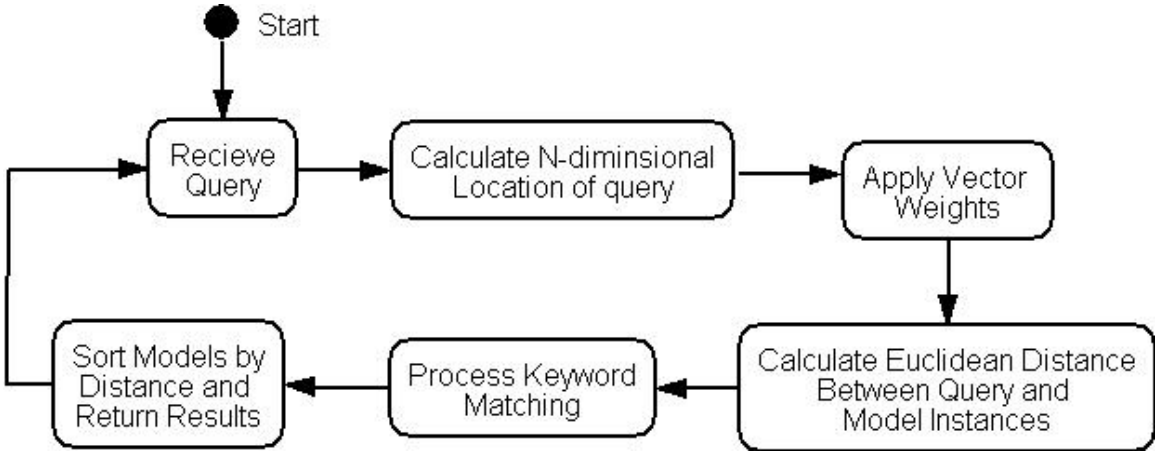

Figure 4.12. Search Process.

Figure 4.13 defines how the distance between our feature vectors is computed.

If $a = (x_1, x_2, \ldots, x_n)$ and $b = (y_1, y_2, \ldots, y_n)$

then Euclidean distance can be defined as

$$d(a,b) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}.$$

Figure 4.13. Euclidean Distance Definition.

When determining the distance between a query and model instances it is often the case that the query instance is incomplete. By incomplete, I mean that not all of the

dimensions are represented. For example, in the following query only classification is specified in the query.

```
<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:xsd = "http://www.w3.org/2000/10/XMLschema#"
         xmlns = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
  <VTBModel-Controls rdf:ID="SEARCH">
  </VTBModel-Controls>
</rdf:RDF>
```

When this happens all irrelevant parts of the feature vector are collapsed. This means that in the case of the above query the parts of the vector that deal with connections and parameters are not used in the distance calculations. This is done so that those portions of the vector do not have a negative effect on the search result. Since the query has no information pertaining to connections and parameters they would have values of zero in the feature vector. This would result in a distance potentially far greater than should be, since model instances contain all available information and would not have vectors of zero.

4.5.7 Post-Processing Steps

The final step is in our process of determining semantic distance is to adjust the distance based on keyword matching. There are three different query inputs that use this feature; keywords, part number, and manufacturer. This part of our search is very basic. We simply give values for complete and partial matches, and subtract that from the computed distance.

CHAPTER 5

Tests

5.1 Query Interface

The Model Search interface is the default way for users to interact with the Search Server, and it is how all tests were performed. Figure 5.1 is a combination of two screenshots; the left side is the search interface, and the right side is the results page.

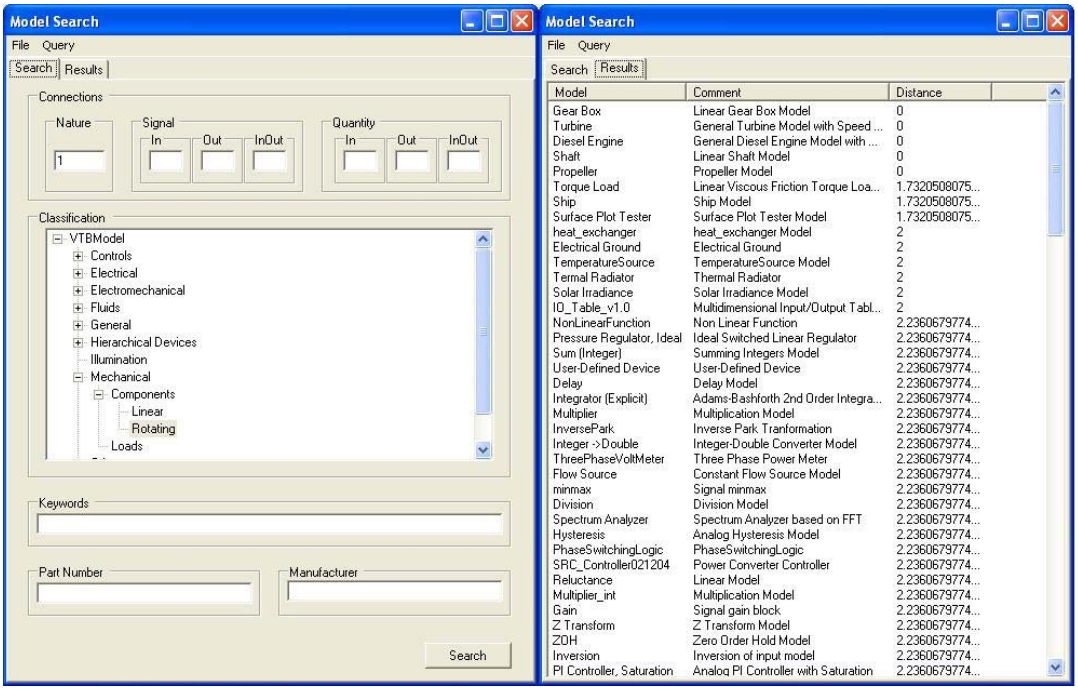

Figure 5.1. Model Search Interface.

The Model Search interface also allows the option of a similarity search. Double-clicking a model in the results list will create a dialog box with has the option of doing a similarity search. Figure 5.2 is a picture of that dialog.

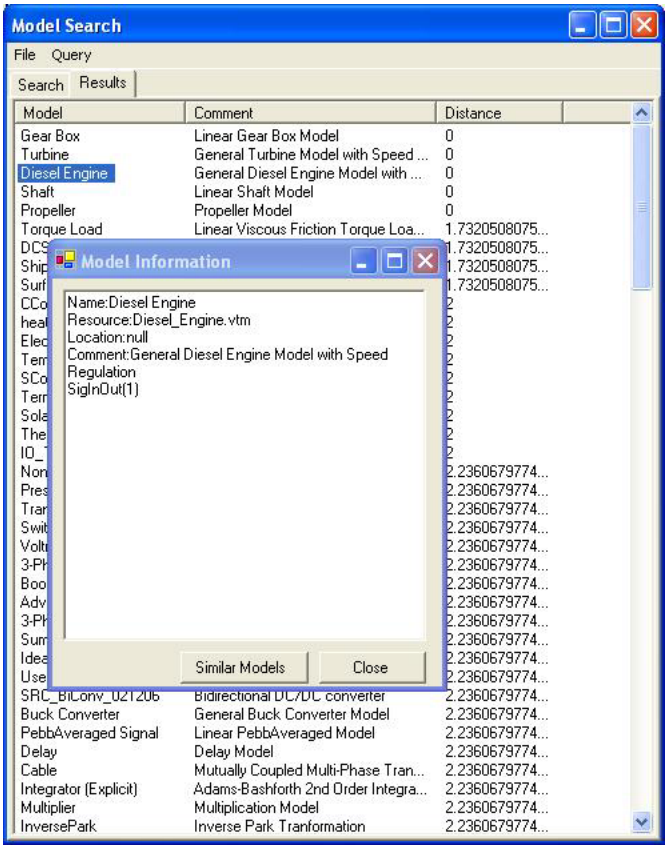

Figure 5.2. Model Information Dialog.

There is one more input option available in the Model Search interface. You can also directly send a DAML instance to the server. This was added to test queries that agents

might send to the search server. Figure 5.3 shows an example of a DAML instance being sent, this dialog is accessed through the main menu 'Query->DAML Instance'.

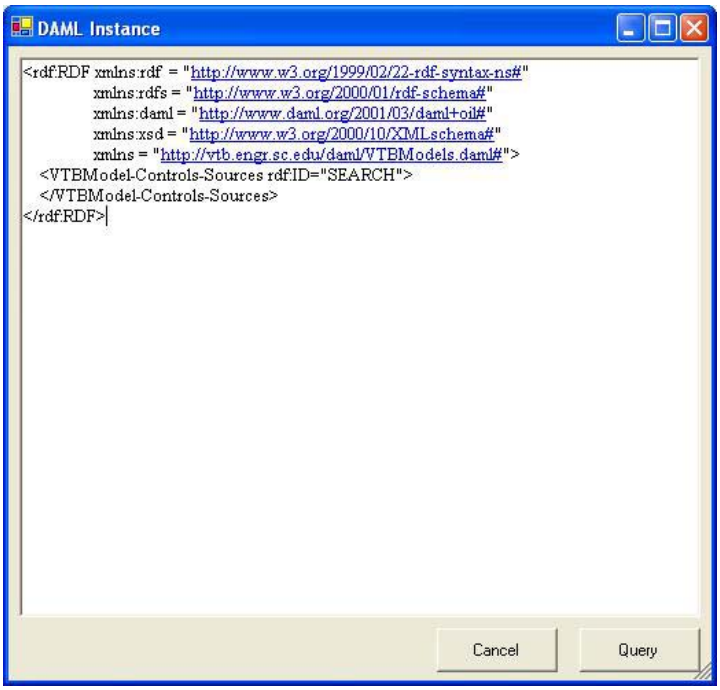

Figure 5.3. DAML Instance Input.

5.2 Search Engine

The search engine needs to be told what DAML databases it is searching. We have created a large DAML database using the DAML Generator/Translator. Our search engine loads this database along with the VTB Model ontology.

DAML files:

1. http://vtb.engr.sc.edu/daml/VTBModels.daml
2. http://vtb.engr.sc.edu/daml/DAMLModels.daml

Queries from the user interface program are sent to the search server. In the search server the query is translated into a DAML instance. This DAML instance is then asserted into Jess, and it is placed in N-Dimensional space in the same manner as models from the DAML database. In the case of a similarity search an instance already exists, so there is no need to add any information to Jess. The distances are calculated between the query and the models and the results are sent back to the query interface and displayed in the results tab.

5.3 Test Query 1

The following query is a simple query that that uses both DAML rules and domain specific rules to enable a more sophisticated query result.

User Interface Selection:

Connections: -none-

Classification: VTBModel – Controls – Sources

Keywords: -none-

Part Number: -none-

Manufacturer: -none-

DAML Instance:

```
<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:xsd = "http://www.w3.org/2000/10/XMLschema#"
         xmlns = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
  <VTBModel-Controls-Sources rdf:ID="SEARCH">
  </VTBModel-Controls-Sources>
</rdf:RDF>
```

Results:

Model	Comment	Distance from Query
step.vtm	Step	0.0
SRC_SourceNodeDep.vtm	Source (Node Dependent)	0.0
SRC_SourceSinusoidal.vtm	Source (Sine)	0.0
SRC_SourceMatrix.vtm	Source (Matrix)	0.0
SRC_Constant.vtm	Constant	0.0
SRC_SourceDependent.vtm	Current Source, Signal Controlled	0.0
SRC_SourceDependentV.vtm	Voltage Source, Signal Controlled	0.0
lookuptable.vtm	Look-up table	0.0
SRC_SourceRamp.vtm	Source (Ramp)	0.0
SRC_SourceTime.vtm	Source (Time)	0.0
SRC_SourceSquare.vtm	Source (Square)	0.0
triangularsource.vtm	triangularsource	0.0
CurrentSource.vtm	Current Source	2.23606797749979
VoltageSource.vtm	Voltage Source	2.23606797749979
VCVS_030805.vtm	VCVS	2.23606797749979
LiIonBatt_V3.vtm	Lithium-ion Battery (V3.0)	2.23606797749979
CCCS_030530.vtm	CCCS	2.23606797749979
CCVS_030808.vtm	CCVS	2.23606797749979
3PhSource.vtm	3-Phase Source	2.23606797749979
VCCS_030808.vtm	VCCS	2.23606797749979

This type of query is a basic search a user might execute. A search that specifies a classification makes use of the inference capabilities supplied by Jess. As you can see from the query instance our search was of type VTBModel-Controls-Sources, so by the

DAML subclass rule we can infer that the search is also of type VTBModel-Controls, and VTBModel. This is all information that is used when placing our search instance in N-Dimensional space. This query also makes use of the following domain specific rule.

```
(defrule sources
        (or
        (PropertyValue rdf:type
                ?thing
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-Controls-Sources)
        (PropertyValue rdf:type
                ?thing
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-ElectroMechanical-Motors)
        (PropertyValue rdf:type
                ?thing
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-Electrical-Sources)
        )
=>
        (assert (PropertyValue rdf:type ?thing VTB_SOURCE))
)
```

This rule helps us to draw a correlation between sources from different disciplines. This demonstrates an important feature of our method of searching. We are able to add rules that increase our programs knowledge of a domain so that more advanced results can be achieved. If we simply used our inference capabilities we would not get results from such different paths such as VTBModel-Electrical-Sources, since there are models that are much closer to VTBModel-Controls-Sources.

5.4 Test Query 2

The following queries demonstrate the use of keyword matching. These queries do not make use of Jess as a rule engine, but they are important as a post-processing step for searches that use Jess, and are an important search interface for the user.

5.4.1 Keywords

User Interface Selection:

Connections: -none-

Classification: -none-

Keywords: cap

Part Number: -none-

Manufacturer: -none-

DAML Instance:

```
<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:xsd  = "http://www.w3.org/2000/10/XMLschema#"
          xmlns      = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
  <VTBModel rdf:ID="SEARCH">
    <hasKeywords>capacitor </hasKeywords>
  </VTBModel>
</rdf:RDF>
```

Results:

Model	Comment	Distance from Query
LiIonBatt_V3.vtm	Lithium-ion Battery (V3.0)	-1.0
Capacitor.vtm	Capacitor	-1.0
SuperCap.vtm	SuperCap	-0.25
simulink11.vtm	Simulink11	0.0
step.vtm	Step	0.0
NonLinearFunction.vtm	NonLinearFunction	0.0
TorqueLoad.vtm	Torque Load	0.0
Regulator_Ideal_020409.vtm	Pressure Regulator, Ideal	0.0
3-PhTransformer.vtm	Transformer	0.0
CurrentSource.vtm	Current Source	0.0
Switching_cell.vtm	Switch_cell	0.0
Voltmeter.vtm	Voltmeter	0.0
CConcurrentSim_030326.vtm	CConcurrentSim	0.0

VoltageSource.vtm	Voltage Source	0.0
3PhRectifier.vtm	3-Phase Rectifier	0.0
LabviewWrapper.vtm	LabVIEW Wrapper	0.0
BoostConverter.vtm	Boost Converter	0.0
SRC_CurrentSensor.vtm	Current Sensor	0.0
IGBT_FZ1000R25KF1.vth	IGBT_Level_I_FZ1000R25KF1	0.0
AdvProgLoad_030816.vtm	AdvProgLoad_v1.2	0.0

5.4.2 Part Number

User Interface Selection:

Connections: -none-

Classification: -none-

Keywords: -none-

Part Number: FZ1600

Manufacturer: -none-

DAML Instance:

```
<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:xsd  = "http://www.w3.org/2000/10/XMLschema#"
xmlns      = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
<VTBModel rdf:ID="SEARCH4">
<hasPartNum>FZ1600</hasPartNum>
</VTBModel>
</rdf:RDF>
```

Results:

Model	Comment	Distance from Query
IGBT_FZ1600R12KL4C.vth	IGBT_FZ1600R12KL4C	-0.5
IGBT_FZ1600R17KF6CB.vth	IGBT_FZ1600R17KF6CB	-0.5

simulink11.vtm	Simulink11	0.0
step.vtm	Step	0.0
NonLinearFunction.vtm	NonLinearFunction	0.0
TorqueLoad.vtm	Torque Load	0.0
Regulator_Ideal_020409.vtm	Pressure Regulator, Ideal	0.0
3-PhTransformer.vtm	Transformer	0.0
CurrentSource.vtm	Current Source	0.0
Switching_cell.vtm	Switch_cell	0.0
Voltmeter.vtm	Voltmeter	0.0
CConcurrentSim_030326.vtm	CConcurrentSim	0.0
VoltageSource.vtm	Voltage Source	0.0
3PhRectifier.vtm	3-Phase Rectifier	0.0
LabviewWrapper.vtm	LabVIEW Wrapper	0.0
BoostConverter.vtm	Boost Converter	0.0
SRC_CurrentSensor.vtm	Current Sensor	0.0
IGBT_FZ1000R25KF1.vth	IGBT_Level_I_FZ1000R25KF1	0.0
AdvProgLoad_030816.vtm	AdvProgLoad_v1.2	0.0
VCVS_030805.vtm	VCVS	0.0

5.4.3 Manufacturer

User Interface Selection:

Connections: -none-

Classification: -none-

Keywords: -none-

Part Number: -none-

Manufacturer: fugi

DAML Instance:

```
<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:xsd  = "http://www.w3.org/2000/10/XMLschema#"
xmlns          = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
<VTBModel rdf:ID="SEARCH1">
<hasManufacturer>fugi</hasManufacturer>
```

```
</VTBModel>
</rdf:RDF>
```

Results:

Model	Comment	Distance from Query
IGBT_FZ1000R25KF1.vth	IGBT_Level_I_FZ1000R25KF1	-1.0
IGBT_FZ1500R25KF1.vth	IGBT_FZ1500R25KF1	-1.0
IGBT_FZ1200R17KF6CB2.vth	IGBT_FZ1200R17KF6CB2	-1.0
IGBT_FZ1200R33KF2.vth	IGBT_FZ1200R33KF2	-1.0
simulink11.vtm	Simulink11	0.0
step.vtm	Step	0.0
NonLinearFunction.vtm	NonLinearFunction	0.0
TorqueLoad.vtm	Torque Load	0.0
Regulator_Ideal_020409.vtm	Pressure Regulator, Ideal	0.0
3-PhTransformer.vtm	Transformer	0.0
CurrentSource.vtm	Current Source	0.0
Switching_cell.vtm	Switch_cell	0.0
Voltmeter.vtm	Voltmeter	0.0
CConcurrentSim_030326.vtm	CConcurrentSim	0.0
VoltageSource.vtm	Voltage Source	0.0
3PhRectifier.vtm	3-Phase Rectifier	0.0
LabviewWrapper.vtm	LabVIEW Wrapper	0.0
BoostConverter.vtm	Boost Converter	0.0
SRC_CurrentSensor.vtm	Current Sensor	0.0
AdvProgLoad_030816.vtm	AdvProgLoad_v1.2	0.0

5.5 Test Query 3

The third query is a test that uses the similar-to search feature. It makes use of all feature vectors since it is a complete DAML instance, unlike a query that uses just classification, or just connection information. Because it uses all feature vectors in it's comparison, it makes use of all rules we have defined for Jess. However, it does not use the keyword string matching.

User Interface Selection:

Similar To: IGBT_Level_I.vtm

DAML Instance:

```
<vtb:VTBModel-Electrical-Power_Semiconductors
      rdf:ID="_IGBT_Level_I.vtm_01.01.00_-_Release_Build_0">
    <vtb:Resource>IGBT_Level_I.vtm</vtb:Resource>
    <vtb:Comment>NonLinear Model</vtb:Comment>
    <vtb:ModelName>IGBT_Level_I</vtb:ModelName>
    <vtb:hasPort>
      <vtb:Nature />
    </vtb:hasPort>
    <vtb:hasPort>
      <vtb:Nature />
    </vtb:hasPort>
    <vtb:hasPort>
      <vtb:Nature />
    </vtb:hasPort>
    <vtb:hasPort>
      <vtb:Nature />
    </vtb:hasPort>
    <vtb:hasParameter rdf:resource="#_IGBT_Level_I.vtm_01.01.00_-
      _Release_Build_Param_0" />
<!--hasParameters 2 – 19 not shown-->
    <vtb:hasParameter rdf:resource="#_IGBT_Level_I.vtm_01.01.00_-
      _Release_Build_Param_20" />
  </vtb:VTBModel-Electrical-Power_Semiconductors>
 <vtb:Parameter rdf:ID="_IGBT_Level_I.vtm_01.01.00_-_Release_Build_Param_0">
    <vtb:ParameterName>Gate Resistance</vtb:ParameterName>
    <vtb:ParameterValue>13.0</vtb:ParameterValue>
    <vtb:ParameterUnits>ohms</vtb:ParameterUnits>
    <vtb:ParameterComment>Gate Resistance</vtb:ParameterComment>
    <vtb:hasType rdf:resource="http://vtb.engr.sc.edu/daml/VTBModels.daml#REAL"
      />
  </vtb:Parameter>
   <!--parameters 2 – 19 not shown-->
  <vtb:Parameter rdf:ID="_IGBT_Level_I.vtm_01.01.00_-_Release_Build_Param_20">
    <vtb:ParameterName>Tj_max</vtb:ParameterName>
    <vtb:ParameterValue>423</vtb:ParameterValue>
    <vtb:ParameterUnits>Kelvins</vtb:ParameterUnits>
    <vtb:ParameterComment>Maximum junction
      temperature</vtb:ParameterComment>
    <vtb:hasType rdf:resource="http://vtb.engr.sc.edu/daml/VTBModels.daml#REAL"
      />
  </vtb:Parameter>
```

Results:

Model	Comment	Distance from Query
DiodeLevel2.vtm	Diode(Level-2)	0.0
IGBT_Level_I.vtm	IGBT_Level_I	0.0
3PhBreaker.vtm	3Phase Breaker	1.4142135623730951
IGBT_FZ1000R25KF1.vth	IGBT_Level_I_FZ1000R25KF1	1.9364916731037085
IGBT_FZ800R33KF2.vth	IGBT_FZ800R33KF2	1.9364916731037085
IGBT_FZ1800R17KF6CB2.vth	IGBT_FZ1800R17KF6CB2	1.9364916731037085
IGBT_FZ400R65KF1.vth	IGBT_FZ400R65KF1	1.9364916731037085
IGBT_FZ2400R17KF6CB2.vth	IGBT_FZ2400R17KF6CB2	1.9364916731037085
IGBT_FZ600R65KF1.vth	IGBT_FZ600R65KF1	1.9364916731037085
IGBT_FZ200R65KF1.vth	IGBT_FZ200R65KF1	1.9364916731037085
IGBT_FZ1500R25KF1.vth	IGBT_FZ1500R25KF1	1.9364916731037085
IGBT_FZ1200R17KF6CB2.vth	IGBT_FZ1200R17KF6CB2	1.9364916731037085
IGBT_FZ800R17KF6CB2.vth	IGBT_FZ800R17KF6CB2	1.9364916731037085
IGBT_FZ1200R33KF2.vth	IGBT_FZ1200R33KF2	1.9364916731037085
IGBT_FZ1600R12KL4C.vth	IGBT_FZ1600R12KL4C	1.9364916731037085
IGBT_FZ1600R17KF6CB.vth	IGBT_FZ1600R17KF6CB	1.9364916731037085
IGCT.vtm	IGCT	2.0
Switching_cell.vtm	Switch_cell	2.449489742783178
3PhSwitch.vtm	3-Phase Switch	2.449489742783178
IdealDiode2.vtm	Ideal Diode	2.449489742783178

The results of this query are significant because it returns hierarchical-devices (h-devices). H-devices are a special case in our classification, since they take into account the models contained inside of them. This test demonstrates that our method of handling h-devices works reasonably well.

5.6 Test Query 4

The fourth query is a test that also uses the similar-to search feature. It differs from test query 3 in that it doesn't have any h-devices that are related to it, and it utilizes our

domain specific rules to a greater extent. It still uses all feature vectors in its comparison, and does not use the keyword string matching.

Similar To: ProgrammableLoad.vtm

DAML Instance:

```
<vtb:VTBModel-Electrical-Loads rdf:ID="_ProgrammableLoad.vtm_01.01.00_-
_Release_Build_0">
        <vtb:Resource>ProgrammableLoad.vtm</vtb:Resource>
        <vtb:Comment>Programmable Load</vtb:Comment>
        <vtb:ModelName>Programmable Load</vtb:ModelName>
        <vtb:hasPort>
            <vtb:Nature />
        </vtb:hasPort>
        <vtb:hasPort>
            <vtb:Nature />
        </vtb:hasPort>
        <vtb:hasParameter rdf:resource="#_ProgrammableLoad.vtm_01.01.00_-
              _Release_Build_Param_0" />
       <!--hasParameters 1 – 14 not shown-->
        <vtb:hasParameter rdf:resource="#_ProgrammableLoad.vtm_01.01.00_-
              _Release_Build_Param_15" />
</vtb:VTBModel-Electrical-Loads>
<vtb:Parameter rdf:ID="_ProgrammableLoad.vtm_01.01.00_-_Release_Build_Param_0">
        <vtb:ParameterName>LoadType</vtb:ParameterName>
        <vtb:ParameterValue>2</vtb:ParameterValue>
        <vtb:ParameterUnits>N/A</vtb:ParameterUnits>
        <vtb:ParameterComment>Power</vtb:ParameterComment>
        <vtb:hasType
              rdf:resource="http://vtb.engr.sc.edu/daml/VTBModels.daml#INTEGER" />
</vtb:Parameter>
   <!--Parameters 1 – 14 not shown-->
<vtb:Parameter rdf:ID="_ProgrammableLoad.vtm_01.01.00_-_Release_Build_Param_15">
        <vtb:ParameterName>Enable Initial Condition</vtb:ParameterName>
        <vtb:ParameterValue>true</vtb:ParameterValue>
        <vtb:ParameterUnits>N/A</vtb:ParameterUnits>
        <vtb:ParameterComment />
        <vtb:hasType
              rdf:resource="http://vtb.engr.sc.edu/daml/VTBModels.daml#BOOLEAN" />
</vtb:Parameter>
```

Results:

Model	Comment	Distance from Query

AdvProgLoad_030816.vtm	AdvProgLoad_v1.2	0.0
ProgrammableLoad.vtm	Programmable Load	2.0
Reluctance.vtm	Reluctance	2.0
Fault.vtm	Fault	2.692582403567252
Voltmeter.vtm	Voltmeter	2.8722813232690143
Resistor.vtm	Resistor	2.8722813232690143
TorqueLoad.vtm	Torque Load	3.0
Ship.vtm	Ship	3.0
IGBT_FZ1000R25KF1.vth	IGBT_Level_I_FZ1000R25KF1	3.1231056256176606
IGBT_FZ800R33KF2.vth	IGBT_FZ800R33KF2	3.1231056256176606
IGBT_FZ1800R17KF6CB2.vth	IGBT_FZ1800R17KF6CB2	3.1231056256176606
IGBT_FZ400R65KF1.vth	IGBT_FZ400R65KF1	3.1231056256176606
IGBT_FZ2400R17KF6CB2.vth	IGBT_FZ2400R17KF6CB2	3.1231056256176606
IGBT_FZ600R65KF1.vth	IGBT_FZ600R65KF1	3.1231056256176606
IGBT_FZ200R65KF1.vth	IGBT_FZ200R65KF1	3.1231056256176606
IGBT_FZ1500R25KF1.vth	IGBT_FZ1500R25KF1	3.1231056256176606
IGBT_FZ1200R17KF6CB2.vth	IGBT_FZ1200R17KF6CB2	3.1231056256176606
IGBT_FZ800R17KF6CB2.vth	IGBT_FZ800R17KF6CB2	3.1231056256176606
IGBT_FZ1200R33KF2.vth	IGBT_FZ1200R33KF2	3.1231056256176606
IGBT_FZ1600R12KL4C.vth	IGBT_FZ1600R12KL4C	3.1231056256176606

The results of this query are significant because it utilizes the domain rules to return intuitive results.

5.7 Test Query 5

Test 5 is meant to resemble a query that an agent would likely automatically generate. This is the type of query that we envision being used in the future within the VTB software. For this example let us assume that we have three software agents. One inside of Schematic Editor (SE) that has the power to execute some events inside of SE and also sends messages about activities going on inside of SE to agents that request that service. Then we have a design agent that receives SE activity messages from the agent inside of

SE and uses a rule engine to reason on the messages that it receives. And finally we have a search agent that uses our Search Server. For this example the 'design-agent' has determined that we need to add a model to a schematic, so using the data it has it asks the 'search-agent' for a model with certain characteristics which it in turn return to the design-agent who then tells the SE-agent to add that model to the schematic.

DAML Instance:

```
<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:xsd  = "http://www.w3.org/2000/10/XMLschema#"
xmlns              = "http://vtb.engr.sc.edu/daml/VTBModels.daml#">
<VTBModel-Mechanical-Components-Rotating rdf:ID="SEARCH">
<hasPort><Nature/></hasPort>
</VTBModel-Mechanical-Components-Rotating>
</rdf:RDF>
```

Results:

Model	Comment	Distance from Query
GearBox.vtm	GearBox.vtm	0.0
Turbine.vtm	Turbine.vtm	0.0
Diesel_Engine.vtm	Diesel_Engine.vtm	0.0
Shaft.vtm	Shaft.vtm	0.0
Propeller.vtm	Propeller.vtm	0.0
TorqueLoad.vtm	TorqueLoad.vtm	1.7320508075688772
Ship.vtm	Ship.vtm	1.7320508075688772
SurfacePlotTester.vtm	SurfacePlotTester.vtm	1.7320508075688772
heat_exchanger.vtm	heat_exchanger.vtm	2.0
ElectricalGround.vtm	ElectricalGround.vtm	2.0
TemperatureSource.vtm	TemperatureSource.vtm	2.0
ThermalRadiator.vtm	ThermalRadiator.vtm	2.0
Irradiance.vtm	Irradiance.vtm	2.0
IO_Table.vtm	IO_Table.vtm	2.0
NonLinearFunction.vtm	NonLinearFunction.vtm	2.23606797749979
Regulator_Ideal_020409.vtm	Regulator_Ideal_020409.vtm	2.23606797749979
SRC_SumInt.vtm	SRC_SumInt.vtm	2.23606797749979

UDDev.vtm	UDDev.vtm	2.23606797749979
SRC_UnitDelay.vtm	SRC_UnitDelay.vtm	2.23606797749979
SRC_IntegratorEx.vtm	SRC_IntegratorEx.vtm	2.23606797749979

This query demonstrates how an automatically generated search can return results that could be utilized by intelligent software agents.

The five basic queries presented in this chapter demonstrate the benefits of our Semantic Web approach to model searching. These queries use inference and domain rules to return results that are in my opinion are logical and intuitive. Plus there are other benefits such as the ability to search over distributed data, and the enabling of agent interaction through the use of an ontology.

CHAPTER 6

Conclusion

6.1 Analysis

The Semantic Web aims to make the Web more useful by encoding semantics into documents. By encoding semantics into documents we can build intelligent software that can meaningfully manipulate its contents. Once software can do this we can begin to do much more with the Web. In this work our focus was on using basic reasoning to achieve an improved search result. In our work we have used tools developed for the semantic web. In particular we use DAML as our ontology language, Jess as our rule engine, Jena ARP as our DAML parser, and DAMLJessKB as our source of DAML semantic rules.

For our search we used the concept of semantic-distance. We wanted to find out how far away a DAML instance of a query is from an instance of a VTB model described in DAML. To determine the semantic-distance we use an instance based learning method, specifically a K-Nearest neighbor approach. We place both the model and the query in N-dimensional space and compute the distance. To place the models in N-dimensional space we use the Jess rule-engine with rules that define the DAML syntax and rules that

apply domain specific knowledge. These rules give our system basic reasoning skills, which enable a more intelligent search.

The overall performance of the system is adequate. The results from queries and the time to receive results are both reasonable. The test machine used was a 2.0 Gigahertz Pentium 4 with 512 Megabytes of RAM running Windows XP. The average time to calculate the distance of a query is approximately 150 milliseconds. The majority of the time spent in the calculation is the process of asserting the DAML query into Jess. The execution performance of the system is linear to the number of VTB Model Instances. The memory consumption of this application is significant due to Jess, which maximizes speed at the cost of memory. The memory used is linearly related to the number of facts in the knowledge base. For our test cases we have 36,625 facts and 90 megabytes of memory consumption. Based on the performance of our system I do not envision an approach such as this being feasible for a huge dataset. But, I still think it may be feasible for Semantic Web applications where a specific ontology is queried.

Another difficult problem with this work is deciding what is an intelligent search and how it can be tested. In Chapter 5 we showed several different tests that demonstrate how by using knowledge representation along with reasoning tools, a search result is achieved that is closer to what a person may come up with. My background is Computer Science, so the results and examples might not accurately reflect the true potential of this approach. I believe that this type of system could become very effective as knowledge from different backgrounds and specialties are added into the system.

Our approach of placing models in N-dimensional space is effective, but it has several problems. The largest problem is that most of the information we use to classify models doesn't easily translate to a feature vector. Part of this is due to the nature of a VTB model. If we were dealing with a domain where every characteristic could be defined by a real number this would have been a better fit. For example, in the case of our database classification, we were forced to extend our vector and make each classification it's own feature in our vector. While the approach works it's not as fast and intuitive as it could be for other cases. Although we had to force our data into a feature vector structure it does have benefits. First, it is easy to add additional characteristics to the vector and it is easy adjust the importance of features by weighting. Another benefit of this method is that most of the CPU cost is paid at initialization, when all of the models are placed in space. Although the calculations could prove costly if the number of features becomes too large, it was not an issue in this work. In the case of this system the relationship between the search calculation and number of models is linear, but efficient memory indexing can speed this up, at some additional cost of memory [22].

Two of the tools that are used in this work are Jena and Jess. Both of these tools have a lot of potential in future applications. Jess is fast, powerful, and extensible. I believe that Jess can function well as an inference engine for Semantic Web applications. The only part of Jena utilized by this work is the ARP parser, which works very well, especially in conjunction with Jess. The only difficulty using Jess is that it can be difficult to debug Jess programs. I spent a lot of time looking through the facts in the Knowledge Base, trying to determine why Jess did not work as I expected. As Jess matures, and more

efficient methods of debugging are developed, it will become more useful and easier to use.

Using ontologies and inference appears to be an effective method of searching. It seems to more closely resemble the human thought process than traditional keyword matching methods. Supplying the domain knowledge for ontologies, which will make this method of searching more intelligent, is a huge task that needs to be researched. Overall, the technologies used and the methods implemented in this work provide an effective search technique that can be applied to other domains.

6.2 Future Work

There are several benefits to the method we employed for model searching. One of these is that it enables the use of distributed data. By this I mean that there can be many DAML database files located at many different locations. This distributed data allows different setups for setting up the search engine. For example it could be set up as a peer-to-peer network of model servers that talk to each other exchanging model databases, or as a web service with one single master database file, or even a combination of the two. Also, new rules can easily be added to Jess that can enhance search capabilities and changes to the feature vectors are easy to make. Both of these things can greatly impact search results. And searching can be fine-tuned by adjusting the vector weights.

Along with enhancements to the search capabilities there is also the possibility of transforming this into a multi-agent based system. By adding a TCP/IP interface into Schematic Editor we can enable an agent assisted system development environment. This search engine can be easily converted into an agent, which would in turn communicate with Schematic Editor. This would be a first step in adding assisted system design into the VTB software. Specifically, there would be three separate software agents, the SE communication-agent, the design-agent, and the search-agent. The design-agent would register to receive messages from the communication-agent agent which would tell the design agent what is happening within a schematic. The design agent would process those messages by asserting them into Jess and running domain specific rules. The design agent would then use the search-agent to find needed models, and tell the communication-agent what actions to perform in SE. This work presents a practical application of Semantic Web technology in the VTB domain and could enable some exciting things that are enabled by using ontologies, rule-engines, and agents.

REFERENCES

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[2] W3C. Resource Description Framework (RDF) model and syntax specification. http://www.w3.org/ TR/1999/REC-rdf-syntax-19990222/, 1999.

[3] W3C, Naming and Addressing: URIs, URLs, … http://www.w3.org/Addressing/

[4] W3C. Frank Manola, Eric Miller. RDF Primer. http://www.w3.org/TR/2003/WD-rdf-primer-20030123/, January 2003.

[5] W3C. Resource Description Framework Schema Specification (RDF-S). http://www.w3c.org/ TR/2000/CR-rdf-schema-20000327/, 2000.

[6] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, 2003.

[7] DARPA. DAML march 2001 specifications (DAML+OIL). http://www.daml.org/ 2001/03/daml+oil-index, 2001.

[8] Deborah L. McGuiness, Richard Fikes, James Hendler, Lynn Andrea Stein. DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems,* September/October 2002.

[9] DARPA, Adam Pease. Why Use DAML? http://www.daml.org/2002/04/why.html, April 10, 2002.

[10] Brian McBride. Jena: A semantic Web Toolkit. *IEEE Internet Computing*, November/December 2002.

[11] L. Miller, A. Seaborne, and A. Reggiori. Thre Implementations of SquishQL, a Simple RDF Query Language. http://www.hpl.hp.com/techreports/2002/HPL-2002-110.html

[12] Charles L. Forgy. Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. Artificial Intelligence 19(1982), 17-37.

[13] Ernest J. Friedman-Hill. Jess, the Expert System Shell for the Java Platform. http://herzberg.ca.sandia.gov/jess/docs/61/, January 28, 2003.

[14] Joe Kopena, William C. Regli. DAMLJessKB: A Tool for Reasoning with the Semantic Web. http://edge.mcs.drexel.edu/assemblies/software/damljesskb/damljesskb.html

[15] W3C. Web Services Description Language (WSDL) Version 1.2. http://www.w3.org/TR/2003/WD-wsdl12-20030124/, January 24, 2003.

[16] Tyler Jewell, Dave Chappell. UDDI: Universal Description, Discovery, and Integration, Part 1. http://www.onjava.com/pub/a/onjava/excerpt/jws_6/index1.html

[17] uddi.org. UDDI Technical White Paper. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, September 6, 2000.

[18] VTB. Participants. http://vtb.ee.sc.edu/participants/

[19] WebScripter. http://www.isi.edu/webscripter/

[20] edaparts, PartMiner Benefits http://www.edaparts.com/freetradezone.htm

[21] PartMiner http://www.freetradezone.com/

[22] Tom M. Mitchell (1997). Machine Learning. McGraw-Hill Science/Engineering/Math, 230-248

Appendix A

VTBModel Base Classes

VTBModel.h

```
// Base Class for Virtual Test Bed Models
// Name:  CVTBModel.h
//
//  Version      Date      Author      Comment
// ---------    -------   --------    ---------
//  00.01       01/08/01   Lovett      Creation.
#ifdef WIN32
#pragma warning(disable:4190) // C linkage warning
#endif // WIN32
#ifndef VTBMODEL_H
#define VTBMODEL_H
#include <string>
#include <vector>

#ifdef _VTBDLL
#define ENTITY_INTERFACE __declspec(dllexport)
#else
#define ENTITY_INTERFACE __declspec(dllimport)
#endif

//
// The functions used by th models are not exported or imported on Linux.
// The point of the below preprocessing magic is to turn ENTITY_INTERFACE
// into nothing if WIN32 is not defined, which of course it is not on Linux.
//
// wrm (4/17/2002) - Hackarama!
#ifndef WIN32 // If running on Linux
#ifdef ENTITY_INTERFACE // If it is already defined...
#undef ENTITY_INTERFACE // undefine it.
#endif // ENTITY_INTERFACE
#define ENTITY_INTERFACE // Then define it to be nothing.
#endif //WIN32
// wrm (4/17/2002) - End of Hackarama for Linux!

class VtbEntityInformation;

class VtbModel
{
public:
        virtual                                                 ~VtbModel(){};

        virtual VtbEntityInformation*           GetEntityInformation() = 0;
        virtual  bool                                           UpdateParameters() = 0;
```

```
        virtual    bool                                                    Properties() = 0;

        virtual    const std::string                    GetVectorIcon() = 0;

        virtual std::string                                         GenericString(const std::string &strGeneric)
= 0;
        virtual void
        InsertNodeNames(std::vector<std::string>* pvNodeNames,int iNodeType) = 0;
        virtual void
        InsertScopeNames(std::vector<std::string>* pvScopeNames) = 0;
};

extern "C"
{

ENTITY_INTERFACE  std::string          __VTBMODEL__GetSolverName();
ENTITY_INTERFACE  std::string          __VTBMODEL__GetVersionNumber();
ENTITY_INTERFACE  int                                  __VTBMODEL__GetBaseVersion();
ENTITY_INTERFACE  int                                  __VTBMODEL__GetSolverBaseVersion();
ENTITY_INTERFACE  VtbModel*              __VTBMODEL__CreateInstance();
ENTITY_INTERFACE  void                             __VTBMODEL__DestroyInstance(VtbModel*);

}
#endif
```

VTBEntityInformation.h

```
#ifndef VTBENTITYINFORMATION
#define VTBENTITYINFORMATION

#include <string>
#include <vector>
#include "VtbEnumerations.h"

//***************************************************************************
class VtbDataStorage
{
public:
        VtbDataStorage();
        virtual ~VtbDataStorage();
        VtbDataStorage(const VtbDataStorage &data);
        VtbDataStorage& operator=(const VtbDataStorage &data);
        bool SetSize(int iRows, int iColumns);
        VtbMatrixSize GetSize();
        void SetName(const std::string &sName);
        void SetUnits(const std::string &sUnits);
        void SetComment(const std::string &sComment);
        bool SetValue(const std::string &sValue,int iRow, int iColumn);
        void SetType(int iType);
        const std::string &GetValue( int iRow, int iColumn);
        const std::string &GetName();
        const std::string &GetUnits();
```

```
        const std::string &GetComment();
        int GetType();

private:
        bool AllocateMemory(std::string* &saValue);

private:
        std::string* m_saValue;
        VtbMatrixSize m_MatrixSize;

        int m_iType;

        std::string m_sComment;
        std::string m_sName;
        std::string m_sUnits;
        std::string m_sError;
};

class VtbParameter : public VtbDataStorage
{
public:
        VtbParameter();
        virtual ~VtbParameter();
};

class VtbState : public VtbDataStorage
{
public:
        VtbState();
        virtual ~VtbState();
};

class VtbViewable : public VtbDataStorage
{
public:
        VtbViewable();
        virtual ~VtbViewable();
        void SetActive(bool bActive);
        bool GetActive();

private:
        bool m_bActive;
};

//*****************************************************************************
class VtbPortType
{
public:
        VtbPortType();
        virtual ~VtbPortType();
        bool SetSize(int iRows, int iColumns);
        VtbMatrixSize GetSize();
        void SetOpenPort(bool bOpen);
        bool IsOpenPort();
        void SetPortName(const std::string &sPortName);
        const std::string &GetPortName();
```

```
private:
        bool m_bOpenPort;
   VtbMatrixSize m_MatrixSize;
        std::string m_sPortName;
};

class VtbSignal : public VtbPortType
{
public:
        VtbSignal();
        virtual ~VtbSignal();
        void SetSignalType(int iSignalType);
        int  GetSignalType();
        void SetSignalDirection(int iSignalDirection);
        int  GetSignalDirection();

private:
        int m_iSignalType;
        int m_iSignalDirection;
};

class VtbQuantity : public VtbPortType
{
public:
        VtbQuantity();
        virtual ~VtbQuantity();
        void SetQuantityType(int iQuantityType);
        int  GetQuantityType();
        void SetQuantityDirection(int iQuantityDirection);
        int  GetQuantityDirection();

private:
        int m_iQuantityType;
        int m_iQuantityDirection;
};

class VtbNature : public VtbPortType
{
public:
        VtbNature();
        virtual ~VtbNature();
        void SetTerminalNature(const std::string &sTerminalNature);
        const std::string &GetTerminalNature();
        void SetTerminalAcross(const std::string &sTerminalAcross);
        const std::string &GetTerminalAcross();
        void SetTerminalThrough(const std::string &sTerminalThrough);
        const std::string &GetTerminalThrough();
        void SetInternalNode(bool bInternalNode);
        bool IsInternalNode();

private:
        std::string m_sTerminalNature;
        std::string m_sTerminalAcross;
        std::string m_sTerminalThrough;
        bool m_bInternalNode;
```

```
};

//*************************************************************************
class VtbPortLayer
{
public:
	VtbPortLayer();
	virtual ~VtbPortLayer();
	VtbPortLayer(const VtbPortLayer &data);
	VtbPortLayer& operator=(const VtbPortLayer &data);

	int AddNaturePort(bool bInternalNode,int iRows,int iColumns,const std::string &sTerminalName,
		const std::string &sNature,const std::string &sAcross,const std::string &sThrough,
		bool bOpen = false);
	int AddSignalPort(int iRows,int iColumns,const std::string &sSignalName,int iType,
		int iDirection, bool bOpen = false);
	int AddQuantityPort(int iRows,int iColumns,const std::string &sQuantityName,int iType,
		int iDirection, bool bOpen = false);
	int GetNumberNaturePorts();
	int GetNumberSignalInputPorts();
	int GetNumberSignalOutputPorts();
	int GetNumberSignalInOutPorts();
	int GetNumberQuantityInputPorts();
	int GetNumberQuantityOutputPorts();
	int GetNumberQuantityInOutPorts();
	int GetNumberSignalPorts();
	int GetNumberQuantityPorts();
	std::vector<VtbNature*>* GetVectorNaturePorts();
	std::vector<VtbSignal*>* GetVectorSignalInputPorts();
	std::vector<VtbSignal*>* GetVectorSignalOutputPorts();
	std::vector<VtbSignal*>* GetVectorSignalInOutPorts();
	std::vector<VtbQuantity*>* GetVectorQuantityInputPorts();
	std::vector<VtbQuantity*>* GetVectorQuantityOutputPorts();
	std::vector<VtbQuantity*>* GetVectorQuantityInOutPorts();

private:
	std::vector<VtbNature*> m_vNaturePorts;
	std::vector<VtbSignal*> m_vSignalInputPorts;
	std::vector<VtbSignal*> m_vSignalOutputPorts;
	std::vector<VtbSignal*> m_vSignalInOutPorts;
	std::vector<VtbQuantity*> m_vQuantityInputPorts;
	std::vector<VtbQuantity*> m_vQuantityOutputPorts;
	std::vector<VtbQuantity*> m_vQuantityInOutPorts;

};

//*************************************************************************
class VtbEntityInformation
{
public:
	VtbEntityInformation();
	virtual ~VtbEntityInformation();
	VtbEntityInformation(const VtbEntityInformation &data);
	VtbEntityInformation& operator=(const VtbEntityInformation &data);
```

```
        void SetEntityName(const std::string &sEntityName);
        void SetEntityShortName(const std::string &sEntityShortName);
        void SetEntityPrefix(const std::string &sEntityPrefix);
        void SetEntityComment(const std::string &sEntityComment);
        void SetEntityId(int iEntityId);
        void SetCustomPropertiesDialog(bool bPropertiesDialog);
        void SetEntityGround(bool bEntityGround);

        const std::string &GetEntityName();
        const std::string &GetEntityShortName();
        const std::string &GetEntityPrefix();
        const std::string &GetEntityComment();
        int GetEntityId();
        bool HasCustomPropertiesDialog();
        bool IsEntityGround();

        int AddParameter(const std::string &sParameterName,const std::string &sParameterUnits,
                const std::string &sParameterComment,int iType,int iRows,int iColumns);
        int AddState(const std::string &sStateName,const std::string &sStateUnits,
                const std::string &sStateComment,int iType,int iRows,int iColumns);
        int AddViewable(const std::string &sViewableName,const std::string &sViewableUnits,
                const std::string &sViewableComment,int iType,int iRows,int iColumns);
        int AddLayer();

        int GetNumberParameters();
        int GetNumberStates();
        int GetNumberViewables();
        int GetNumberLayers();

        std::vector<VtbParameter*>* GetVectorParameters();
        std::vector<VtbState*>* GetVectorStates();
        std::vector<VtbViewable*>* GetVectorViewables();
        std::vector<VtbPortLayer*>* GetVectorLayers();

private:
        std::string m_sEntityName;
        std::string m_sEntityShortName;
        std::string m_sEntityPrefix;
        std::string m_sEntityComment;
        int m_iEntityId;
        bool m_bPropertiesDialog;
        bool m_bEntityGround;

        std::vector<VtbParameter*> m_vEntityParameters;
        std::vector<VtbState*> m_vEntityStates;
        std::vector<VtbViewable*> m_vEntityViewables;
        std::vector<VtbPortLayer*> m_vEntityLayers;

};

#endif
```

Appendix B

Hierarchical-Device File

IGBT.vth

```
<?xml version="1.0"?>
<se_hierarchical_device1.0>
<device>
<device_name>IGBT_FZ200R65KF1</device_name>
<device_description></device_description>
<device_prefix>IGBT</device_prefix>
<solver_name>VtbRCSignalExSolver|VtbResistiveCompanionSolver</solver_name>
        <group_params>
        </group_params>
        <group_viewables>
        </group_viewables>
</device>
<icon_data>
VI3 0
BEGIN
CIRCLE "Circle0" 13 [-5.2 -8.45 10.4 7.45001 0 0 360 12632256 6008319 0 1 1 1]
LINESTRIP "Line0" 16 [6 0 5 0 5 0 -4 0 -2 0 1 0 1 0 3 0]
LINESTRIP "Line1" 12 [4 -2 3 -2 3 -2 -3 -2 -3 0 3 0]
CIRCLE "Circle1" 13 [5 -2 6 -1 0 0 360 0 0 0 1 1 1]
LINESTRIP "Line6" 8 [2 -11.85 -0.05 -2.05 2.23517e-008 0 3 0]
LINESTRIP "Line7" 8 [2 -11.35 -1.75 -11.35 -1.75 0 3 0]
LINESTRIP "Line5" 10 [3 8.0945 5.37596 8.0945 15.7431 8.0445 15.7431 0 3 0]
TEXT "Text0" 11 [-5 10 3 0 0 0 2 0 "IGBT_Level_I" 0 0]
PORT "Port0" 6 [-12 0 0 0 0 0]
PORT "Port1" 6 [6.39749 13.5747 0 0 0 0]
PORT "Port2" 6 [8.18338 -15.9389 0 0 0 0]
PORT "Port3" 6 [23.0012 -1 0 0 0 0]
LINESTRIP "Line2" 8 [2 0.0714567 1.17904 8.1762 5.45611 0 3 0]
LINESTRIP "Line3" 10 [3 -0.0611257 -1.19195 8.00746 -6.14313 8.00746 -15.3731 0 3 0]
LINESTRIP "Line4" 10 [3 5.56243 -1.5587 10.3914 -0.0305628 23.2889 0.0305628 0 3 0]
TEXT "Text1" 11 [-12 3 2.5 0 0 0 0 0 "G" 0 0]
TEXT "Text2" 11 [11 16 2.5 0 0 0 0 0 "C" 0 0]
TEXT "Text3" 11 [10 -15 2.5 0 0 0 0 0 "E" 0 0]
TEXT "Text4" 11 [24 3 2.5 0 0 0 0 0 "T" 0 0]
POLY "Polygon0" 22 [5 5.48044 -3.51942 12632256 5.48044 -3.51942 12632256 4.24839 -5.38545
12632256 4.24839 -5.38545 12632256 8.34646 -6.48352 12632256 0 0 1 1 1 0]
END
</icon_data>
<se_documentVer1.2>
        <solver>
                <library_name>RCSignalExSolver.vtx</library_name>
                <version>01.01.00 - Release Build</version>
                <solver_info>
```

```
		Version 1
		SimulationParameters
		{
			Timestep 0.0001
			Tolerance 7e-006
			EndTime 1
			NewtonsIterations 10
			NewtonsMethod 0
			InversionMethod 0
			DisplayMode 0
			DisplayStep 0.0001
			DisplayTolerance 0.1
		}
		SimulationAbilities
		{
			TimeStep 1
			NullStep 1
			RunTimeChanges 1
			StateRestart 1
		}
		CustomObject
		{
		}
		</solver_info>
	</solver>
	<groups>
	</groups>
	<entities>
		<entity>
			<library_name>IGBT_Level_I.vhd.vtm</library_name>
			<version>01.01.00 - Release Build</version>
			<id>2</id>
			<group_id>-1</group_id>
			<prefix>IGBTS0</prefix>
			<comment></comment>
			<parameters>
				<parameter>
				<comment>Gate Resistance</comment>
					<matrix_node>
						<row>0</row>
						<column>0</column>
						<value>13.0</value>
					</matrix_node>
				</parameter>
				<parameter>
				<comment>Resistance 1 of thermal circuit</comment>
					<matrix_node>
						<row>0</row>
						<column>0</column>
						<value>0.0201953</value>
					</matrix_node>
				</parameter>
				<parameter>
				<comment>Resistance 2 of thermal circuit</comment>
					<matrix_node>
						<row>0</row>
```

```
                    <column>0</column>
                    <value>0.0063022</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Resistance 3 of thermal circuit</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>0.0048262</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Resistance 4 of thermal circuit</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>0.0016763</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Control voltage for on state</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>10.0</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Control voltage for off state</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>5.0</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>On Resistance</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>0.00008</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Off Resistance</comment>
                <matrix_node>
                    <row>0</row>
                    <column>0</column>
                    <value>1000000</value>
                </matrix_node>
            </parameter>
            <parameter>
            <comment>Gate Capacitance</comment>
                <matrix_node>
                    <row>0</row>
```

```
                                        <column>0</column>
                                        <value>0.000000652</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Capacitance 1 of thermal circuit</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>1.689</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Capacitance 2 of thermal circuit</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>17.334</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Capacitance 3 of thermal circuit</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>101.055</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Capacitance 4 of thermal circuit</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>295.406</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Linearized on-state resistance at 300K</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>0.0090625</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>On-state voltage at zero current at 300K</comment>
                                <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>2.25</value>
                                </matrix_node>
                        </parameter>
                        <parameter>
                        <comment>Temperature dependance exponent for on-state
resistance</comment>
                                <matrix_node>
```

```
                                        <row>0</row>
                                        <column>0</column>
                                        <value>0.99417</value>
                                    </matrix_node>
                                </parameter>
                                <parameter>
                                <comment>Temperature dependance exponent for on-state voltage at
zero current</comment>
                                    <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>0.30249</value>
                                    </matrix_node>
                                </parameter>
                                <parameter>
                                <comment>Maximum collector-emitter voltage</comment>
                                    <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>6500</value>
                                    </matrix_node>
                                </parameter>
                                <parameter>
                                <comment>Maximum collector current</comment>
                                    <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>200</value>
                                    </matrix_node>
                                </parameter>
                                <parameter>
                                <comment>Maximum junction temperature</comment>
                                    <matrix_node>
                                        <row>0</row>
                                        <column>0</column>
                                        <value>423</value>
                                    </matrix_node>
                                </parameter>
                            </parameters>
                            <entity_node_info>
                            </entity_node_info>
                            <entity_point>
                            <point>
                                <x_value>0</x_value>
                                <y_value>0</y_value>
                            </point>
                            </entity_point>
                            <prefix_point>
                            <point>
                                <x_value>24</x_value>
                                <y_value>16</y_value>
                            </point>
                            </prefix_point>
                            <variable_point>
                            <point>
                                <x_value>0</x_value>
```

```
                                        <y_value>0</y_value>
                                </point>
                                </variable_point>
                                <entity_orientation>0</entity_orientation>
                        <isground>0</isground>
                        </entity>
                        <entity>
                                <library_name>PortNature</library_name>
                                <version>1.0</version>
                                <id>3</id>
                                <group_id>-1</group_id>
                                <prefix>Pt0</prefix>
                                <comment></comment>
                                <parameters>
                                        <parameter>
                                        <comment>index of the port - must correlate with the vector icon
port</comment>
                                                <matrix_node>
                                                        <row>0</row>
                                                        <column>0</column>
                                                        <value>0</value>
                                                </matrix_node>
                                        </parameter>
                                </parameters>
                                <entity_node_info>
                                </entity_node_info>
                                <entity_point>
                                <point>
                                        <x_value>-44</x_value>
                                        <y_value>0</y_value>
                                </point>
                                </entity_point>
                                <prefix_point>
                                <point>
                                        <x_value>-36</x_value>
                                        <y_value>3</y_value>
                                </point>
                                </prefix_point>
                                <variable_point>
                                <point>
                                        <x_value>0</x_value>
                                        <y_value>0</y_value>
                                </point>
                                </variable_point>
                                <entity_orientation>0</entity_orientation>
                        <isground>0</isground>
                        </entity>
                        <entity>
                                <library_name>PortNature</library_name>
                                <version>1.0</version>
                                <id>4</id>
                                <group_id>-1</group_id>
                                <prefix>Pt1</prefix>
                                <comment></comment>
                                <parameters>
                                        <parameter>
```

```
                                        <comment>index of the port - must correlate with the vector icon
port</comment>
                                                <matrix_node>
                                                        <row>0</row>
                                                        <column>0</column>
                                                        <value>1</value>
                                                </matrix_node>
                                        </parameter>
                                </parameters>
                                <entity_node_info>
                                </entity_node_info>
                                <entity_point>
                                <point>
                                        <x_value>4</x_value>
                                        <y_value>36</y_value>
                                </point>
                                </entity_point>
                                <prefix_point>
                                <point>
                                        <x_value>12</x_value>
                                        <y_value>39</y_value>
                                </point>
                                </prefix_point>
                                <variable_point>
                                <point>
                                        <x_value>4</x_value>
                                        <y_value>36</y_value>
                                </point>
                                </variable_point>
                                <entity_orientation>0</entity_orientation>
                        <isground>0</isground>
                        </entity>
                        <entity>
                                <library_name>PortNature</library_name>
                                <version>1.0</version>
                                <id>5</id>
                                <group_id>-1</group_id>
                                <prefix>Pt2</prefix>
                                <comment></comment>
                                <parameters>
                                        <parameter>
                                        <comment>index of the port - must correlate with the vector icon
port</comment>
                                                <matrix_node>
                                                        <row>0</row>
                                                        <column>0</column>
                                                        <value>2</value>
                                                </matrix_node>
                                        </parameter>
                                </parameters>
                                <entity_node_info>
                                </entity_node_info>
                                <entity_point>
                                <point>
                                        <x_value>12</x_value>
                                        <y_value>-32</y_value>
```

```
                        </point>
                        </entity_point>
                        <prefix_point>
                        <point>
                                <x_value>20</x_value>
                                <y_value>-29</y_value>
                        </point>
                        </prefix_point>
                        <variable_point>
                        <point>
                                <x_value>0</x_value>
                                <y_value>0</y_value>
                        </point>
                        </variable_point>
                        <entity_orientation>0</entity_orientation>
                <isground>0</isground>
                </entity>
                <entity>
                        <library_name>PortNature</library_name>
                        <version>1.0</version>
                        <id>6</id>
                        <group_id>-1</group_id>
                        <prefix>Pt3</prefix>
                        <comment></comment>
                        <parameters>
                                <parameter>
                                <comment>index of the port - must correlate with the vector icon
port</comment>
                                        <matrix_node>
                                                <row>0</row>
                                                <column>0</column>
                                                <value>3</value>
                                        </matrix_node>
                                </parameter>
                        </parameters>
                        <entity_node_info>
                        </entity_node_info>
                        <entity_point>
                        <point>
                                <x_value>40</x_value>
                                <y_value>0</y_value>
                        </point>
                        </entity_point>
                        <prefix_point>
                        <point>
                                <x_value>48</x_value>
                                <y_value>3</y_value>
                        </point>
                        </prefix_point>
                        <variable_point>
                        <point>
                                <x_value>0</x_value>
                                <y_value>0</y_value>
                        </point>
                        </variable_point>
                        <entity_orientation>0</entity_orientation>
```

```
            <isground>0</isground>
            </entity>
        </entities>
        <scopes>
        </scopes>
        <twocomms>
        </twocomms>
        <lines>
            <line_group>
                <line_group_node_num>0</line_group_node_num>
                <line_group_node_name>Nat_0</line_group_node_name>
                <line_group_node_type>0</line_group_node_type>
            <line>
                <points>
                    <point>
                        <x_value>8</x_value>
                        <y_value>-16</y_value>
                    </point>
                    <point>
                        <x_value>12</x_value>
                        <y_value>-32</y_value>
                    </point>
                    <point>
                        <x_value>12</x_value>
                        <y_value>-32</y_value>
                    </point>
                </points>
                <type>0</type>
                <size>1</size>
                <autocolor>1</autocolor>
                <color>
                    <red>0</red>
                    <green>0</green>
                    <blue>0</blue>
                </color>
                <node_name>Nat_0</node_name>
                <node_num>0</node_num>
            </line>
            </line_group>
            <line_group>
                <line_group_node_num>1</line_group_node_num>
                <line_group_node_name>Nat_1</line_group_node_name>
                <line_group_node_type>0</line_group_node_type>
            <line>
                <points>
                    <point>
                        <x_value>24</x_value>
                        <y_value>0</y_value>
                    </point>
                    <point>
                        <x_value>40</x_value>
                        <y_value>0</y_value>
                    </point>
                    <point>
                        <x_value>40</x_value>
                        <y_value>0</y_value>
```

```
                    </point>
                </points>
                <type>0</type>
                <size>1</size>
                <autocolor>1</autocolor>
                <color>
                    <red>0</red>
                    <green>0</green>
                    <blue>0</blue>
                </color>
                <node_name>Nat_1</node_name>
                <node_num>1</node_num>
            </line>
            </line_group>
            <line_group>
                <line_group_node_num>2</line_group_node_num>
                <line_group_node_name>Nat_2</line_group_node_name>
                <line_group_node_type>0</line_group_node_type>
            <line>
                <points>
                    <point>
                        <x_value>-12</x_value>
                        <y_value>0</y_value>
                    </point>
                    <point>
                        <x_value>-44</x_value>
                        <y_value>0</y_value>
                    </point>
                    <point>
                        <x_value>-44</x_value>
                        <y_value>0</y_value>
                    </point>
                </points>
                <type>0</type>
                <size>1</size>
                <autocolor>1</autocolor>
                <color>
                    <red>0</red>
                    <green>0</green>
                    <blue>0</blue>
                </color>
                <node_name>Nat_2</node_name>
                <node_num>2</node_num>
            </line>
            </line_group>
            <line_group>
                <line_group_node_num>3</line_group_node_num>
                <line_group_node_name>Nat_3</line_group_node_name>
                <line_group_node_type>0</line_group_node_type>
            <line>
                <points>
                    <point>
                        <x_value>4</x_value>
                        <y_value>36</y_value>
                    </point>
                    <point>
```

```
                                        <x_value>8</x_value>
                                        <y_value>16</y_value>
                                </point>
                                <point>
                                        <x_value>8</x_value>
                                        <y_value>16</y_value>
                                </point>
                        </points>
                        <type>0</type>
                        <size>1</size>
                        <autocolor>1</autocolor>
                        <color>
                                <red>0</red>
                                <green>0</green>
                                <blue>0</blue>
                        </color>
                        <node_name>Nat_3</node_name>
                        <node_num>3</node_num>
                </line>
                </line_group>
        </lines>
        <view>
                <point>
                        <x_value>-155.521</x_value>
                        <y_value>-100</y_value>
                </point>
                <point>
                        <x_value>155.521</x_value>
                        <y_value>100</y_value>
                </point>
                <grid_type>4</grid_type>
                <line_type>3</line_type>
        </view>
</se_documentVer1.2>
</se_hierarchical_device1.0>
```

Appendix C

VTB Model Ontology

VTBModel.daml

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<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:xsd  = "http://www.w3.org/2000/10/XMLschema#"
  xmlns      = "http://vtb.engr.sc.edu/daml/VTBModels.daml#"
>

<!-- VTBModel Ontology -->

<daml:Ontology rdf:about="">
        <daml:versionInfo>VTBModel_v0.1</daml:versionInfo>
        <rdfs:comment>A first pass VTBModel Ontology</rdfs:comment>
</daml:Ontology>

<!-- Begin class definitions -->

<!-- Base class for all vtb models -->

<daml:Class rdf:ID="VTBObject"/>

<daml:Class rdf:ID="VTBModel">
        <daml:subClassOf rdf:resource="#VTBObject"/>
        <rdfs:label>VTBModel</rdfs:label>
        <rdfs:comment>This is the base class for all vtb models</rdfs:comment>
</daml:Class>

<!-- Seperate Disciplines -->
<daml:Class rdf:ID="VTBModel-Controls">
        <rdfs:label>Controls</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical">
        <rdfs:label>Electrical</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electromechanical">
        <rdfs:label>Electromechanical</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
```

```
</daml:Class>

<daml:Class rdf:ID="VTBModel-Fluids">
        <rdfs:label>Fluids</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-General">
        <rdfs:label>General</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices">
        <rdfs:label>Hierarchical Devices</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Illumination">
        <rdfs:label>Illumination</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Mechanical">
        <rdfs:label>Mechanical</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Other">
        <rdfs:label>Other</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Thermal">
        <rdfs:label>Thermal</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel"/>
</daml:Class>

<!-- VTBModel-Controls Sub-Catergories -->

<daml:Class rdf:ID="VTBModel-Controls-Discrete_Time">
        <rdfs:label>Discrete Time</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Others">
        <rdfs:label>Others</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-ModulationTechniques">
        <rdfs:label> Modulation Techniques</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Displays">
        <rdfs:label>Displays</rdfs:label>
```

```
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Hybrid">
        <rdfs:label>Hybrid</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Linear">
        <rdfs:label>Linear</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Logic">
        <rdfs:label>Logic</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Nonlinear">
        <rdfs:label>Nonlinear</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Sources">
        <rdfs:label>Sources</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Controls-Transformations">
        <rdfs:label>Transformations</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Controls"/>
</daml:Class>

<!-- VTBModel-Electrical Sub-Catergories -->

<daml:Class rdf:ID="VTBModel-Electrical-Cables">
        <rdfs:label>Cables</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Disturbances">
        <rdfs:label>Disturbances</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Loads">
        <rdfs:label>Loads</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Meters">
        <rdfs:label>Meters</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Passives">
```

```
        <rdfs:label>Passives</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Power_Converters">
        <rdfs:label>Power Converters</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Power_Semiconductors">
        <rdfs:label>Power Semiconductors</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Semiconductors">
        <rdfs:label>Semiconductors</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Sensors">
        <rdfs:label>Sensor</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Sources">
        <rdfs:label>Sources</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Switches">
        <rdfs:label>Switches</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Transformers">
        <rdfs:label>Transformers</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical"/>
</daml:Class>

<!-- VTBModel-Electromechanical Sub-Catergories -->

<daml:Class rdf:ID="VTBModel-Electromechanical-Generators">
        <rdfs:label>Generators</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electromechanical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electromechanical-Motors">
        <rdfs:label>Motors</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electromechanical"/>
</daml:Class>

<!-- Fluid Sub-Catergories -->

<daml:Class rdf:ID="VTBModel-Fluids-Fluid">
        <rdfs:label>Fluid</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Fluids"/>
```

```
</daml:Class>

<daml:Class rdf:ID="VTBModel-Fluids-Gaseous">
        <rdfs:label>Gaseous</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Fluids"/>
</daml:Class>

<!-- General subs -->

<daml:Class rdf:ID="VTBModel-General-Native">
        <rdfs:label>Native</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-General"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-General-Wrapper">
        <rdfs:label>Wrapper</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-General"/>
</daml:Class>

<!-- VTBModel-Hierarchical_Devices subs -->

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-Control">
        <rdfs:label>Control</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-IGBT">
        <rdfs:label>IGBT</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-IGCT">
        <rdfs:label>IGCT</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-PowerConverter">
        <rdfs:label>PowerConverter</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-Sensors">
        <rdfs:label>Sensors</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices"/>
</daml:Class>


<!-- Mechanical Sub-Catergories -->

<daml:Class rdf:ID="VTBModel-Mechanical-Components">
        <rdfs:label>Components</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Mechanical-Loads">
        <rdfs:label>Loads</rdfs:label>
```

```
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical"/>
</daml:Class>

<!-- sources subs -->

<daml:Class rdf:ID="VTBModel-Electrical-Sources-Controlled_Sources">
        <rdfs:label>Controlled Sources</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical-Sources"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Sources-Generic_Sources">
        <rdfs:label>Generic Sources</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical-Sources"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Electrical-Sources-Sources">
        <rdfs:label>Sources</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Electrical-Sources"/>
</daml:Class>

<!-- Components subs -->

<daml:Class rdf:ID="VTBModel-Mechanical-Components-Linear">
        <rdfs:label>Linear</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical-Components"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-Mechanical-Components-Rotating">
        <rdfs:label>Rotating</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Mechanical-Components"/>
</daml:Class>

<!-- VTBModel-General-Wrapper subs-->

<daml:Class rdf:ID="VTBModel-General-Wrapper-Matlab">
        <rdfs:label>Matlab</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-General-Wrapper"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-General-Wrapper-ACSL">
        <rdfs:label>Matlab</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-General-Wrapper"/>
</daml:Class>

<daml:Class rdf:ID="VTBModel-General-Wrapper-LabView">
        <rdfs:label>Matlab</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-General-Wrapper"/>
</daml:Class>

<!-- VTBModel-Hierarchical_Devices-Control -->

<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-Control-ModulationTechniques">
        <rdfs:label>ModulationTechniques</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices-Control"/>
</daml:Class>
```

```
<daml:Class rdf:ID="VTBModel-Hierarchical_Devices-Control-Others">
        <rdfs:label>Others</rdfs:label>
        <daml:subClassOf rdf:resource="#VTBModel-Hierarchical_Devices-Control"/>
</daml:Class>

<!-- Properties and stuff -->
<!-- general classes and property statements -->

<daml:ObjectProperty rdf:ID="hasSize">
        <rdfs:range rdf:resource="#Size"/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Size">
</daml:Class>

<daml:DatatypeProperty rdf:ID="ModelName"/>

<daml:DatatypeProperty rdf:ID="Comment"/>

<daml:DatatypeProperty rdf:ID="Rows">
        <rdfs:domain rdf:resource="#Size"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Columns">
        <rdfs:domain rdf:resource="#Size"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="IODirection">
        <daml:oneOf rdf:parseType="daml:collection">
                <IODirection rdf:ID="INPUT"/>
                <IODirection rdf:ID="OUTPUT"/>
                <IODirection rdf:ID="INPUT_OUTPUT"/>
                <IODirection rdf:ID="UNDEFINED_DIRECTION"/>
        </daml:oneOf>
</daml:Class>

<daml:Class rdf:ID="ValueType">
        <daml:oneOf rdf:parseType="daml:collection">
                <ValueType rdf:ID="INTEGER"/>
                <ValueType rdf:ID="REAL"/>
                <ValueType rdf:ID="BIT"/>
                <ValueType rdf:ID="BOOLEAN"/>
                <ValueType rdf:ID="CHARACTER"/>
                <ValueType rdf:ID="STRING"/>
                <ValueType rdf:ID="TIME"/>
                <ValueType rdf:ID="BIT_VECTOR"/>
                <ValueType rdf:ID="UNDEFINED"/>
        </daml:oneOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="hasType">
        <rdfs:range rdf:resource="#ValueType"/>
</daml:ObjectProperty>
```

```
<daml:ObjectProperty rdf:ID="hasKeywords"/>

<!-- Model Parameters Information -->

<daml:Class rdf:ID="Parameter">
</daml:Class>

<daml:DatatypeProperty rdf:ID="ParameterProperty"/>

<daml:DatatypeProperty rdf:ID="ParameterName">
        <rdfs:subPropertyOf rdf:resource="#ParameterProperty"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="ParameterValue">
        <rdfs:subPropertyOf rdf:resource="#ParameterProperty"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="ParameterUnits">
        <rdfs:subPropertyOf rdf:resource="#ParameterProperty"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="ParameterComment">
        <rdfs:subPropertyOf rdf:resource="#ParameterProperty"/>
</daml:DatatypeProperty>


<daml:Class rdf:about="#Parameter">
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
        <daml:onProperty rdf:resource="#ParameterName"/>
        </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#ParameterValue"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#ParameterUnits"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
        <daml:onProperty rdf:resource="#ParameterComment"/>
        </daml:Restriction>
        </daml:subClassOf>
        <!-- <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasSize"/>
        </daml:Restriction>
        </daml:subClassOf>        -->
        <daml:subClassOf>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasType"/>
                </daml:Restriction>
```

```
        </daml:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="hasParameter">
        <rdfs:domain rdf:resource="#VTBModel"/>
</daml:ObjectProperty>

<daml:Class rdf:about="#VTBModel">
        <daml:subClassOf>
                <daml:Restriction>
                        <daml:onProperty rdf:resource="#hasParameter"/>
                </daml:Restriction>
        </daml:subClassOf>
</daml:Class>

<!-- Model port information -->

<daml:Class rdf:ID="Port"/>

<daml:DatatypeProperty rdf:ID="PortName">
        <rdfs:domain rdf:resource="#Port"/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Nature">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Signal"/>
        <daml:disjointWith rdf:resource="#Quantity"/>
</daml:Class>

<daml:Class rdf:ID="Signal">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Nature"/>
        <daml:disjointWith rdf:resource="#Quantity"/>
        <daml:subClassOf>
                <daml:Restriction daml:maxcardinality="1">
                        <daml:onProperty rdf:resource="#hasDirection"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:maxcardinality="1">
                        <daml:onProperty rdf:resource="#hasType"/>
                </daml:Restriction>
        </daml:subClassOf>
</daml:Class>

<daml:Class rdf:ID="Quantity">
        <daml:subClassOf rdf:resource="#Port"/>
        <daml:disjointWith rdf:resource="#Signal"/>
        <daml:disjointWith rdf:resource="#Nature"/>
        <daml:subClassOf>
                <daml:Restriction daml:maxcardinality="1">
                        <daml:onProperty rdf:resource="#hasDirection"/>
                </daml:Restriction>
        </daml:subClassOf>
        <daml:subClassOf>
                <daml:Restriction daml:maxcardinality="1">
```

```
                        <daml:onProperty rdf:resource="#hasType"/>
                    </daml:Restriction>
            </daml:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="hasDirection">
            <rdfs:domain rdf:resource="#Port"/>
            <rdfs:range rdf:resource="#IODirection"/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID="TeminalNature">
            <rdfs:domain rdf:resource="#Nature"/>
</daml:DatatypeProperty >

<daml:DatatypeProperty rdf:ID="TeminalAcross">
            <rdfs:domain rdf:resource="#Nature"/>
</daml:DatatypeProperty >

<daml:DatatypeProperty rdf:ID="TeminalThrough">
            <rdfs:domain rdf:resource="#Nature"/>
</daml:DatatypeProperty >

<daml:ObjectProperty rdf:ID="hasPort">
            <rdfs:domain rdf:resource="#VTBModel"/>
</daml:ObjectProperty>

<daml:Class rdf:about="#VTBModel">
            <daml:subClassOf>
                        <daml:Restriction>
                                    <daml:onProperty rdf:resource="#hasPort"/>
                        </daml:Restriction>
            </daml:subClassOf>
</daml:Class>

<!-- sub models -->

<daml:ObjectProperty rdf:ID="hasModel">
            <rdfs:domain rdf:resource="#VTBModel"/>
</daml:ObjectProperty>


<!-- Resource link location -->

<daml:DatatypeProperty rdf:ID="Resource">
            <rdfs:comment>The actual file name of the model.</rdfs:comment>
            <rdfs:domain rdf:resource="#VTBModel"/>
</daml:DatatypeProperty>

<daml:Class rdf:about="#VTBModel">
            <daml:subClassOf>
                        <daml:Restriction daml:cardinality="1">
                                    <daml:onProperty rdf:resource="#Resource"/>
                        </daml:Restriction>
            </daml:subClassOf>
            <daml:subClassOf>
                        <daml:Restriction daml:cardinality="1">
```

```
                        <daml:onProperty rdf:resource="#hasKeywords"/>
                    </daml:Restriction>
                </daml:subClassOf>
</daml:Class>
</rdf:RDF>
```

Appendix D

DAML rules

damljesskb-facts.jess

```
;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
;;----- THE FACTS -----------------------------------------------------
;;---------------------------------------------------------------------
;;---------------------------------------------------------------------

(deffacts damljess-basic-facts
  "The basic facts about RDF, RDFS, DAML."

  ;;-------------------------------------------
  ;;----- RDF/RDFS DEFINITIONS -------

  (PropertyValue rdf:type rdf:Resource rdfs:Class)
  (PropertyValue rdf:type rdf:type rdf:Property)
  (PropertyValue rdf:type rdf:Property rdf:Resource)
  (PropertyValue rdf:type rdf:Property rdfs:Class)
  (PropertyValue rdf:type rdfs:Class rdfs:Class)
  (PropertyValue rdfs:subClassOf rdfs:Class rdf:Resource)
  (PropertyValue rdfs:subClassOf rdf:Property rdf:Resource)
  (PropertyValue rdf:type rdfs:subClassOf rdf:Property)
  (PropertyValue rdf:type rdf:value rdf:Property)

  (PropertyValue rdfs:domain rdf:type rdf:Resource)
  (PropertyValue rdfs:range rdf:type rdfs:Class)
  (PropertyValue rdfs:domain rdfs:subClassOf rdfs:Class)
  (PropertyValue rdfs:range rdfs:subClassOf rdfs:Class)

  (PropertyValue rdf:type rdfs:subPropertyOf rdf:Property)
  (PropertyValue rdf:type rdfs:domain rdf:Property)
  (PropertyValue rdf:type rdfs:range rdf:Property)
  (PropertyValue rdfs:domain rdfs:subPropertyOf rdf:Property)
  (PropertyValue rdfs:range rdfs:subPropertyOf rdf:Property)
  (PropertyValue rdfs:domain rdfs:domain rdf:Property)
  (PropertyValue rdfs:range rdfs:domain rdfs:Class)
  (PropertyValue rdfs:domain rdfs:range rdf:Property)
  (PropertyValue rdfs:range rdfs:range rdfs:Class)

  (PropertyValue rdf:type rdfs:seeAlso rdf:Property)
  (PropertyValue rdf:type rdfs:isDefinedBy rdf:Property)
  (PropertyValue rdf:type rdfs:comment rdf:Property)
  (PropertyValue rdf:type rdfs:label rdf:Property)
  (PropertyValue rdfs:subPropertyOf rdfs:isDefinedBy rdfs:seeAlso)
  (PropertyValue rdfs:range rdfs:comment rdf:Literal)
```

```
(PropertyValue rdfs:range rdfs:label rdf:Literal)

(PropertyValue rdf:type rdf:Literal rdfs:Class)

;;-------------------------------------------
;;----- DAML DEFINITIONS -------

(PropertyValue rdfs:subClassOf daml:Class rdfs:Class)
(PropertyValue rdf:type daml:Class rdfs:Class)
(PropertyValue rdf:type daml:Thing daml:Class)

(PropertyValue rdf:type daml:Datatype rdfs:Class)
(PropertyValue rdfs:subClassOf daml:Datatype rdfs:Class)

(PropertyValue daml:complimentOf daml:Thing daml:Nothing)

(PropertyValue rdf:type daml:Restriction rdfs:Class)
(PropertyValue rdfs:subClassOf daml:Restriction daml:Class)

(PropertyValue rdf:type daml:ObjectProperty rdfs:Class)
(PropertyValue rdfs:subClassOf daml:ObjectProperty rdf:Property)

(PropertyValue rdf:type daml:DatatypeProperty rdfs:Class)
(PropertyValue rdfs:subClassOf daml:DatatypeProperty rdf:Property)

(PropertyValue rdf:type daml:TransitiveProperty rdfs:Class)
(PropertyValue rdfs:subClassOf daml:TransitiveProperty daml:ObjectProperty)

(PropertyValue rdf:type daml:Ontology rdfs:Class)

(PropertyValue rdf:type daml:disjointWith daml:ObjectProperty)
(PropertyValue rdfs:domain daml:disjointWith daml:Class)
(PropertyValue rdfs:range daml:disjointWith rdfs:Class)

(PropertyValue rdf:type daml:subClassOf rdf:Property)
(PropertyValue rdfs:subPropertyOf daml:subClassOf rdfs:subClassOf)
(PropertyValue rdfs:subPropertyOf rdfs:subClassOf daml:subClassOf)

(PropertyValue rdf:type daml:intersectionOf rdf:Property)
(PropertyValue rdfs:domain daml:intersectionOf daml:Class)
(PropertyValue rdfs:range daml:intersectionOf daml:List)

(PropertyValue rdf:type daml:unionOf rdf:Property)
(PropertyValue rdfs:domain daml:unionOf daml:Class)
(PropertyValue rdfs:range daml:unionOf daml:List)

(PropertyValue rdf:type daml:subClassOf daml:TransitiveProperty)
;;Added by A. Finkbeiner
; (PropertyValue daml:samePropertyAs daml:subClassOf rdfs:subClassOf)

(PropertyValue rdf:type daml:subClassOf rdfs:subClassOf)
(PropertyValue rdfs:subClassOf daml:subClassOf rdfs:subClassOf)
;; Some of this is gettin' kinda dubious...

(PropertyValue rdf:type daml:List daml:Class)
(PropertyValue rdf:type daml:first daml:ObjectProperty)
```

```
  (PropertyValue rdf:type daml:rest daml:ObjectProperty)
  (PropertyValue rdfs:domain daml:first daml:List)
  (PropertyValue rdfs:domain daml:rest daml:List)
  (PropertyValue rdfs:range daml:rest daml:List)

  (PropertyValue rdf:type daml:nil daml:List)

  ;; end the basic deffacts
)
```

damljesskb-base-constraints.jess

```
;;-----------------------------------------------------------------------
;; damljesskb-basic-constraints.jess.src --- Rules implementing some of
;; the basic constraints present in DAML/RDFS. Examples include
;; rdfs:domain, rdfs:range and some of the daml:Restriction objects.
;; Dated 2002/07/17 --- Joe Kopena (joe@plan.mcs.drexel.edu)
;;-----------------------------------------------------------------------


;;-----
;; This file is generated from sources written in Perlcam.
;; Perlcam 0.1---Joe Kopena (joe@plan.mcs.drexel.edu)
;; More information is available at http://plan.mcs.drexel.edu/~joe/
;;-----


;;-----------------------------------------------------------------------
;;-----------------------------------------------------------------------
(defrule rdfs-domain
  "An object value for an property with a domain restriction must be
   a member of that domain."

  (declare (salience -100))

  (PropertyValue rdfs:domain ?p ?c)
  (PropertyValue ?p ?i ?o)
  (not (PropertyValue rdf:type ?i ?c))
=>
  (assert (PropertyValue rdf:type ?i ?c))
 ; (gentle-warning "Set object '" ?i "' type to '" ?c "' due to a domain restriction on '" ?p "'")
)

;;-----------------------------------------------------------------------
;;-----------------------------------------------------------------------
(defrule daml-disjointWith
  "Two classes can be disjoint and should not share any members."

  (PropertyValue daml:disjointWith ?x ?y)
  (PropertyValue rdf:type ?i ?x)
  (PropertyValue rdf:type ?i ?y)
=>
  (error "Object '" ?i "' is of type '" ?x "' and '" ?y "' but they are disjoint")
)
```

damljesskb-base-semantics.jess

```
;;-------------------------------------------------------------------
;; damljesskb-base-semantics.jess.src --- Rules implementing basic DAML
;;   semantics. Other files implement more complex semantics, some
;;   semantics are represented/implementing in DAMLJessKB java code
;; Dated 2002/05/03 --- Joe Kopena (joe@plan.mcs.drexel.edu)
;;   (perlcamelized sometime later)
;;-------------------------------------------------------------------


;;-----
;; This file is generated from sources written in Perlcam.
;; Perlcam 0.1---Joe Kopena (joe@plan.mcs.drexel.edu)
;; More information is available at http://plan.mcs.drexel.edu/~joe/
;;-----


;;-------------------------------------------------------------------

;;-------------------------------------------------------------------
;(defrule subclass-idempotent
;  (PropertyValue rdf:type ?class rdfs:Class)
;=>
;  (assert (PropertyValue rdfs:subClassOf ?class ?class))
;)

;;-------------------------------------------------------------------
(defrule subclass-instances
  "An instance of a subclass is an instance of the parent class. This
   enforces and makes meaningful the daml:subClassOf relationship."

  (PropertyValue daml:subClassOf ?child ?parent)
  (PropertyValue rdf:type ?instance ?child)
=>
 (assert
  (PropertyValue rdf:type ?instance ?parent)
 )
)

;;-------------------------------------------------------------------
;;-------------------------------------------------------------------
(defrule subproperty-instances
  "An instance of a subproperty is an instance of the parent property. This
   enforces and makes meaningful the rdfs:subPropertyOf relationship."

  ;; Note the use of the rdfs predicate instead of the daml version
  (PropertyValue rdfs:subPropertyOf ?child ?parent)
  (PropertyValue ?child ?s ?o)
=>
 (assert
   (PropertyValue ?parent ?s ?o)
 )
)

;;-------------------------------------------------------------------
;;-------------------------------------------------------------------
```

```
;; removed by me
;(defrule subclass-rdf-resource
;  "Every root class is a subclass of rdf:resource."
;
;  (PropertyValue rdf:type ?class rdfs:Class)
;=>
;  (assert
;    (PropertyValue rdfs:subClassOf ?class rdf:Resource)
;  )
;)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
;; removed by me
;(defrule subclass-daml-thing
;  "Every class is a subclass of daml:Thing."
;
;  (PropertyValue rdf:type ?class&~daml:Thing daml:Class)
;  ; (not (PropertyValue daml:subClassOf ?class ?root))
;=>
;  (assert
;    (PropertyValue rdfs:subClassOf ?class daml:Thing)
;  )
;)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(defrule transitive-property
  "Defines transitiveness for a property."

  (PropertyValue rdf:type ?prop daml:TransitiveProperty)
  (PropertyValue ?prop ?x ?y)
  (PropertyValue ?prop ?y ?z)
=>
  (assert
    (PropertyValue ?prop ?x ?z)
  )
;  (printout t "transitive" crlf)
)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(defrule intersection-of-implication
  "Defines the notion that intersectionOf defines subclass relationships
   between a set of classes and a class (intersectionOf is considered
   neccesary."

  (PropertyValue daml:intersectionOf ?c1 ?l)
  (list-item ?l ?c2)
=>
  (assert (PropertyValue rdfs:subClassOf ?c1 ?c2))
)

;;---------------------------------------------------------------------
(defrule intersection-of-reverse-implication
  "Defines the notion that membership in each class of an intersectionOf
```

```
   denotes membership in an outer class (intersectionOf is considered
   sufficient)."

  ; This has to fire after the other rules as it has a not.
  (declare (salience -50))

  (PropertyValue daml:intersectionOf ?rootClass ?list)
  (list-item ?list ?secondClass)
  (PropertyValue rdf:type ?instance ?secondClass)
  (not
    (and
      (list-item ?list ?thirdClass)
      (not (PropertyValue rdf:type ?instance ?thirdClass))
    )
  )
=>
  (assert (PropertyValue rdf:type ?instance ?rootClass))
;  (printout t "INTERSECTION on root " ?rootClass " with list " ?list
;              ", second class " ?secondClass " and instance " ?instance crlf)
)

;;-----------------------------------------------------------------------
(defrule intersection-of-subsumption
  "Implements a kind of structural subsumption for classes defined by
   intersectionOf elements."

  (declare (salience -50))

  (PropertyValue daml:intersectionOf ?topClass ?topList)
  (PropertyValue daml:intersectionOf ?botClass&~?topClass ?botList)

  ; foreach member of topList there is a member of botList which subclasses it
  ; there does not exist a member of topList which does not have a subclass in
  ; botList

  (not
    (and
      (list-item ?topList ?y)

      (not
        (or
          (list-item ?botList ?y)
          (and
            (list-item ?botList ?x)
            (PropertyValue rdfs:subClassOf ?x ?y)
          )
        )
      )
    )
  )

=>
  (assert (PropertyValue daml:subClassOf ?botClass ?topClass))

;  (printout t "Intersection subsumption: " ?botClass " subsumed by " ?topClass
;   crlf)
```

```
)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(defrule union-of-implication
  "Defines the notion that a unionOf is a superclass of each of its list
   members."

  (PropertyValue daml:unionOf ?c1 ?l)
  (list-item ?l ?c2)
=>
  (assert (PropertyValue rdfs:subClassOf ?c2 ?c1))
)

; Do we need a rule to handle the reverse case? I'm not convinced it's not
; handled by normal subsumption tasks

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(defrule closed-world-list-glomming
  (PropertyValue rdf:type ?l1 daml:List)
  (PropertyValue daml:rest ?l1 ?l2&~daml:nil)
  (list-item ?l2 ?foo)
=>
  (assert (list-item ?l1 ?foo))
  ; (printout t "LIST ADDED" crlf)
)

;;---------------------------------------------------------------------
(defrule closed-world-list-starting
  (PropertyValue rdf:type ?l1 daml:List)
  (PropertyValue daml:first ?l1 ?foo)
=>
  (assert (list-item ?l1 ?foo))
  ; (printout t "LIST BEGUN" crlf)
)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------

; These aren't going to work quite correctly on anonymous literals

(defrule has-value-restriction-forward

  (PropertyValue rdf:type ?class daml:Restriction)

  ; The hasValue comes before the other patterns to help scope the search
  ; efficiently
  (PropertyValue daml:hasValue ?class ?val)
  (PropertyValue daml:onProperty ?class ?prop)
  (PropertyValue rdf:type ?i ?class)
=>
  (assert (PropertyValue ?prop ?i ?val))
  ; (printout t "MUSHI1" crlf)

)
```

```
;;-------------------------------------------------------------------
(defrule has-value-restriction-reverse

  (PropertyValue rdf:type ?class daml:Restriction)

  ; The hasValue comes before the other patterns to help scope the search
  ; efficiently
  (PropertyValue daml:hasValue ?class ?val)
  (PropertyValue daml:onProperty ?class ?prop)
  (PropertyValue ?prop ?i ?val)
=>
  (assert (PropertyValue rdf:type ?i ?class))
  ; (printout t "MUSHI2" crlf)
)

;;-------------------------------------------------------------------
(defrule mincardinalityq-subsumption

  (PropertyValue rdf:type ?restriction1 daml:Restriction)
  (PropertyValue daml:onProperty ?restriction1 ?prop1)
  (PropertyValue daml:minCardinalityQ ?restriction1 ?lit1)
  (PropertyValue daml:hasClassQ ?restriction1 ?class1)
  (PropertyValue rdf:value ?lit1 ?val1)

  (PropertyValue rdf:type ?restriction2&~?restriction1 daml:Restriction)
  (PropertyValue daml:onProperty ?restriction2 ?prop2)
  (or
    (PropertyValue daml:minCardinalityQ ?restriction2 ?lit2)
    (PropertyValue daml:cardinalityQ ?restriction2 ?lit2)
  )
  (PropertyValue daml:hasClassQ ?restriction2 ?class2)
  (PropertyValue rdf:value ?lit2 ?val2)

  (test (and (integerp ?val1) (integerp ?val2) (>= ?val2 ?val1)))
; (and
  (or
  (test (eq ?class2 ?class1))
    (PropertyValue rdfs:subClassOf ?class2 ?class1)
  )

=>
;  (printout t "Mushi: " ?restriction2 " (" ?val2 "; " (integerp ?val2) "; "
;    ?class2 ") subsumed by " ?restriction1 " (" ?val1 "; " (integerp ?val2)
;    "; " ?class1 ")" crlf crlf)
  (assert (PropertyValue daml:subClassOf ?restriction2 ?restriction1))
)

;;-------------------------------------------------------------------
(defrule cardinalityq-subsumption

  (PropertyValue rdf:type ?restriction1 daml:Restriction)
  (PropertyValue daml:onProperty ?restriction1 ?prop1)
  (PropertyValue daml:cardinalityQ ?restriction1 ?lit1)
  (PropertyValue daml:hasClassQ ?restriction1 ?class1)
  (PropertyValue rdf:value ?lit1 ?val1)
```

```
  (PropertyValue rdf:type ?restriction2&~?restriction1 daml:Restriction)
  (PropertyValue daml:onProperty ?restriction2 ?prop2)
  (PropertyValue daml:cardinalityQ ?restriction2 ?lit2)
  (PropertyValue daml:hasClassQ ?restriction2 ?class2)
  (PropertyValue rdf:value ?lit2 ?val2)

  (test (and (integerp ?val1) (integerp ?val2) (= ?val2 ?val1)))
  (or
  (test (eq ?class2 ?class1))
    (PropertyValue rdfs:subClassOf ?class2 ?class1)
  )

=>
;  (printout t "Mushi: " ?restriction2 " (" ?val2 "; " (integerp ?val2) "; "
;    ?class2 ") subsumed by " ?restriction1 " (" ?val1 "; " (integerp ?val2)
;    "; " ?class1 ")" crlf crlf)
  (assert (PropertyValue daml:subClassOf ?restriction2 ?restriction1))
)
```

damljesskb-xsd-semantics.jess

```
;;---------------------------------------------------------------------
;; damljesskb-xsd-semantics.jess --- Definitions of XML Schema Datatypes
;;---------------------------------------------------------------------

; Don't forget that you should add in some error rules here...

;;---------------------------------------------------------------------
(defrule mininclusive-classification

  (PropertyValue rdf:type ?dt daml:Datatype)
  (PropertyValue xsd:minInclusive ?dt ?anon)
  (PropertyValue rdf:value ?anon ?value)

  (PropertyValue rdf:type ?inst rdf:Literal)
  (PropertyValue rdf:value ?inst ?ival)
  (test (and (integerp ?ival) (integerp ?value) (>= ?ival ?value)))

=>
   (assert (PropertyValue rdf:type ?inst ?dt))

; (printout t "DATATYPE: " ?dt "; " ?inst " " ?ival crlf)
)

;;---------------------------------------------------------------------
(defrule maxinclusive-classification

  (PropertyValue rdf:type ?dt daml:Datatype)
  (PropertyValue xsd:maxInclusive ?dt ?anon)
  (PropertyValue rdf:value ?anon ?value)

  (PropertyValue rdf:type ?inst rdf:Literal)
  (PropertyValue rdf:value ?inst ?ival)
  (test (and (integerp ?ival) (integerp ?value) (<= ?ival ?value)))
=>
```

```
   (assert (PropertyValue rdf:type ?inst ?dt))

; (printout t "DATATYPE: " ?dt "; " ?inst " " ?ival crlf)
)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(defrule mininclusive-subclassing
  (PropertyValue rdf:type ?dt1 daml:Datatype)
  (PropertyValue xsd:minInclusive ?dt1 ?anon1)
  (PropertyValue rdf:value ?anon1 ?value1)

  (PropertyValue rdf:type ?dt2&~?dt1 daml:Datatype)
  (PropertyValue xsd:minInclusive ?dt2 ?anon2)
  (PropertyValue rdf:value ?anon2 ?value2)

  (test (and (integerp ?value1) (integerp ?value2) (>= ?value1 ?value2)))

=>
; (printout t "SUB " ?dt1 "; " ?dt2 crlf)

  (assert (PropertyValue rdfs:subClassOf ?dt1 ?dt2))
)

;;---------------------------------------------------------------------
(defrule maxinclusive-subclassing
  (PropertyValue rdf:type ?dt1 daml:Datatype)
  (PropertyValue xsd:maxInclusive ?dt1 ?anon1)
  (PropertyValue rdf:value ?anon1 ?value1)

  (PropertyValue rdf:type ?dt2&~?dt1 daml:Datatype)
  (PropertyValue xsd:maxInclusive ?dt2 ?anon2)
  (PropertyValue rdf:value ?anon2 ?value2)

  (test (and (integerp ?value1) (integerp ?value2) (<= ?value1 ?value2)))
=>
; (printout t "SUB " ?dt1 "; " ?dt2 crlf)

  (assert (PropertyValue rdfs:subClassOf ?dt1 ?dt2))
)

;;---------------------------------------------------------------------
;;---------------------------------------------------------------------
(deffacts xsd-facts
  "Basic facts defining XSD stuffage."

  (PropertyValue rdf:type xsd:string rdfs:Class)
  (PropertyValue rdfs:subClassOf xsd:string rdf:Literal)

  (PropertyValue rdf:type xsd:number rdfs:Class)
  (PropertyValue rdf:type xsd:integer rdfs:Class)
  (PropertyValue rdf:type xsd:nonNegativeInteger rdfs:Class)

  (PropertyValue rdfs:subClassOf xsd:number rdf:Literal)
  (PropertyValue rdfs:subClassOf xsd:integer xsd:number)
  (PropertyValue rdfs:subClassOf xsd:nonNegativeInteger xsd:integer)
```

```
  (PropertyValue rdf:type xsd:nonNegativeInteger daml:Datatype)
  (PropertyValue xsd:minInclusive xsd:nonNegativeInteger xsd:xsd_anon1)
  (PropertyValue rdf:value xsd:xsd_anon1 0)

  (PropertyValue rdf:type xsd:positiveInteger daml:Datatype)
  (PropertyValue xsd:minInclusive xsd:positiveInteger xsd:xsd_anon2)
  (PropertyValue rdf:value xsd:xsd_anon2 1)

  (PropertyValue rdf:type xsd:nonPositiveInteger daml:Datatype)
  (PropertyValue xsd:maxInclusive xsd:nonPositiveInteger xsd:xsd_anon3)
  (PropertyValue rdf:value xsd:xsd_anon3 0)

  (PropertyValue rdf:type xsd:negativeInteger daml:Datatype)
  (PropertyValue xsd:maxInclusive xsd:negativeInteger xsd:xsd_anon4)
  (PropertyValue rdf:value xsd:xsd_anon4 -1)

)

;;-------------------------------------------------------------------
;;-------------------------------------------------------------------
```

Appendix E

VTB rules

VTB-rules.jess

```
(deffunction has_ports (?model ?query)
        (if (> (count-query-results ?query ?model) 0) then
                (return TRUE)
        else
                (return FALSE))
)


(defrule has-signal-backplane
        (PropertyValue rdf:type
                ?model
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (test (has_ports ?model check-signal-backplane))
=>
        (assert
                (PropertyValue HAS_BACKPLANE ?model B_SIGNAL)
        )
)


(defrule has-nature-backplane
        (PropertyValue rdf:type
                ?model
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (test (has_ports ?model nature_ports))
=>
        (assert
                (PropertyValue HAS_BACKPLANE ?model B_NATURE)
        )
)

(defrule has-quantity-backplane
        (PropertyValue rdf:type
                ?model
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (test (has_ports ?model check-quantity-backplane))
=>
        (assert
                (PropertyValue HAS_BACKPLANE ?model B_QUANTITY)
        )
)

(defrule has-power
        (PropertyValue rdf:type
```

```
                        ?model
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                        ?param
                        ?literal)
        (PropertyValue rdf:value
                        ?literal
                        ?unit)
        (test (or (string-search ?unit watt) (string-search ?unit watts)))
=>
        (assert
                (PropertyValue HAS_POWER ?model ?param)
        )
)

(defrule has-resistance
        (PropertyValue rdf:type
                        ?model
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                        ?param
                        ?literal)
        (PropertyValue rdf:value
                        ?literal
                        ?unit)
        (test (or (string-search ?unit ohm) (string-search ?unit ohms)))
=>
        (assert
                (PropertyValue HAS_RESISTANCE ?model ?param)
        )
)

(defrule has-voltage
        (PropertyValue rdf:type
                        ?model
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                        ?param
```

```
                    ?literal)
        (PropertyValue rdf:value
                    ?literal
                    ?unit)
        (test (or (string-search ?unit volt) (string-search ?unit volts) (string-search ?unit v)))
=>
        (assert
                (PropertyValue HAS_VOLTAGE ?model ?param)
        )
)

(defrule has-current
        (PropertyValue rdf:type
                    ?model
                    http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                    ?model
                    ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                    ?model
                    ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                    ?param
                    ?literal)
        (PropertyValue rdf:value
                    ?literal
                    ?unit)
        (test (or (string-search ?unit amp) (string-search ?unit amps) (string-search ?unit ampere) (string-
search ?unit a)))
=>
        (assert
                (PropertyValue HAS_CURRENT ?model ?param)
        )
)

(defrule has-inductance
        (PropertyValue rdf:type
                    ?model
                    http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                    ?model
                    ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                    ?model
                    ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                    ?param
                    ?literal)
        (PropertyValue rdf:value
                    ?literal
                    ?unit)
        (test (or (string-search ?unit henries) (string-search ?unit henry) ))
=>
        (assert
                (PropertyValue HAS_INDUCTANCE ?model ?param)
        )
```

```
)

(defrule has-frequency
        (PropertyValue rdf:type
                        ?model
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                        ?model
                        ?param)
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                        ?param
                ?literal)
        (PropertyValue rdf:value
                        ?literal
                        ?unit)
        (test (or (string-search ?unit hz) (string-search ?unit hertz) ))
=>
        (assert
                (PropertyValue HAS_FREQUENCY ?model ?param)
        )
)

(defrule sources
        (or
        (PropertyValue rdf:type
                                        ?thing
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-
Controls-Sources)
        (PropertyValue rdf:type
                                        ?thing
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-
ElectroMechanical-Motors)
        (PropertyValue rdf:type
                                        ?thing
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-
Electrical-Sources)
        )
=>
        (assert (PropertyValue rdf:type
                                                ?thing
                                                VTB_SOURCE)
        )
)

(defrule loads
        (or
        (PropertyValue rdf:type
                                        ?thing
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-
Mechanical-Loads)
        (PropertyValue rdf:type
                                        ?thing
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBModel-
```

```
Electrical-Loads)
        )
=>
        (assert (PropertyValue rdf:type
                                                ?thing
                                                VTB_LOAD)
        )
)
```

Appendix F

Jess Queries

VTB-queries.jess

```
;;
;;          CORE QUERIES
;;
;;

(defquery vtbmodel-subclasses
  "Gets all subclasses of VTBModel"
  (PropertyValue daml:subClassOf
                ?x
                http://vtb.engr.sc.edu/daml/VTBModels.daml#VTBObject
  )
)

(defquery class-subclasses
  "Gets all subclasses of VTBModel"
  (declare (variables ?class ?subclass))
  (PropertyValue daml:subClassOf
                ?subclass
                ?class
  )
)

(defquery class-type-exact
          "Searches for models a a specific class"
          (declare (variables ?class))
          (PropertyValue rdf:type
                                                ?model
                                                ?class
          )
)

(defquery class-query
          (declare (variables ?model ?class))
          (PropertyValue rdf:type
                    ?model
                    ?class)
)

(defquery keywords_literal
          (declare(variables ?model))
          (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasKeywords
                                                ?model
                                                ?literal_label
          )
```

```
)

(defquery nature_ports
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                ?model
                                ?port)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Nature)
)

(defquery signal_in
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                ?model
                                ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#INPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Signal)
)

(defquery signal_out
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                ?model
                                ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#OUTPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Signal)
)

(defquery signal_inout
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                ?model
                                ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port

        http://vtb.engr.sc.edu/daml/VTBModels.daml#INPUT_OUTPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Signal)
)

(defquery quantity_in
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                ?model
```

```
                                        ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#INPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Quantity)
)

(defquery quantity_out
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                        ?model
                                        ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#OUTPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Quantity)
)

(defquery quantity_inout
        (declare (variables ?model))
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                                        ?model
                                        ?port)
                (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasDirection
                                        ?port

        http://vtb.engr.sc.edu/daml/VTBModels.daml#INPUT_OUTPUT)
                (PropertyValue    rdf:type
                                        ?port
                                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Quantity)
)

(defquery model_comment
        (declare (variables ?model))
        (PropertyValue    http://vtb.engr.sc.edu/daml/VTBModels.daml#Comment
                                        ?model
                                        ?literal)
)

(defquery model_name
        (declare (variables ?model))
        (PropertyValue    http://vtb.engr.sc.edu/daml/VTBModels.daml#ModelName
                                        ?model
                                        ?literal)
)

(defquery model_resource
        (declare (variables ?model))
        (PropertyValue    http://vtb.engr.sc.edu/daml/VTBModels.daml#Resource
                                        ?model
                                        ?literal)
)
```

```
(defquery label_literal
        (declare (variables ?object))
        (PropertyValue rdfs:label
                        ?object
                        ?literal)
)

(defquery partnum_literal
        (declare (variables ?object))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPartNum
                                        ?object
                                        ?literal)
)

(defquery manu_literal
        (declare (variables ?object))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasManufacturer
                                        ?object
                                        ?literal)
)

(defquery get_params
        (declare (variables ?model))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasParameter
                                        ?model
                                        ?param)
)

(defquery get_param_type
        (declare (variables ?param))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasType
                                        ?param
                                        ?type)
)

(defquery get_param_value_lit
        (declare (variables ?param))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterValue
                                        ?param
                                        ?value)
)

(defquery get_param_unit_lit
        (declare (variables ?param))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#ParameterUnits
                                        ?param
                                        ?value)
)

(defquery get_has_model_lit
        (declare (variables ?model))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasModel
                                        ?model
                                        ?literal)
)
```

```
(defquery literal_value
        (declare (variables ?literal))
        (PropertyValue  rdf:value
                                        ?literal
                                        ?value)
)

(defquery check-signal-backplane
        (declare (variables ?model))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                ?model
                ?Port
        )
        (PropertyValue rdf:type
                        ?Port
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Signal
        )
)

(defquery check-nature-backplane
        (declare (variables ?model))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                ?model
                ?Port
        )
        (PropertyValue rdf:type
                        ?Port
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Nature
        )
)

(defquery check-quantity-backplane
        (declare (variables ?model))
        (PropertyValue http://vtb.engr.sc.edu/daml/VTBModels.daml#hasPort
                ?model
                ?Port
        )
        (PropertyValue rdf:type
                        ?Port
                        http://vtb.engr.sc.edu/daml/VTBModels.daml#Quantity
        )
)

(defquery signal_backplane_query
        (declare (variables ?model))
        (PropertyValue HAS_BACKPLANE ?model B_SIGNAL)
)

(defquery nature_backplane_query
        (declare (variables ?model))
        (PropertyValue HAS_BACKPLANE ?model B_NATURE)
)

(defquery quantity_backplane_query
        (declare (variables ?model))
```

```
        (PropertyValue HAS_BACKPLANE ?model B_QUANTITY)
)

(defquery source_query
        (declare (variables ?model))
        (PropertyValue rdf:type ?model VTB_SOURCE)
)

(defquery load_query
        (declare (variables ?model))
        (PropertyValue rdf:type ?model VTB_LOAD)
)


(defquery hasfrequency_query
        (declare (variables ?model))
        (PropertyValue HAS_FREQUENCY ?model ?param)
)

(defquery hasinductance_query
        (declare (variables ?model))
        (PropertyValue HAS_INDUCTANCE ?model ?param)
)

(defquery hascurrent_query
        (declare (variables ?model))
        (PropertyValue HAS_CURRENT ?model ?param)
)

(defquery hasvoltage_query
        (declare (variables ?model))
        (PropertyValue HAS_VOLTAGE ?model ?param)
)

(defquery hasresistance_query
        (declare (variables ?model))
        (PropertyValue HAS_RESISTANCE ?model ?param)
)

(defquery haspower_query
        (declare (variables ?model))
        (PropertyValue HAS_POWER ?model ?param)
)
```