



## THE SEMANTIC WEB

*Information with Knowledge*

---

**Intervise Consultants, Inc.**

10110 Molecular Drive Suite 100  
Rockville, MD 20850

Michael Priddy  
President & CEO  
mpriddy@intervise.com  
240.599-9301

## TABLE OF CONTENTS

<b>THE INFORMATION CHALLENGE</b>	<b>1</b>
<b>INFORMATION WITHOUT KNOWLEDGE</b>	<b>1</b>
<b>FEWER RESULTS = MORE MEANING</b>	<b>1</b>
<b>FOUR COMPONENTS OF THE SEMANTIC WEB</b>	<b>2</b>
<b>URI – UNIFORM RESOURCE IDENTIFIER</b>	<b>2</b>
<b>RDF – RESOURCE DESCRIPTION FRAMEWORK</b>	<b>2</b>
<b>RDF SCHEMA</b>	<b>2</b>
<b>ONTOLOGIES</b>	<b>3</b>
<b>THE SCIENCE - VARIABLE ONTOLOGY</b>	<b>3</b>
<b>ONTOLOGY</b>	<b>3</b>
XML Example 2: Expanded Ford Motor Company XML	5
XML Example 3: Manufacturers XML	7
OWL Example 1: Manufacturer Ontology in OWL	8
OWL Example 2: Adding the Sable to the Ontology	10
OWL Example 3: Adding Brake Pads to the Ontology	11
OWL Example 4: Adding four Wheel Drive Restrictions to cars	11
<b>APPLICATIONS OF RDF</b>	<b>13</b>
OWL Example 5: RDF XML OWL specificity and restriction.	13
RDF Example 1: RDF Meta Data example.	14
RSS Example 1: Interwise RSS feed.	14
<b>BENEFITS OF THE SEMANTIC WEB</b>	<b>15</b>
<b>PRACTICAL BENEFITS – AN EXAMPLE</b>	<b>15</b>

## **THE INFORMATION CHALLENGE**

The amount of information on the Web and within organizations has grown at an astounding rate; however, the technology used to accommodate and process this information has not. Computers still cannot automate this information or perform complex tasks. HTML can only provide a structure for text-based information in a document; it cannot do anything with it.

Different formats and structures make it difficult for computers or individuals to exchange data. Additionally, requirements change faster than we can solve the problems. Frequently, users are overwhelmed by too much unrelated data rather than the specific data required to solve their problem.

In order to meet this requirement, the technology must be capable of:

- Retrieving large amounts of textual data quickly.
- Allowing users to add annotations so that a reasoning capability exists.
- Making text retrieval more specific.
- Allowing conclusions to be drawn by data on the Web and across organizations.

## **INFORMATION WITHOUT KNOWLEDGE**

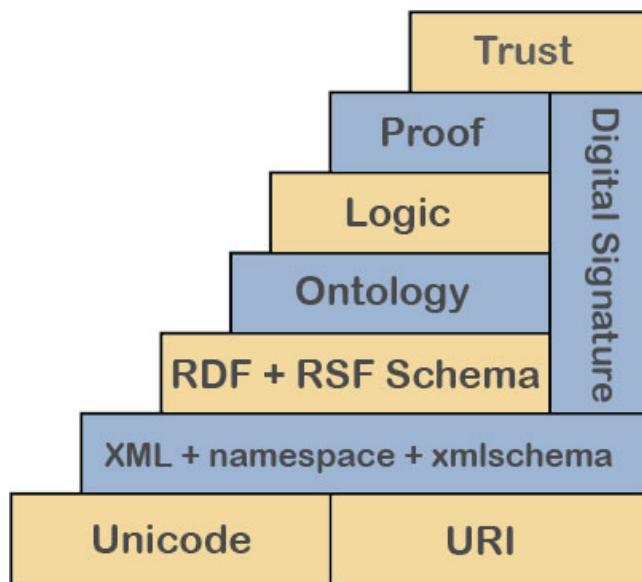
In the past, work focusing on attaching meaning to web-based knowledge produced mixed results. No solution capable of addressing the problem materialized. Because of this, even small successes were limited to local data hardwired into an isolated system. The result was no operability, as semantic data on a single system is neither discoverable nor accessible by other systems.

## **FEWER RESULTS = MORE MEANING**

Simply put, The Semantic Web identifies Web-based data to produce more efficient and productive Web searches. It is a growing extension of the World Wide Web, allowing web-based content to be expressed in natural, or human, language, as well as languages that can be read and utilized by automated systems. As a result, people and machines are able to find, share and utilize information conveniently.

The Semantic Web is the next generation of the current Web in which computers can interpret the meaning of web content because of explicit semantics provided in markup. The Semantic Web components are deployed in the layers of Web technologies and specifications.

Figure 1. The Semantic Web Layers



## FOUR COMPONENTS OF THE SEMANTIC WEB

### URI – UNIFORM RESOURCE IDENTIFIER

URI's are simple web identifiers that are often found on the World Wide Web (i.e. http, ftp). The Semantic Web is structured around syntaxes which utilize URI's as a naming convention to represent data.

### RDF – RESOURCE DESCRIPTION FRAMEWORK

RDF is used by The Semantic Web to describe data, thus allowing it to be shared more conveniently. It enables software developers to design products that will deploy better search engines by utilizing the metadata. As a result, users have more control over what they are viewing. Additionally, RDF is vocabulary agnostic which creates an interoperable environment capable of supporting a diverse range of ontologies.

### RDF SCHEMA

The RDF Schema is a language used by The Semantic Web to describe the data properties used in RDF.

## ONTOLOGIES

An ontology differs from traditional web schema in that it represents knowledge, not just a message format. The Semantic Web utilizes OWL, a web ontology language used by applications that need to process data rather than just display it. Semantic Web uses OWL to add reason to data by identifying and describing relationships between data items. OWL ontologies are capable of processing the content of information, rather than just presenting the data to users.

## **THE SCIENCE - VARIABLE ONTOLOGY**

The Semantic Web allows information to be processed automatically by tools as well as manually and can infer potential relationships among pieces of data. It extends principles of the World Wide Web from documents to data through the development of a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.

It provides for interchangeable format systems to exchange data removing the possibilities for rewrites when information changes and provides the mechanisms to represent Artificial Intelligence and the decision making process behind it. At its core, Semantic Web is a variable ontology.

## ONTOLOGY

Ontology is defined as a set of information or concepts and the relationships between the information/concepts within a domain. For instance a Ford Taurus may belong to the ontology describing vehicles produced by The Ford Motor Company. This is fairly easy to represent in a fixed ontology. Figure 2 describes the hierarchy of just the Ford Taurus.

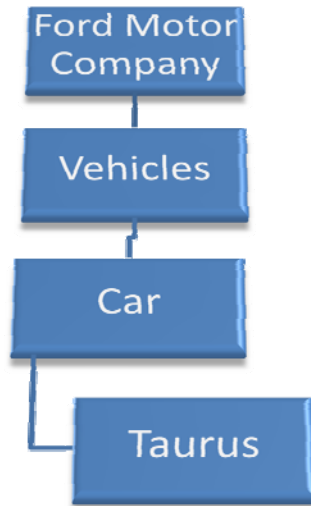


FIGURE 1 TAURUS HIERARCHY DIAGRAM

Figure 2 illustrates the hierarchical path. This is relatively simple to model in XML.

### **Example 1: Taurus XML Diagram**

```
<fordMotorCompany>
  <vehicles>
    <car>
      <taurus/>
    </car>
  </vehicles>
</fordMotorCompany>
```

Expanding the Ford Motor Company ontology will lead to a more formalized data structure and with implied relationships between different pieces of data. For instance Ford produces more than one type of vehicle, and they also make parts for the vehicles they manufacture. Look at what happens when the F-150 Truck and Expedition Sport Utility Vehicle and some vehicle parts like a chassis and engine for each are added.

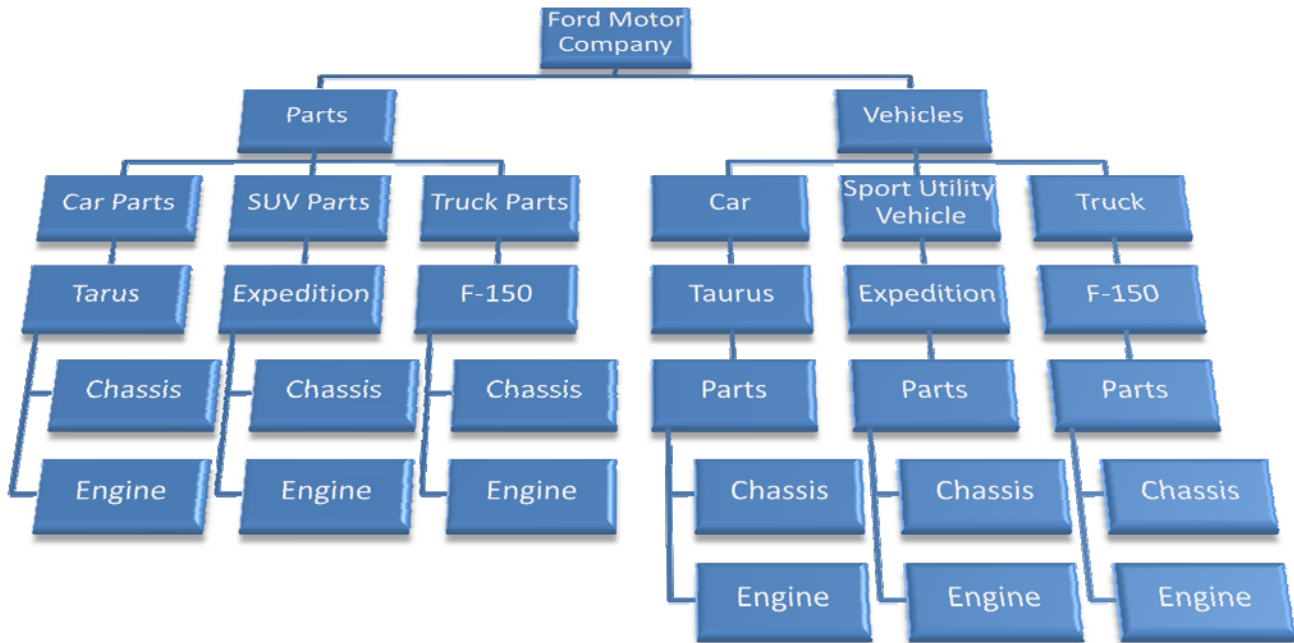


FIGURE 3 TAURUS HIERARCHY DIAGRAM TWO

Based on the diagram below, information in the XML will be repeated to insure that the relationship between parts and vehicles can be shown. The width and depth of the diagram has increased dramatically which will also increase the depth and complexity of the XML representation.

**XML Example 2: Expanded Ford Motor Company XML**

```

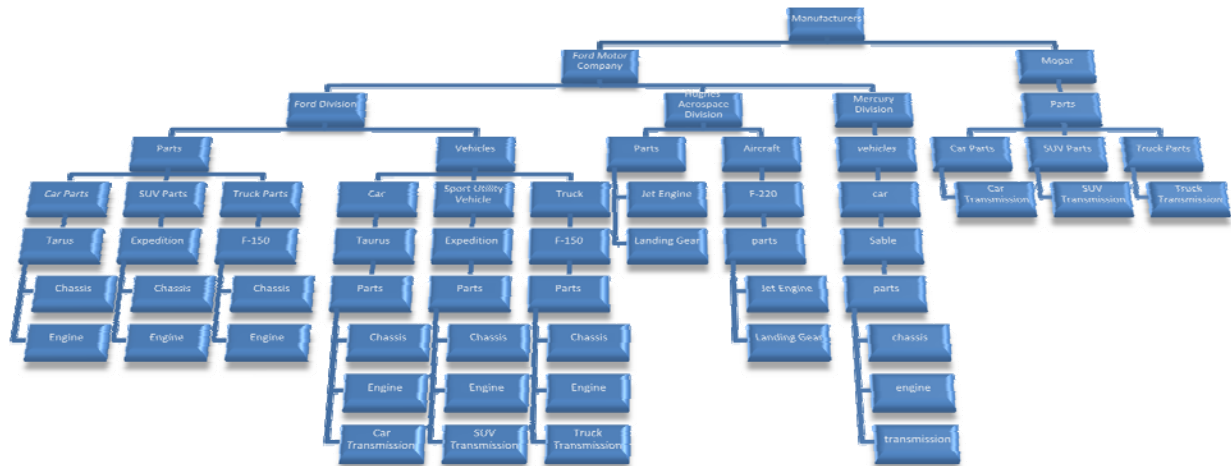
<FordMotorCompany>
  <parts>
    <carParts>
      <taurus>
        <chassis/>
        <engine/>
      </taurus>
    </carParts>
    <suvParts>
      <expedition>
        <chassis/>
        <engine/>
      </expedition>
    </suvParts>
    <truckParts>
      <f150>
        <chassis/>
        <engine/>
      </f150>
    </truckParts>
  </parts>
  <vehicles>
    <car>
      <taurus>
        <parts>
          <chassis/>
          <engine/>
        </parts>
      </taurus>
    </car>
    <sportUtilityVehicle>
      <expedition>
        <parts>
          <chassis/>
          <engine/>
        </parts>
      </expedition>
    </sportUtilityVehicle>
    <truck>
      <f150>
        <parts>
          <chassis/>
          <engine/>
        </parts>
      </f150>
    </truck>
  </vehicles>
</FordMotorCompany>

```

```
</suvParts>
<truckParts>
  <f150>
    <chassis/>
    <engine/>
  </f150>
</truckParts>
</parts>
<vehicles>
  <car>
    <taurus>
      <parts>
        <truck>
          <f150>
            <parts>
              <chassis/>
              <engine/>
            </parts>
          </f150>
        </truck>
      </vehicles>
    </fordMotorCompany>
  </car>
</vehicles>
</parts>
</suvParts>
```

What happens when Ford Motor Company uses an outside parts manufacturer to supply a part for one of the vehicles it manufactures? What happens when Ford produces another vehicle under a different make and model like the Mercury Sable that is virtually identical to the Taurus? What happens when Ford Motor Company purchases Hughes Aerospace and begins producing aircraft and aircraft parts?

FIGURE 4 MANUFACTURERS DIAGRAM



**XML Example 3: Manufacturers XML**

```

                                <truckTransmission/>
<manufacturers>                </parts>
  <fordMotorComapny>            </f150>
    <fordDivision>              </truck>
      <parts>                    </vehicles>
        <carParts>              </fordDivision>
          <taurus>               <hughesAerospaceDivision>
            <chassis/>           <parts>
              <engine/>         <jetEngine/>
            </taurus>           <landingGear/>
          </carParts>           </parts>
        <suvParts>              <vehicles>
          <expedition>          <f220>
            <chassis/>          <parts>
              <engine/>         <jetEngine/>
            </expedition>      <landingGear/>
          </suvParts>          </parts>
        <truckParts>           </f220>
          <f150>                 </vehicles>
            <chassis/>          </hughesAerospaceDivision>
              <engine/>        <mercuryDivision>
            </f150>            <vehicles>
          </truckParts>        <Sable>
        </parts>              <parts>
      <vehicles>              <chassis/>
        <car>                  <engine/>
          <taurus>              <carTransmission/>
            <parts>            </parts>
              <chassis/>      </Sable>
                <engine/>    </vehicles>
              <carTransmission/> </mercuryDivision>
            </parts>          </fordMotorcompany>
          </taurus>          <mopar>
        </car>              <parts>
      <suv>                  <carParts>
        <expedition>          <carTransmission/>
          <parts>            </carParts>
            <chassis/>      <suvParts>
              <engine/>    <suvTransmission/>
            <suvTransmission/> </suvParts>
          </parts>          <truckParts>
        </expedition>      <truckTransmission/>
      </suv>                </truckParts>

```

```

<truck>
  <f150>
    <parts>
      <chassis/>
      <engine/>
    </parts>
  </f150>
</truck>
</mopar>
</manufacturers>

```

Adding a few more items of information has significantly increased the size and complexity of the XML and provided no real guidance as to the relationship between vehicles and parts. Consumers of the XML would have to have some previous understanding as to the meaning of the XML to interpret and manipulate the data properly. If the XML changes, the consumer would then have to reevaluate the way in which the data is processed.

Several different methodologies have been utilized to relay data between two user agents. Various levels of understanding of data could require this communication to be very tightly controlled with little to no variation in data type and structure. Providing access to variable ontology as opposed to fixed ontology can serve data to consumers as well as provide the definition of what the data means and how it relates to an ontology as a whole. This generic format and format definition allows the consumer to determine how to utilize the information from the supplied ontological definition as opposed to a using predefined understanding of the data.

### **OWL Example 1: Manufacturer Ontology in OWL.**

```

<rdf:RDF
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
  xmlns:rdflib="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:vccard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="http://www.purl.org/net/ontology/manufacturers#"
>
  <owl:Ontology rdf:about=""/>
    <owl:versionInfo>Sample OWL
ontology</owl:versionInfo>
    <rdfs:label>Manufacturer example OWL</rdfs:label>

    <ManufacturesCar rdf:ID="#Sable"/>
    <ManufacturesSuv rdf:ID="#Expedition"/>
    <ManufacturesTruck rdf:ID="#F150"/>
    <ManufacturesPart rdf:ID="#CarChassis"/>
    <ManufacturesPart rdf:ID="#SuvChassis"/>
    <ManufacturesPart
rdf:ID="#TruckChassis"/>
    <ManufacturesPart rdf:ID="#CarEngine"/>
    <ManufacturesPart rdf:ID="#SuvEngine"/>
    <ManufacturesPart
rdf:ID="#TruckEngine"/>
    <ManufacturesPart rdf:ID="#JetEngine"/>
    <ManufacturesPart
rdf:ID="#LandingGear"/>
  </Manufacturer>
  <Manufacturer rdf:ID="Mopar">
    <ManufacturesPart
rdf:ID="#CarTransmission"/>
    <ManufacturesPart

```

```

<owl:Ontology>
<owl:Class rdf:ID="Manufacturer"/>
<owl:Class rdf:ID="Aircraft"/>
<owl:Class rdf:ID="Car"/>
<owl:Class rdf:ID="Suv"/>
<owl:Class rdf:ID="Truck"/>
<owl:Class rdf:ID="Part"/>
<owl:ObjectProperty rdf:ID="AircraftManufacturedBy">
  <owl:inverseOf
rdf:resource="#ManufacturesAircraft"/>
  <rdfs:domain rdf:resource="#Aircraft"/>
  <rdfs:range rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="CarManufacturedBy">
  <owl:inverseOf rdf:resource="#ManufacturesCar"/>
  <rdfs:domain rdf:resource="#Car"/>
  <rdfs:range rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="PartManufacturedBy">
  <owl:inverseOf rdf:resource="#ManufacturesPart"/>
  <rdfs:domain rdf:resource="#Part"/>
  <rdfs:range rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="SuvManufacturedBy">
  <owl:inverseOf rdf:resource="#ManufacturesSuv"/>
  <rdfs:domain rdf:resource="#Suv"/>
  <rdfs:range rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="TruckManufacturedBy">
  <owl:inverseOf rdf:resource="#ManufacturesTruck"/>
  <rdfs:domain rdf:resource="#Truck"/>
  <rdfs:range rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesAircraft">
  <rdfs:range rdf:resource="#Aircraft"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesCar">
  <rdfs:range rdf:resource="#Car"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesPart">
  <rdfs:range rdf:resource="#Part"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesSuv">
  <rdfs:range rdf:resource="#SuvTransmission"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesTruck">
  <rdfs:range rdf:resource="#TruckTransmission"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="F220">
  <AircraftManufacturedBy
rdf:ID="#FordMotorCompany"/>
  <AircraftPart rdf:ID="#JetEngine"/>
  <AircraftPart rdf:ID="#LandingGear"/>
</Aircraft>
<Car rdf:ID="Taurus">
  <CarManufacturedBy
rdf:ID="#FordMotorCompany"/>
  <CarPart rdf:ID="#CarTransmission"/>
  <CarPart rdf:ID="#CarChassis"/>
  <CarPart rdf:ID="#CarEngine"/>
</Car>
<Part rdf:ID="JetEngine">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="LandingGear">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="CarEngine">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="TruckEngine">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="SuvEngine">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="CarChassis">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="SuvChassis">
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
</Part>
<Part rdf:ID="TruckChassis">

```

```

<rdfs:range rdf:resource="#Suv"/>
<rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ManufacturesTruck">
  <rdfs:range rdf:resource="#Truck"/>
  <rdfs:domain rdf:resource="#Manufacturer"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:resource="AircraftPart"/>
  <rdfs:range rdf:resource="#Part"/>
  <rdfs:domain rdf:resource="#Aircraft"/>
</owl>
<owl:ObjectProperty rdf:resource="CarPart"/>
  <rdfs:range rdf:resource="#Part"/>
  <rdfs:domain rdf:resource="#Car"/>
</owl>
<owl:ObjectProperty rdf:resource="SuvPart"/>
  <rdfs:range rdf:resource="#Part"/>
  <rdfs:domain rdf:resource="#Suv"/>
</owl>
<owl:ObjectProperty rdf:resource="TruckPart"/>
  <rdfs:range rdf:resource="#Part"/>
  <rdfs:domain rdf:resource="#Truck"/>
</owl>
<Manufacturer rdf:ID="FordMotorCompany">
  <ManufacturesAircraft rdf:ID="#F220"/>
  <ManufacturesCar rdf:ID="#Taurus"/>
  <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
  </Part>
  <Part rdf:ID="CarTransmission">
    <PartManufacturedBy rdf:ID="#Mopar"/>
  </Part>
  <Part rdf:ID="SuvTransmission">
    <PartManufacturedBy rdf:ID="#Mopar"/>
  </Part>
  <Part rdf:ID="TruckTransmission">
    <PartManufacturedBy rdf:ID="#Mopar"/>
  </Part>
  <Suv rdf:ID="Expedition">
    <SuvManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <SuvPart rdf:ID="#SuvTransmission"/>
    <SuvPart rdf:ID="#SuvChassis"/>
    <SuvPart rdf:ID="#SuvEngine"/>
  </Suv>
  <Truck rdf:ID="F150">
    <TruckManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <TruckPart rdf:ID="#TruckTransmission"/>
    <TruckPart rdf:ID="#TruckChassis"/>
    <TruckPart rdf:ID="#TruckEngine"/>
  </Truck>
</rdf:RDF>

```

In the previous XML examples, the ontology is implied by the XML's structure itself or perhaps a DTD or namespace could further define what does and does not belong within each corresponding tag. What happens to the OWL when two identical cars sold under a different make and models are present in the ontology? Add the Sable pointing outward. It is the same as the Taurus with no repeat of information.

**OWL Example 2: Adding the Sable to the Ontology.**

```

<rdf:RDF
...
  <Car rdf:ID="Sable">
    <owl:sameAs rdf:resource="#Taurus"/>
  </Car>
...
</rdf:RDF>

```

What happens when a new part needs to be represented? Add the part and the associations to vehicles.

### OWL Example 3: Adding Brake Pads to the Ontology

```

<rdf:RDF
...
  <Manufacturer rdf:ID="Mopar">
    <ManufacturesPart rdf:ID="#CarTransmission"/>
    <ManufacturesPart rdf:ID="#SuvTransmission"/>
    <ManufacturesPart
rdf:ID="#TruckTransmission"/>
    <ManufacturesPart rdf:ID="#BrakePad"/>
  </Manufacturer>
...
  <Aircraft rdf:ID="F220">
    <AircraftManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <AircraftPart rdf:ID="#JetEngine"/>
    <AircraftPart rdf:ID="#LandingGear"/>
    <AircraftPart rdf:ID="#BrakePad"/>
  </Aircraft>
  <Car rdf:ID="Taurus">
    <CarManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <CarPart rdf:ID="#CarTransmission"/>
    <CarPart rdf:ID="#CarChassis"/>
    <CarPart rdf:ID="#CarEngine"/>
    <CarPart rdf:ID="#BrakePad"/>
  </Car>
  <Car rdf:ID="Sable">
    <owl:sameAs rdf:resource="#Taurus"/>
  </Car>
...
  <Suv rdf:ID="Expedition">
    <SuvManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <SuvPart rdf:ID="#SuvTransmission"/>
    <SuvPart rdf:ID="#SuvChassis"/>
    <SuvPart rdf:ID="#SuvEngine"/>
    <SuvPart rdf:ID="#BrakePad"/>
  </Suv>
  <Truck rdf:ID="F150">
    <TruckManufacturedBy
rdf:ID="#FordMotorCompany"/>
    <TruckPart rdf:ID="#TruckTransmission"/>
    <TruckPart rdf:ID="#TruckChassis"/>
    <TruckPart rdf:ID="#TruckEngine"/>
    <TruckPart rdf:ID="#BrakePad"/>
  </Truck>
...
</rdf:RDF>

```

Relationships between data can be further defined by providing Boolean relationships and property restrictions. For example, a four wheel drive part available to vehicles can be added, but its availability can be limited to cars.

### OWL Example 4: Adding four Wheel Drive Restrictions to cars

```

<rdf:RDF
...
  <owl:Class rdf:ID="Car">
    <rdfs:subClassOf>
      <owl:onProperty
rdf:resource="#CarPart"/>
      <owl:Restriction>
        <owl:hasValue
rdf:resource="#TwoWheelDrive"/>
      </owl:Restriction>
...
  <Part rdf:ID="TwoWheelDrive">
    <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
  </Part>
  <Part rdf:ID="FourWheelDrive">
    <PartManufacturedBy
rdf:ID="#FordMotorCompany"/>
  </Part>
...

```

```
</rdfs:subClassOf>
</owl:Class>
```

```
<rdf:RDF>
```

Raw XML would have to be retooled and repeated to reflect the changes made above, and perhaps the DTD or namespace would have to be altered to reflect said change depending on the structure utilized. The consumer would then have to take into account any changes in the XML when they are processing the received data. If changes could be introduced instead, with a little guidance as to what and where the new data applies we could change things on the fly. The data can not only be provided, but the relationship between the data resources can as well, allowing the consumer to concentrate on manipulating and reporting on the data as opposed to building and manipulating interim data structures.

The diagram below shows how the OWL has simplified the data representation cleaning up the duplication and establishing the relationships between the different data objects.

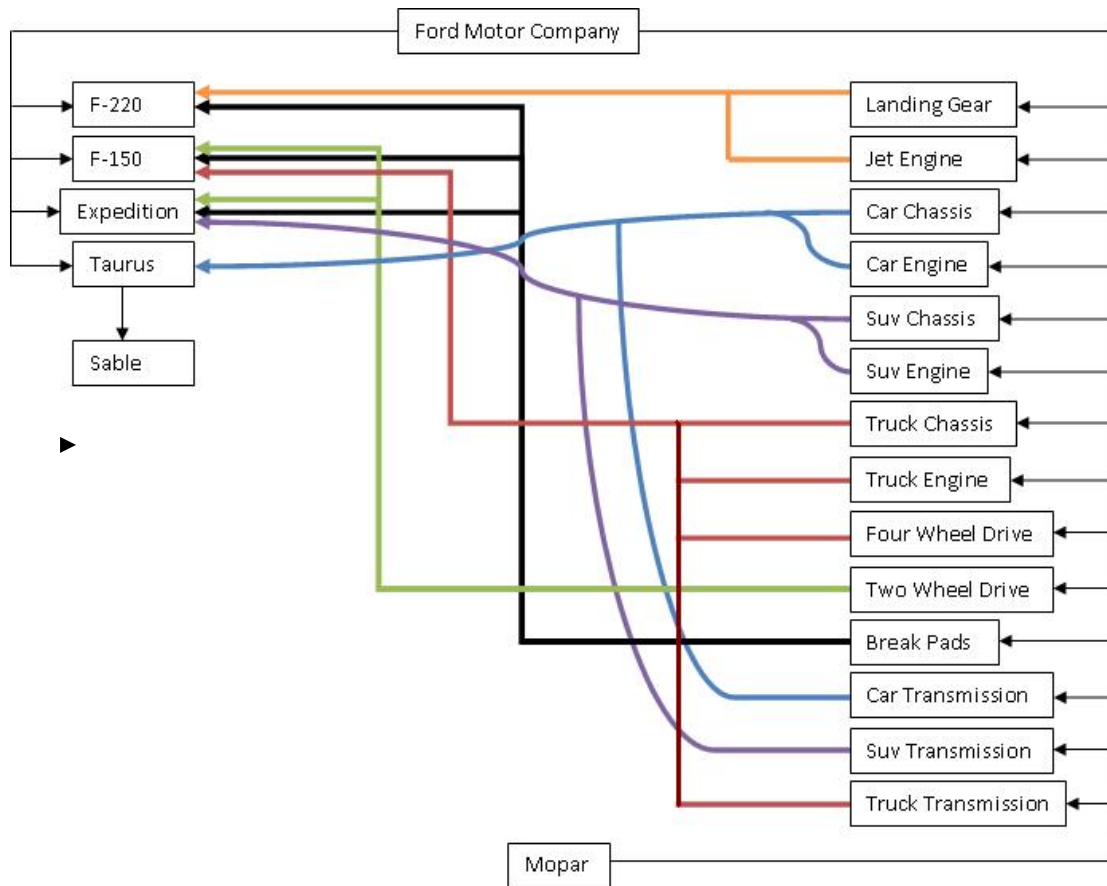


FIGURE 5 NEW DATA RELATIONSHIPS

**APPLICATIONS OF RDF**

RDF provides a standardized method to make a statement about anything. In RDF you relate a resource property to a value or another resource using a triplet structure of subject, predicate, object.

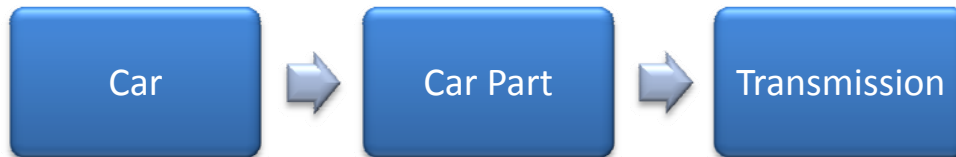


FIGURE 6 RDF RESOURCE TO RESOURCE STATEMENT



FIGURE 7 RESOURCE TO VALUE STATEMENT

These RDF statements are understandable by machines allowing for processing by system and user agents but lack more complex relationships such as value restriction. Because RDF is only capable of making simple statements about object and object relationships, it needs to be coupled with XML to provide the constraints and complexities of relational data. When combining RDF with XML RDF is enhanced with the ability to incorporate namespaces and schemas to further restrict and define the data relationships. The OWL example below shows the relation of the Sable to the Taurus and the restriction of car having to have at least one Two Wheel Drive part.

**OWL Example 5: RDF XML OWL specificity and restriction**

```

<rdf:RDF
...
  <owl:Class rdf:ID="Car">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty
...
          </owl:Restriction>
        </rdfs:subClassOf>
      </owl:Class>
    ...
    <Car rdf:ID="Sable">
      <owl:sameAs rdf:resource="#Taurus"/>
  
```

```

rdf:resource="#CarPart"/>          </Car>
  <owl:hasValue
rdf:resource="#TwoWheelDrive"/>    ...
                                   </rdf:RDF>

```

There are several real world uses for RDF. Website metadata was at one time considered the major case for the creation of RDF. An agent could use sample RDF Example 1 to determine if **subject** and **type** set by the RDF apply to a consumers search for consulting or recruiting agencies.

**RDF Example 1: RDF Meta Data example.**

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:DC="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.intervise.com/">
    <DC:Type>Homepage</DC:Type>
    <DC:Subject>Interwise Consultants Inc, Federal Consultants, Project Management, HR
    Recruiting</DC:Subject>
  </rdf:Description>
</rdf:RDF>

```

News aggregation is another example of RDF’s real world application. RDF Site Summary (RSS) is commonly used by small and large news agencies to publish news that is easy to plug into more elaborate websites. Products and projects such as Thunderbird, Microsoft Outlook, and Google Reader are examples of web based and client based news aggregators that “subscribe” to site supplied RSS feeds to digest new information as it is presented to the feed and display it to an end user.

**RSS Example 1: Intervise RSS feed**

```

<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>Interwise Consultants RSS</title>
    <description>Interwise Consultants RSS
    feed</description>
    <link>http://www.intervise.com/rss</link>
    <lastBuildDate>Mon, 17 Mar 2008
    11:12:55 -0400 </lastBuildDate>
    <pubDate>Mon, 17 Mar 2008 11:12:55 -
    0400</pubDate>
  </channel>
  <item>
    <title>Variable Ontology White Paper</title>
    <description>Document describing Intervise’s approach
    to the Semantic Web</description>
    <link>http://www.intervise.com/whitepapers/variableOnt
    ology.xhtml</link>
    <guid isPermaLink="false"> 1232523</guid>
    <pubDate>Mon, 17 Mar 2008 11:12:55 -
    0400</pubDate>
  </item>
</rss>

```

RDF has also been tightly related to Artificial Intelligence (AI) because of the problems it can be used to solve such as searches, cataloging data. RDF's triplet structure, predicate, subject, object is easy to represent in languages such as prologue again lending to AI appropriate structure and usability.

## BENEFITS OF THE SEMANTIC WEB

- Information is captured in a language agnostic format.
- A central repository for knowledge is created.
- More precise, relevant information is captured.
- Processes and procedures are mapped to data sources.
- One collective view of knowledge across enterprise applications is created.

As a result:

- Point-to-point integration becomes obsolete.
- Application integration is easy and efficient.
- Superfluous data decreases.
- knowledge across applications becomes consistent.
- Upgrades and maintenance are simplified.

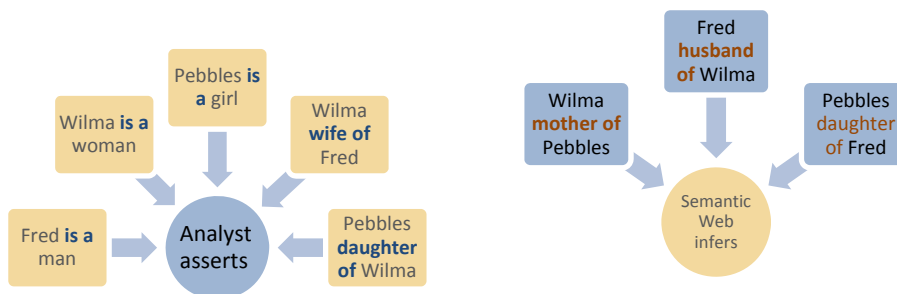


Figure 2. Example of Web

## PRACTICAL BENEFITS – AN EXAMPLE

An employee at a health-related organization is searching for information on a patient with very little search criteria (i.e. gender and birth year). A regular search will pull countless documents that the employee will have to work her way through in order to isolate the specific information she is looking for.

If the organization's information is streamlined ally, the simple search component will make associations and produce fewer results containing more relevant information from departments, applications and systems across the enterprise. The result is more precise information found using one simple search – relevant knowledge in less time.

Semantic Web is the next generation in information management. To begin the process of adding knowledge to your information, please contact us at 240.599.9300 or [information@interwise.com](mailto:information@interwise.com).