

# How can ontologies help repair your car?

*Martin Bryan, Senior Technical Consultant, CSW Group Ltd*

*Reuben Wright, Technical Consultant, CSW Group Ltd*

## **Abstract**

The MYCAREVENT pan-European project is, among other things, exploring ways in which ontologies might be used to improve the efficiency of roadside assistance and workshop mechanics by providing them with access to relevant repair information and parts. This information must be delivered at the point of repair, which may be at the roadside, so access will be provided to the system via mobile communications networks. Our services must identify solutions using terms related to the fault described by the owner or mechanic rather than the terms manufacturers have traditionally provided in the contents lists and indexes of their repair manuals and parts lists.

The terms used by all three parties to the repair process - car manufacturers, owners and repairers - need to be recorded in a way that computers can use to manage searches for information. Because we need to understand the relationship between terms used by different communities we need to define the terms in an ontology rather than a simple classification scheme. *Ontologies* are networks of related terms that can be navigated to find information. They differ from the hierarchically organized *taxonomies* typically used by car manufacturers to organize data, such as exploding part lists and repair manuals, in that they allow multiple relationships to be defined from each term. Ontologies also differ from the thesauri that have traditionally been used to allow users to identify that the "trunk of an American sedan" is the same thing as the "boot of an English saloon", and from the multilingual dictionaries that are used to identify that the German term for a car boot is kofferraum.

Ontologies allow all the traditional types of relationships to be defined in a unified way, and also allow new forms of relationships to be defined where needed. For example, a part listed in an ontology might have a relationship that records the fault code that an on-board diagnostic tool will receive if that part fails on a particular model. It could also have a relationship to a symptom that will be detectable if the part starts to perform incorrectly rather than fail completely.

## **What's wrong with my Car?**

Have you ever stopped to think how you can describe what's wrong with your car to someone who does not speak the same language as yourself, like your garage mechanic? You, as the car's owner, might think that your car "Won't start". Your mechanic might think "He can't activate the starter motor." You describe a symptom that best describes what the effect on the car is while your mechanic is seeking to identify the component that is the cause of the problem.

There can be a number of reasons why a particular symptom could appear. Just consider the possibilities for something as simple as "Won't start". Maybe the ignition switch has failed, or the immobiliser has not been switched off, or the battery is flat, or the power supply to the starter motor has failed, ... , or you have simply run out of petrol. The list of possible causes for a single symptom can be very long. Your mechanic will want you to provide clues as to the context in which the problem has occurred that will allow the range of relevant options to be narrowed.

Sometimes you may be lucky, and the first cause that the mechanic thinks of is the correct one. Hopefully he will know from experience which is the most common reason for this symptom appearing on your specific vehicle. Or maybe the manufacturer will allow him access to information held in his database on the most likely cause. If your car is modern enough you may have on-board diagnostics which are able to report a specific error in a clearly identified component. If the diagnostic trouble code does not identify the component causing the problem

your mechanic may have access to information on which tests need to be carried out to identify the relevant cause.

While other projects, such as SAMOVAR [SAM], have studied the vocabularies used in car production scenarios, the unique feature of MYCAREVENT is that it will study the vocabularies of car repairers and owners, and determine how these can be related to the terms used by different car manufacturers to describe problems with components and the solutions that are available to overcome these problems.

### ***How can ontologies help?***

For MYCAREVENT the Web Ontology Language [OWL] is expected to form a basis for recording the relationships between symptoms, the problems they identify and solutions that could be applied to correct the problem.

The top-level structure of the vehicle-related parts of the model we are developing is outlined in Figure 1.



Figure 1: Top Level Structure of Vehicle-related Classes used for MYCAREVENT

### A Brief Overview of OWL

The Web Ontology Language (OWL) is an application of the World Wide Web Consortium's Resource Description Framework (RDF). OWL ontologies are enclosed in RDF wrappers, and make use of RDF identifiers (`rdf:ID`) and related RDF properties such as `rdf:datatype`. OWL

annotations are defined using RDF Schema (`rdfs`) comments and labels. Dublin Core (`dc`) attributes can be used to assign metadata relating to the provenance of specific OWL statements.

Our first example illustrates how the start of a MYCAREVENT ontology could be defined.

```
<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:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.mycarevent.com/WP3-Ontology#"
  xml:base="http://www.mycarevent.com/WP3-Ontology">
  <owl:Ontology rdf:about="" />
  <owl:Class rdf:ID="Cars">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Vehicles" />
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Vehicles used for passenger transport
  </rdfs:comment>
  </owl:Class>
```

### Example 1: Class Definition at start of OWL Ontology

Elements in the OWL (`owl`) namespace identify the logical components of the ontology. As well as declaring classes using `owl:Class` elements, users can create instances (individuals) of each class using elements whose name is the name of the class the individual belongs to, as is illustrated in our second example:

```
<VehicleSteeringAndSuspensionProblems rdf:ID="fiat.panda.H26">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    No Power Assisted Steering: No Warning Light
  </rdfs:comment>
  <rdfs:label xml:lang="it" company="fiat">
    MANCANZA DI ASSERVIMENTO DELLA GUIDA ELETTRICA
    SENZA ACCENSIONE DELLA SPIA AVARIA
  </rdfs:label>
  <rdfs:label xml:lang="en" company="fiat">
    NO ELECTRIC STEERING POWER ASSISTANCE WITH FAILURE WARNING LIGHT NOT ON
  </rdfs:label>
  <correctUsing>
    <DiagnosticTests rdf:ID="fiat.panda.H26.test">
      <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        Diagnostic tests for when power steering on Fiat Panda is
        not working but no warning light is on
      </rdfs:comment>
      <faultsCorrected rdf:resource="#fiat.panda.H26" />
    </DiagnosticTests>
  </correctUsing>
</VehicleSteeringAndSuspensionProblems>
```

Formatiert: Italienisch (Italien)

### Example 2: Related Instances of Classes defined within OWL Ontology

Each individual entry also has a unique identifier, and can optionally be given one or more labels in different languages, or a definition in the form of an `rdfs:comment`. The ontology can also identify that an individual is related to another one which, in our example, records how the problem can be corrected. Note that OWL permits related entries to be nested inside one another.

OWL can also define the properties of classes, and the relationships classes, properties and individuals have with one another. This is typically done using:

- Object properties, which can be used to relate an instance of a class to an instance of another (or the same) class, are illustrated in our next example, which shows how the `faultsCorrected` property used in the above example has been defined. The first related class is called the domain, the second is called the range,

```
<owl:ObjectProperty rdf:about="#faultsCorrected">
  <rdfs:domain rdf:resource="#FaultRepairOptions"/>
  <rdfs:range rdf:resource="#FaultDescriptions"/>
  <owl:inverseOf rdf:resource="#correctUsing"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Identifies faults that can be cured using fault repair option
  </rdfs:comment>
</owl:ObjectProperty>
```

#### Example 3: Object Property Definitions

Note that, in Example 3, there is a second property definition, for, `correctUsing`, that identifies the inverse of the `faultsCorrected` property.

- Datatype properties, which can be used to restrict an individual member of a class to a specified datatype range, as illustrated in our next example:

```
<owl:DatatypeProperty rdf:ID="PartName">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Name used to identify product part in documentation
  </rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#ProductParts"/>
</owl:DatatypeProperty>
```

#### Example 4: Datatype Property Definition

- Functional properties, which are properties for which no more than one value can exist for each instance, as in the case of the `bodyType` property, which ensures a vehicle only has one body type:

```
<owl:FunctionalProperty rdf:ID="PaymentContractID">
  <rdfs:domain rdf:resource="#SystemUsers"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Unique identifier of contract to which access is to be billed
  </rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
```

#### Example 5: Functional Property Definition

- Inverse Functional Properties properties, which identify bidirectional relationships between classes, as illustrated in Example 6.

```

<owl:InverseFunctionalProperty rdf:about="#makes">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Products"/>
        <owl:Class rdf:about="#ProductParts"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Makes product
  </rdfs:comment>
  <owl:inverseOf rdf:resource="#isMadeBy"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Manufacturers"/>
        <owl:Class rdf:about="#ManufacturingGroups"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>

```

### Example 6: Inverse Functional Property Definition

OWL also provides `TransitiveProperty` and `SymmetricProperty` options.

Once we have defined the relationships that can exist between classes and individuals we can start to use these relationships to navigate between sets of information. As Figure 2 shows, a fault diagnostic test can be documented in many different information resources (one for each language), require one or more tools to carry out the test, and be used to correct one or more faults.

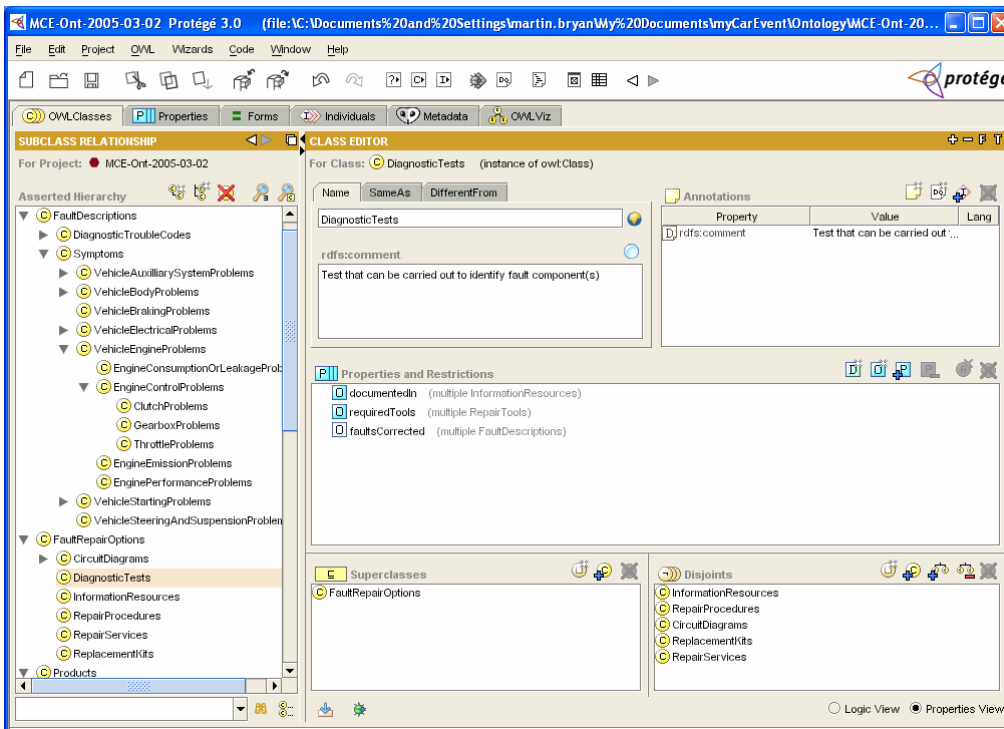


Figure 2: Definition of Diagnostic Test

### ***How did we do that?***

The initial ontology model has been defined using the OWL plug-in for Protégé but, as this is a general purpose tool designed for used by trained ontologists, Protégé is not ideal for allowing end-users to populated the ontology. For long-term maintenance of the MYCAREVENT ontology a dedicated tool set will be developed that can be used by unskilled users to relate product and parts lists with diagnostic trouble codes, diagnostic tests and repair procedures.

When working with OWL it is important to understand the difference between the "open world" viewpoint used by OWL and the "closed world" viewpoint of more traditional ontological tools (see [MAN] for an overview of the types of problems this introduces). OWL is based on the fact that only a partial world view can be provided in an ontology, and that not all possible situations can be described. Because of this the way in which OWL-based reasoners processes statements based on concepts such as "and" or "or" significantly differs from those used in natural languages. A typical problem is that users need to specifically state both that a property has "someValuesFrom" a class and that the same class provides "allValuesFrom" which the property can be populated before a decision logic tool can infer that the value of a property must come from a specific class. Any toolkit developed to allow users to define classes and individuals needs to automate many of the rules required by OWL to achieve a logically consistent set of constraints.

## **References**

[MAN] Rector, A et al, *OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns*, <http://www.co-ode.org/resources/papers/ekaw2004.pdf>

[OWL] *OWL Web Ontology Language Overview*, W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>

[SAM] Golebiowska J, 'SAMOVAR - Setting up and Exploitation of Ontologies for Capitalising on Vehicle Project Knowledge', In Aussenac-Gilles N., Bibow B., Szulman S., eds, *Proceedings of EKAW'2000 Workshop on Ontologies and Texts*, Juan-les-Pins, October 2000, pp 79-90