Lessons Learned Developing an Ontology about Aeroengines

Victoria Uren[1], Fabio Ciravegna[1], Enrico Motta[2] and Colin Cadas[3]

Technical Report CS-11-02

[1]Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, Sheffield, UK

[2]Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, UK

[3]Rolls-Royce plc, PO Box 31, Moor Lane, Derby, DE24 8BJ, UK

v.uren@dcs.shef.ac.uk, f.ciravegna@dcs.shef.ac.uk, e.motta@open.ac.uk, colin.cadas@rolls-royce.com.

Abstract

Ontologies have become one of the key technologies underlying industrial Knowledge Management systems. Such an ontology should provide a common language to facilitate knowledge sharing horizontally, between different parts of an organization, but requires sufficient flexibility to support the building of specialist vertical knowledge systems. We describe the development of an ontology for the aerospace domain, including design decisions made and lessons learned. The design of the ontology includes an upper ontology based on modules with very simple knowledge structures. This provides the sharable core upon which local ontology extensions can be built for communities of users with specialist expertise. The ontology has been successfully applied in Knowledge Management systems for two use cases, one dealing with issue resolution and one with experimental vibration.

1. INTRODUCTION

Ontologies are recognized as a solution to some of the challenges of Knowledge Management (KM) because of their ability to support intelligent search [Fensel 1998] and to bring together distributed information resources stored in databases, file systems etc., which typically exist in large organizations [Maedche 2003]. In this paper, we report our experiences of building ontologies for Rolls-Royce, which describe jet engines and associated engineering matters, as part of a project to investigate the use of semantic annotation to support access to corporate knowledge resources stored in different media. This work is informed by a sustained programme of research led by Rolls-Royce, which has investigated the role ontologies can play in knowledge systems in large, technical organizations, e.g. [Crowder 2010] [Dadzie 2009 ESTC] [Fowler 2008].

The ontologies discussed in this paper concern the design, testing and diagnosis of jet engines. Therefore, we will begin with a brief introduction to how jet engines work, and to the two specialist use cases for which we built KM systems in the project. We will then discuss the design process, the ontologies themselves and how ontology design was tailored to the needs of different communities within the organization. We describe some of the knowledge management tools that were built to exploit the ontology. Issues associated with ontology evolution are raised before we conclude by summarising the lessons learned.

1.1 The Jet Engine Domain

The principle on which a jet engine works is simple. Air is taken in at the front through a fan; this is the part with long blades that is visible from the outside of the aircraft. The air is then passed through a compressor, which has a succession of rotors that compress the air. Fuel is injected into the pressurised air and ignited in the combustion chamber. The hot gas expands, forcing its way through a second set of rotors, called the turbine, which is designed to both maximise thrust and drive the compressor. The force of the gas being pushed out of the back of the engine provides the thrust that drives the aircraft forwards. The principle is simple, but the engineering is not, as this combination of operations produces extreme operating environments. For example, temperatures range from -40 deg. C at the fan at high altitude to in excess of 1500 deg. C in the combustion chamber. The extreme conditions produced in jet engines require very sophisticated engineering design and high performance materials. An accessible introduction to the design of jet engines is provided by [Rolls-Royce 2005].

The ontologies were developed initially to provide KM services for two specific use cases. The first use case is called Issue Resolution (IR) and concerns in-service issues. When an issue is reported, for example a component is rather more worn than normal when examined during a regular service, a team will be set up to try to identify the cause of the wear and to recommend a solution. This might be a way to reduce wear or to change the service regime to pick it up sooner [Dadzie 2009 ESWC]. Issue resolution is particularly interesting from a KM viewpoint because of the very wide range of issues that may come up and because knowledge relevant to each issue may be scattered over a very wide range of repositories, and documents produced during the design, manufacturing or during service. The KM challenges of this use case are three fold. Firstly, the issue resolution teams receive issues of all types and therefore the system needs to cope with a very wide range of document types and topics. Secondly, finding relevant data may require lateral thinking, for example an experienced engineer may recognize a phenomenon as one s/he has seen before on a different kind of component that operates in a similar environment. Thirdly, the team needs tools to help them interpret large quantities of diverse information systematically.

The second use case is called Experimental Vibration (EV) and concerns testing of new engine designs for certification before they enter service. The problem addressed is this: a component, such as a blade on a rotor may have one or more natural frequencies (modes) just like a tuning fork. If the engine is run in a way that keeps a component at one of these frequencies then it will be damaged; this reduces the required time between services to replace the affected component. Testing the vibration properties of engines is a complicated business that generates massive amounts of data. To give an idea of the scale of the EV operation, currently the company stores 35,000 tapes of legacy data in offline storage and retains a testing department with 40 staff. It takes a minimum of five years for an EV engineer to gain the experience required to understand and interpret the data from a new test. Therefore, giving access to legacy data, with interpretations of similar phenomena could be very valuable.

These two use cases pose contrasting challenges. IR uses a very wide range of information sources, primarily text and images, which may have been generated at any stage of the product lifecycle, from design onwards, both by external organizations as well as within the company. EV, on the other hand, uses only internally generated data, and is characterized by large amounts of numeric data. Both use cases share the key aspect that an engineer has to pull together evidence from a range of sources to make an informed judgment.

Ontology based KM is valuable for cases such as these, because it provides a framework in which to bring together knowledge that is scattered across different data stores and may exist in different media, with images and data being important, as well as text documents. The ontology is used to annotate the sources, thus facilitating knowledge sharing and retrieval across sources and media.

1.2 The Ontology Design Process

As should now be clear, the ontologies were developed for an industrial company and with particular tasks in mind. This means that the driving forces behind their development were different to those for ontologies built for research only, or for public data sharing. An ontology which was formally elegant, but which failed to meet the company's needs, would have been inadequate. Past experience at the company [Fowler 2008] had shown that getting the level of granularity right in an ontology for one particular application (describing engine components) was a significant challenge that required several design iterations. Furthermore, the ambition of the programme had now expanded to investigate whether ontologies could be used more widely across the organisation than the design application considered in earlier work [Crowder 2010]. For this a flexible ontology which could be reused in a range of contexts was required.

To achieve this, the involvement of stakeholders in the knowledge acquisition and modelling processes was critical. Therefore, the knowledge engineers worked with knowledge management experts and aerospace engineers from Rolls-Royce, who understood the data sources and the knowledge management tasks, but also with the developers who were building software tools to support those tasks. These groups had different needs and views about how the ontology needed to be organized, some of which will be discussed in detail in the section on functional requirements.

The development process is detailed below. This process includes several steps in which design decisions were made that significantly improved the quality of the ontology. We will discuss these decisions in more detail later.

March-July 2007: Design meetings with the "use case owners" (small groups of engineers with deep understanding of the engineering use cases that were to be addressed) and other stakeholders. These defined the topics the ontology would need to cover and identified materials that could be used as references. An initial ontology was produced, which was modified in several rounds of meetings.

August 2007: Delivery of the first full ontology. This first ontology was "monolithic" and contained the features required for both use cases.

September 2007 - September 2008: Parallel development was carried out by the two teams of use case developers, as required to meet the individual requirements of their prototype ontology-based knowledge management systems, resulting in the production of two versions of the ontology, one for IR and one for EV. The development was distributed over several sites in Europe, which meant that central control of the ontology was deemed impractical. The feedback from this step proved very helpful.

May 2008: Merging of the two ontology versions and review of issues arising so far. A modular design was proposed for the next phase. The domain experts identified a need for more clarity.

November-December 2008: Design of the new modular structure based on an upper level schema, proposed by two of the most experienced Rolls-Royce engineers, in which each module was designed to reflect the viewpoint of a substantial user group within the company. This change made the ontology more naturalistic.

January-March 2009: Design of local ontologies for the two use cases using selected modules.

March-August 2009: "In-use" development of the local ontologies for their application

August-November 2009: Evaluation of the ontologies in demonstrators.

Is it reasonable to ask how our process compares with the recommendations from the literature on how to build good ontologies? Like [Uschold 1996], we carried out brainstorming, grouping and identifying terms, but restricted producing definitions of terms to occasions where there was obvious ambiguity. In addition, we inserted a prototyping step. Building ontology based services revealed gaps in the model and allowed us to focus on real, as opposed to perceived, issues. The result of prototyping was a substantial redesign, including changes to the upper level ontology. The prototypes also engaged users, because they could see the potential benefits and this meant that they were far more proactive and committed to ontology building in the second phase. We draw a contrast here with the V-model proposed in [Stevens 2000], which puts implementation at the bottom of the V as the final step. Our development process was closer to evolutionary design, as seen, for example, in the cyclic pattern of human centered design [ISO 13407], and was broadly consistent with the Iterative-Incremental Life Cycle Model presented in the library of life cycle models described in the NeOn Methodology [Suárez-Figueroa 2009] The prototyping produced substantial improvements in the final quality of the ontology.

We will discuss some of the changes we made in the course of the design in the next section.

2. ADAPTATION TO REQUIREMENTS

As outlined above, the design of the ontology evolved through two design cycles to meet the requirements of the stakeholders. We will concentrate on some of the more important decisions, in particular, ones where lessons were learned, which may be illuminating for future ontology designers.

2.1 Modular Design

The first design was rather monolithic. In the second phase, we changed from a monolithic to a modular ontology design. This was initially considered, based on feedback from the developers, who had made "in-use" changes. Even with just two use cases modelled in the ontology, it was hard for the developers to have a clear and complete view of the ontology - sometimes they could not identify relevant existing classes. This resulted in some cases of duplication, where the developers, unable to find what they needed, added classes and relations that existed elsewhere.

This is a common situation. Applications are normally developed collaboratively. In ontology based applications, developers who were not directly involved in the design of the ontology, and who may not regard knowledge modelling as their main role, are involved in reusing ontologies and may need to make minor modifications. This is typical of the semantic web, which has created mass public availability of ontologies for the first time, opening the potential for large scale reuse of ontologies in applications that were not envisaged by their original designers. This has generated the current interest in modular ontologies among semantic web researchers, which has been explored in the NeON project [NeON], and in events, such as the WOMO workshops [WOMO].

Building modular ontologies is considered good practice for the same reasons that modular software is, i.e. they are "easier to understand, verify, debug, extend, reuse parts of, and thus facilitate collaborative development." [Grau 2006].

At the end of the first phase of the ontology development, we realized that the problems that lead to the interest in modular ontologies on the semantic web were also occurring in this more controlled industrial setting. Therefore, it was decided to change to a modular design. The advantages we found in modular design are discussed below.

The modular design is scalable to large ontologies: The monolithic ontology had become large enough to make it difficult to navigate and understand. We anticipated that a full working version would be considerably larger than the version described here, which was built to support the research project X-Media, but is intended to be extensible for companywide use. For example, the part of the ontology dealing with components could potentially become very large indeed, since each engine has approximately 30,000 components and there are dozens of engine classes, each with multiple variants with different builds. Modular structure splits the ontology into more manageable chunks.

The modular design facilitates ownership: The doctrine in KM has been that ontologies are used to provide a shared understanding, which in turn facilitates sharing of knowledge right across the organization. This is a notion that appeals to senior managers. However, in practice, what Benjamins et al. call "horizontal" KM systems [Benjamins 1998], that serve the whole organization, often fail, while "vertical" systems, with a narrow focus, succeed. In our experience, one cause of this is that horizontal systems can make the mistake of "Dictating the use of a particular ontology to people to which [the ontology] they have not contributed" [Benjamins 1998]. With vertical systems, which aim at smaller groups of user, the users are more likely to be involved in system design, and maintenance, making them more engaged with the systems. The risk of vertical systems is they become information silos. The module structure is designed so that the top-level can support horizontal systems if needed, by imposing a corporate view for a few key terms. Vertical systems that can meet specialist needs can be built as extensions, and the upper level becomes a bridge between specialist repositories, hopefully reducing the risk of them becoming silos.

Table 1. The modules of the core ontology

Module Name	R-R Viewpoint: Contents	Metrics
Mechanisms	Scientific: it concerns physical mechanisms that affect engines.	Classes: 4 Properties: 3
Engines	Products: it concerns the naming of different models and builds of engines.	Classes: 7 Properties: 3
Systems	Functional: it concerns the functional systems in the engine.	Classes: 55 Properties: 0
Commodities	Supply: it concerns the naming and numbering of components,	Classes: 59 Properties: 1
Services	Service: it concerns the different levels of commercial consequences resulting from technical issues in engines for customers.	Classes: 3 Properties: 2
Locations	Topological and Geographical: it concerns the locations within the engine and in the world.	Classes: 19 Properties: 0
Events	Time and Mechanical occurrences: it concerns events such as when a service occurred or when an issue was reported.	Classes: 10 Properties: 2
Physical Objects	Tracking: it concerns the tracking of individual instances of components using serial numbers.	Not built (not required for the use cases)
Knowledge Sources	Archival: it concerns knowledge artifacts such as documents, datafiles and images.	Classes: 25 Properties: 30
Agents	People and organizations: it concerns both R-R personnel and units and external agents such as customers and suppliers.	Classes: 10 Properties: 0

As Table 1 shows, the structure of the upper level modules, proposed by the domain experts, is designed in such a way that there is one module for each function within the engineering part of the business (we did not produce modules for sales, finance, or manufacturing, although these could be added). The company is organized in such a way that functions are reflected in the structure of departments. The domain experts' idea was that, if each department had control of the part of the ontology that mattered the most to them, conforming to the other parts of the ontology would be more acceptable. In an organization with a strong quality culture, this structure means that the responsibility for change control (known in the company as "governance") may be allocated to groups with appropriate technical expertise.

This organizationally dictated structure means that the module design was not targeted at addressing the kinds of module quality criteria proposed in [D'Aquin 2009], since the design was, instead, dictated by the organizational structure. This is known to be critical to success, since "[KM] projects that don't fit the culture probably won't thrive" [Davenport 1997] and some estimates suggest that 84% of KM programs fail [Lucier 1997]. A cost may be that the performance improvements that can be obtained from modular ontologies may be reduced, since the modules have been designed for human understanding rather than for processing speed.

Modular design facilitates reuse: Modular design means that existing ontologies, which have been developed in the company, can be reused. Similarly, modules developed for X-Media can potentially be reused in future KM projects. Techniques for modularization include partitioning, which involves sub-dividing a large ontology into a set of units that together are semantically equivalent to the whole, and module extraction, which involves selecting a subunit of an ontology, typically using the traversal approach to gather connected classes and relations. The module extraction approach was suitable for our purposes, because the new, organizationally based, module structure was being applied top-down. Therefore we needed to extract parts of the ontology that were suitable for including in the new modules. For a bottom up design approach, subdividing the monolithic ontology would have been required.

For example, for the EV use case, the domain experts were satisfied that the model produced first time around was pretty close to what they needed. A module containing 30 classes, 17 object properties and 28 datatype properties was extracted from the monolithic ontology for reuse using the NeOn Toolkit [NeON Toolkit].

2.2 Domain Specific Upper Level

We initially used DOLCE UltraLite (DUL) as the upper level ontology [Gangemi 2007]. The domain experts did not see a benefit in abstract DUL classes such as SocialAgent and PhysicalArtifact and were nervous of maintaining, long term, an ontology, the upper level of which, they did not fully understand. This is not intended as a criticism of DUL. As ontology engineers, we find abstraction to be an elegant and well-formed way of looking at the world. Engineers, however, are very focused on practical issues and, in general, do not see philosophical abstractions as relevant to the issues they face on a day-to-day basis.

However, working with DUL had a serendipitous bonus. We had to explain the principles of using an upper ontology to the users, and, although they had concerns about DUL itself, the idea of an upper level, containing very broad and general classes, was understood as a powerful way of organizing the ontology. In our experience with two use cases, a single unifying ontology, with the complexity to address all the necessary technical detail of different tasks, is unwieldy. An upper level ontology that allowed organization-wide sharing of the core knowledge of the organization, while permitting detailed local extensions, was an idea that the domain experts readily took ownership of.

An affordance of modular design that was noticed by the engineers, and convinced them it was the right approach was that, by defining the upper ontology in the right way, they could envisage ways to control ontology changes in the future. There are two main points. The first is that the core ontology should represent the part of the terminology of the company that must be enforced to ensure knowledge sharing and communication across the company. Local extensions of the ontology, for particular applications, can then be tailored to local needs, with only the minimum of top-down control of the most important terms (classes, properties or instances) through the core modules. The second is that, provided the ontology modules reflect the functional structure of the organization, editorial responsibility for the different modules can be assigned to people with appropriate experience. In short, the design approach should lend itself to constructing a maintainable ontology.

As mentioned above, the design of the upper level was expressed as a series of "core" modules, each of which represents a view of a substantial user group within Rolls-Royce. This design was led by two senior engineers, each with the experience of working in several different sections of the company. They were capable of seeing, not just their current local view, but also the pan-organizational view of what knowledge should be shared between engineering functions to improve overall performance. The modules of this core ontology are summarized in Table 1.

2.2.1 *Examples of Core Modules*

Within each core ontology module, we have built an ontology with an appropriate level of semantic richness to be a template for modelling the sub domain. We applied the normal criterion of minimal ontological commitment [Gruber 1995]. However, note that minimal, in this case, really means minimal. As can be seen from Table 1, some of the core modules are very small and some contain only taxonomies (these can be identified as the ones with no properties). First, it should be remembered that these are upper level ontology modules. Several of them need to be assembled to build the upper level for each use case. Second, there are several driving forces for keeping the modules simple. Obviously, the core modules may be reused in ways we cannot predict. They will be used in an industrial context, where only standard ontology tools are available, rather than research environments in which there might be the option of using modular ontology languages, such as those described in [Bao 2006]. Consequently, we can only rely on importing complete ontologies and cannot expect to be able to partially reuse ontologies. Furthermore, it has been our experience that, in KM applications, there are many tasks for which simple taxonomies are sufficiently descriptive for the task. This is because a significant part of the challenge of KM concerns identifying resources about a topic. For this task quite simple structuring (e.g., subclass and part-of relations) give substantial improvements in performance over keyword search. Therefore, in many cases the core ontology simply provides the top few levels of a taxonomy. Taxonomies have the advantage that developers and users have no difficulty in understanding them. This is significant if the company plans for the ontology to be maintained by domain experts and not by ontology engineers.

The Locations module provides an example of simple modelling. It contains classes that are intended to guide the creation of *Location_tag* instances. There are only two subclasses of *Location_tag*: *Assembled_location*, which describes where something is in an engine, *Geographic_location*, which describes a location in the world, such as an airport. These two have some further subclasses to guide people in populating the ontology.

Only taxonomic properties are expressed in the Locations module. We did not express disjoint constraints between the classes. Since the location classes can overlap, making the subclasses disjoint would not be naturalistic, e.g. a location can be both in the compressor (*Compressor_location*) and close to the axis of the engine (*Axial_location*). Creating locations with multiple inheritance may not be wise and it would be possible for a knowledge engineer to add disjoints, if the application required them. By not putting disjoints in this core ontology, we do not impose an artificial constraint on the behaviour of local extensions. Figure 1 shows the class structure of the module, and an example owl file is provided as an appendix.

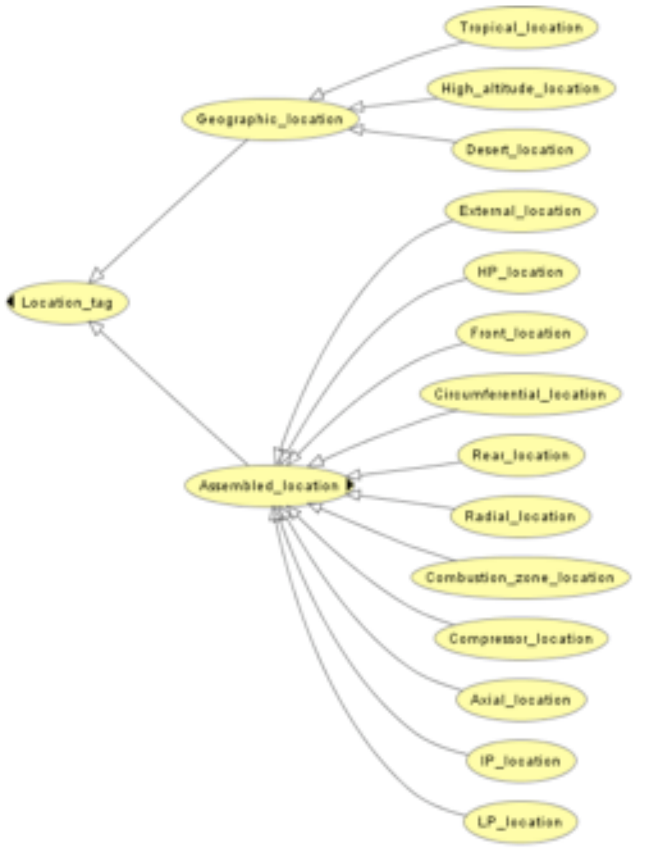

Figure 1. Upper level classes and a selection of subclasses from the locations module of the ontology

Not all the core ontologies are taxonomies, but most have only a handful of classes. The aim is to express, in the core ontologies, basic concepts that could be shared with the whole organization.

Another example is the Engine class. The engine naming system is tightly followed in the company. Modifying this part of the ontology would not be permitted and the Engine module enforces its use in the ontology by putting it in a top-level module.

There are three main components in an engine name, which are illustrated in Figure 2. The *Engine_family* describes a group of engines with related designs, where typically the family name is taken from a British river, such as Trent or Spey. The *Engine_type* refers to the generation of the design in the family, for example the Trent family includes the *T600*, *T800* and *T900* types. The *Engine_mark* identifies more precisely modifications within an engine type. These components of the name are modelled as subclasses of Engine property. An Engine instance can be constructed using the *has_engine_property* relation. An example Engine instance is provided in Table 2, and the appendix contains the core module plus the instance.

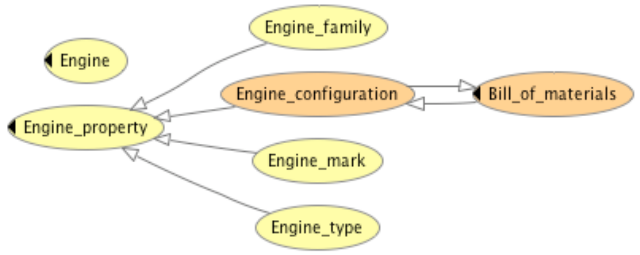

Figure 2. Classes of the engine module

Figure 2 also shows the Engine property, *Engine_configuration*, which is equivalent to *Bill_of_materials*. This is a list of *Engine_components* and may optionally be used to provide a definitive description of an Engine instance. The equivalence relationship acknowledges two alternative vocabularies in use in the company, which need to appear in different local extensions. Such differences in terminology are typical for large organizations and we will discuss them further in the next section.

Table 2. Example of an Engine instance

```
<Engine rdf:ID="Engine_Example">
   <has_engine_property>
     <Bill_of_materials rdf:ID="Bill_of_materials_4"/>
   </has_engine_property>
   <has_engine_property>
     <Engine_type rdf:ID="T800"/>
   </has_engine_property>
   <has_engine_property>
     <Engine_mark rdf:ID="Mark_892"/>
   </has_engine_property>
   <has_engine_property>
     <Engine_family rdf:ID="Trent"/>
   </has_engine_property>
   <has_engine_ID
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >ID111111111</has_engine_ID>
  </Engine>
```

The modular design of the new upper level addresses the fact that not all applications will need to address all the views. As such, it is a naturalistic representation of how knowledge is structured in the company, where different applications have different needs. This makes it suitable for supporting company wide knowledge sharing. In Gruber's terms [Gruber 1995], we claim that that the ontology is extensible, as it offers "a conceptual foundation for a range of anticipated tasks", i.e., for KM tasks for different user communities. In the next section, we will discuss the local extensions, which add whatever additional complexity is required to support specific tasks.

2.3 Communities of Practice

The viewpoint of a user often changes the kind of information they seek when faced with the same problem. Take the example of a particular model of car, which customers keep bringing into garages with a wet carpet. The service department of the company needs to know where the car is leaking in order to identify which seals should be replaced to repair the cars. The manufacturing department needs to know the process steps used to assemble the model to determine whether the leak is caused in the factory and can be remedied for new cars. The design department needs to know about the materials used to make the seals in case a change in material can solve the problem in the next model. One problem "wet carpet" thus needs different types of knowledge to solve it.

This phenomenon is widespread in technical organisations where there are experts with many different specialisms and different conceptualisations of the data they share. One aspect of this is the use of different terminologies by different communities. This work supports the case where, say, a maintenance engineer knows a component by one name, his colleague in design calls it something else and the official name is different again.

We find the notion of communities of practice helpful in understanding this phenomenon. Communities of practice comprise informal groups of knowledge workers with shared conceptual views of the world (i.e., with local knowledge), procedures, operating environments, seminal texts etc. [Wenger 1999]. With their joint emphasis on conceptualization and practices, communities of practice are compelling places to target ontology-based KM systems, since, if the ontology and the tools address the community's needs and preferred working methods, then they are far more likely to be successful. However, as we have discussed above, organizational level knowledge sharing requires uniformity.

One solution is to address the problem of multiple sublanguages, which occurs within Rolls-Royce, as in many other organizations, by supporting synonyms for terms. Methods for terminology recognition are therefore being investigated [Butters 2008] as means to populate ontologies with synonyms. An ontology with rich synonym support is clearly a powerful resource for KM. Uses would include query extension to include synonyms and the provision of specialist thesauri to help experts understand documents produced by staff with other specialities.

In building the ontology discussed here, we have tested whether, using the ontology design we proposed, it is possible to resolve this potential conflict between the organizational drive for uniformity and the need to provide particular communities of practice with KM tools that serve their specific needs. The two communities of engineers involved in EV and IR both needed to access large-scale data across text, images and raw data (numbers).

As we stated earlier, our solution is to use the core ontology modules we have described above to build upper level ontologies on which to base detailed local extensions. An ontology designer begins by selecting the core modules required for their application, for example the IR use case, which has potentially a very wide remit, needs all the modules, whereas EV, with its more specialist focus, needs only a subset including Engines, Mechanism, Commodities and Knowledge Sources. The core ontology for the IR use case is illustrated in Figure 3. The next step is to identify areas where the list of classes and properties need to be extended. For example, for the IR teams, a pre-existing list of damage mechanisms was added to the Mechanisms module to support the root cause analysis task, whereas the EV engineers needed a substantial subontology of vibration specific classes and properties in Mechanisms. Neither community could benefit from the extension that was vital to achieve the goals of the other.

2.3.1 Local Extensions

We present an illustrative example based on the local extension for the IR use case. This local extension may be used, for instance, to allow engineers to search for all past events that occurred in the relevant region of a particular kind of engine, without having to identify every component in the region. Figure 4 illustrates the instantiation of classes from three different core modules: Events (the instance *Burner_seal_event_5*), Commodities (the instance *Burner_seal_1*) and Locations (the instance *Combustion_zone_location*, which has an identical label to its parent class in this case). *Combustion_zone_location* is used to model the location in the engine of the component (*Burner_seal_1*) using the object property *has_position*. The *Burner_seal_event_5* is linked to the component using the *has_Applicability* property. The range of this property includes other classes such as Engine, which are not illustrated here, allowing for a rich semantic description of individual events, but with less complexity compared to having a specific property for each class that needs to be associated to an event.

For the KM tasks that these local extensions serve, we have found that the use of a relatively large number of datatype properties related to relatively few object classes can work well as a means to describe complex objects in KM systems. There were two main drivers for this. The first was the need for some text to supplement the semantic description for the human users. Figure 4 provides several examples of the use of text datatype properties to provide a human readable summary of the event represented by the instance Burner_seal_event_5: for example, "Inner surface of the seal is discoloured". These text descriptions are flexible. When knowledge is created, the engineers can express whatever they need to in them free from the constraints of the ontology. Later, they can give reusers of the knowledge a quick summary of the event. Furthermore, user acceptance of the KM system is enhanced by features, such as adding text annotations that are familiar from normal working practices. The second case arose primarily in the EV use case, where much of the knowledge that needs to be stored is numeric. Numeric datatypes are then required so that this data can, for example, be accurately visualized or used in computation.

The local extensions of the core ontology have been incorporated into a number of KM demonstrators, which have been designed and evaluated with the active participation of their respective user communities. These applications include: search [Bhagdev 2008 ESWC], acquisition [Bhagdev 2008 ISWC], visualization [Petrelli 2009] and sensemaking [Dadzie 2009 ESTC] for the IR use case, and computational tools for recognizing vibrational modes for the EV use case [Buza 2009]. All these are described in detail in the referenced papers, but we give a flavour of their functionality below to illustrate the ways in which using the ontology benefits knowledge management.

K-search [Bhagdev 2008 ESWC] uses a hybrid approach to search, which mixes formal ontology searches with keyword search. This approach gives the best of both worlds: searching the ontology gives high precision, and keyword search plugs the inevitable gaps where ontology terms do not exist, e.g., for rare and novel terms. The tool also allows users to search in multiple repositories across the organization, which is essential for the issue resolution use case in which knowledge required to diagnose or fix an issue may need to be sourced from the records of any of the departments that work on an engine, from initial design through testing to maintenance and in-service support. Searching different sources through a single front end gives easy access to information which might otherwise end up in "silos", while acknowledging that real organizations have good reasons for having multiple data stores. Different data stores may use different local versions of the ontology but, because they share the same upper level modules, knowledge relevant to other parts of the organizations is identifiable.

The K-Forms acquisition tool [Bhagdev 2008 ISWC] supports communities in extending existing ontologies to meet their dynamically changing needs as part of their normal work. This is achieved using forms where the form schema is directly linked to the ontology; filling forms is an integral part of knowledge generation in Rolls-Royce where it has the role of assuring that knowledge about tasks is collected consistently across the organization. Ontologies are populated by filling fields in forms and can be extended by creating new fields. This eases ontology building for domain experts compared to learning to use an ontology editor.

Semantic data for a collection of knowledge can be viewed using familiar visual metaphors, such as timelines and maps [Petrelli 2009] but also with domain specific ones, including plotting locations over a line drawing of an engine. The plots provide gestalt overviews of the data allowing engineers to quickly notice, that for example, many of the reports of a particular issue are from engines being flown out of airports in a particular region of the world. They also provide a browsing tool because each point on the visualisation, representing an instance in the ontology, can be clicked to open the original document.

To support sensemaking, the tools are all brought together in a graphical interface called the XMediaBox [Dadzie 2009 ESTC]. This provides a shared workspace for a team working on a knowledge task. Tools, such as K-search and the visualization tools, can be accessed through the XMediaBox and pertinent items found by team members can be semantically annotated and placed in a shared store. This environment allows teams to work asynchronously on a problem, and share their discoveries and hypotheses.

Some of the tools built for the project were highly specialised. For example, in the EV use case, one of the document types vital to the analysis are the Campbell diagrams, which plot the vibration response of an engine under test. Specialist tools needed to be built to identify, and semantically annotate, lines on these plots which are important for the analysis of the engine's reliability [Buza 2009].

2.4 Ontology Evolution

The need to support evolving ontologies has been recognized for some time [Maedche 2003]. Nonetheless, the question of ontology evolution has been one of the hardest we have tackled on the project and it is not yet fully resolved. Until recently, most ontology design tools have assumed a single editor; this does not match the situation in large organizations, where multiple users may need to modify the ontology. This has begun to be tackled, with systems being produced that aim to support distributed teams of ontology engineers, e.g. [Palma 2008]. Such systems will tackle hard technical challenges, such as the detection of inconsistency and redundancy.

The paucity of ontology tools suitable for domain experts remains problematic. As ontologies become an increasingly mainstream technology, more and more ontology editing, particularly in the evolution stage, will be done directly by domain experts, usually as an additional task, not as their main role. In building this ontology, we have worked with engineers, who typically have very high levels of computing skill, particularly with numerical software. Nonetheless, they are not comfortable with common ontology editing tools. The ontology resources of the company will be maintained by domain experts, such as these, not by knowledge engineers. Therefore, as pointed out in [Dzbor et al., 2006], a key research issue concerns improving the user interface design of ontology editing tools to make them more suitable for domain expert use. This is a technological issue that is being addressed, for example by tools such as PCPACK5 [Epistemics], which are intended to support knowledge elicitation in organization.

Another issue, which is more organizational than technological, concerns change control. Here we have another possible source of conflict between local and corporate interests. At the local level, engineers are working on specific tasks for which they need to be free to create new entities. For example, in the IR use case, the goal is to identify the root cause of an issue. One of the tools we have designed for the use case is a tree visualisation, which the user can populate with potential root causes and then gather evidence for and against different hypotheses [Dadzie 2009 ESTC]. Root causes can be selected from the ontology and new causes can be added. Most issues are caused by a relatively small number of known physical mechanisms, such as various types of corrosion or wear. However, it is best practice during IR to consider all possible root causes; constraining engineers to only pick from a fixed list during hypothesis investigation could lead to suboptimal solutions. Therefore, they need to be able to freely express additional hypotheses. At the corporate level, however, it does not help knowledge sharing to allow the ontology to become bloated with new root causes most of which will be variants of existing classes. For corporate knowledge sharing, tough control of core classes is needed.

Our proposed solution would involve appointing human "gatekeepers" to validate concepts before incorporating them fully into the core ontology and sharing them. Unvalidated concepts would be available, but flagged to alert users to their status. The modular structure has benefits here. Since the modules were designed to reflect the views of key user communities across the company, appropriately qualified people could be identified to oversee changes in different modules.

To maintain consistency, these "gatekeepers" need a clearly defined policy for change control balancing the need for consistency with limited resources for maintenance. Changes can affect any part of the ontology, but one very common kind of change is the extension of classes with new instances. A change control policy would need to identify what level of validation to apply to specific classes. We have identified three levels of control.

Hard control: these are instances of classes that are crucial for corporate knowledge sharing. A typical example is product names, which must be identical in all local ontologies. Users should be blocked from generating new variants of these classes and a request to a gatekeeper would be required. To enforce hard control on a class, its boundaries need to be very clearly defined and its instances should be stable; if they change frequently the inconvenience to gatekeepers could cause the change control process to be abandoned.

Soft control: these are instances of classes where flexibility is needed at the local level, but there is good reason to control the number of instances generated in the shared repository. The Root_cause class in the IR use case, discussed above, is an example of this.

No control: this last class of instances are not contentious and are expected to proliferate. Dates are a typical example. Mechanical checking of their formatting is possible and new ones will constantly be being generated.

2.5 Lessons Learned

We have described above a few of the key design decisions made during the design of an ontology for the aerospace domain. In conclusion, we will summarise some of the lessons we learned in the process that may be of help to others.

A key turning point in the development of this ontology occurred when senior engineers took ownership of the notion of the core ontology and the design of the upper level module structure. In their conclusions concerning lessons learned on an earlier Rolls-Royce ontology project Fowler et al [Fowler 2008] report that *"difficulties were encountered in persuading other partners in the value of ontologies"*. We encountered this also at earlier stages, but after this point it was largely overcome, the value of ontologies was no longer in question once the broad structure was in alignment with the engineers' own way of seeing their world.

The choice of upper level ontology set the ontology on a very different course to that suggested in [Borgo 2009], where a formal extension of DOLCE for engineering applications is proposed. However, we note their statement that their *"formalizations have not yet been implemented as engineering tools for facilitating the sharing and exchanging of descriptions of the behavior and functions of artifacts, and information about their practical use is still required."* [Borgo 2009]. This is in marked contrast to our ontology: it was our experience of designing and building the prototypes that led us to adopt a less formal approach. Perhaps because we are working with KM applications, we focus on relatively lightweight reasoning. As Jim Hendler has said, "a little semantics goes a long way" [Hendler 2003]. Other applications may require a more formal approach to the one we have adopted here. The lesson for ontology designers is that although theory can teach us very important things, for example that upper level ontologies are useful, we must engage with domain experts and build realistic applications to discover the right way to apply the theory for particular cases.

Another key lesson was that, even with just two use cases in a single organization, we were not able to produce a one-size-fits-all solution. Although corporate knowledge sharing is an important ambition of ontology based knowledge management, it is not practical to try to share the most detailed knowledge of every specialist in the company. Ontology-based KM is most effective when it is tailored to fit the requirements of very specific user groups. These groups often form communities of practice, which are distinguished by having their own sublanguages and practices. The conflict between the need for corporate knowledge sharing and tools that can support local practices is one that we recognized in this project, and we predict it will be common in large, technically complex, organizations. The solution we have developed here involves building a core ontology (in our case it is modular) that models the knowledge that should be shared with the whole organization. Specialist needs are then served by allowing local extensions to be built on top of the core. So long as local extensions stick to the core format, both local and corporate interests can be served. We believe this *sharable core with local extensions* approach may be generally useful for KM applications in complex organizations.

Finally, Fowler et al [Fowler 2008] have previously noted the advantages of taking an "early prototype" approach to building ontologies. We concur with this and also with their comment that the approach has *"the disadvantage that the scope of the ontologies was not fully understood while the ontologies were being built"*, which was exacerbated in our case by the involvement of several groups of developers at different locations evolving ontologies for two different applications. In a deployed situation the case would be even more complex, since many end users will potentially be adding to the ontology simultaneously and such tools as those presented in [Palma 2008] will increasingly become key technologies for ontology developers. At present, we consider this problem of ontology evolution with multiple editors to be an open issue that requires further research.

3. ACKNOWLEDGMENTS

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

4. REFERENCES

[Bao 2006] Jie Bao and Vasant G. Honavar, Divide and Conquer Semantic Web with Modular Ontologies - A Brief Review of Modular Ontology Language Formalisms, Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06.

[Benjamins 1998] V. Richard Benjamins, Dieter Fensel, Asunción Gómez-Pérez: Knowledge Management through Ontologies. PAKM 1998.

[Bhagdev 2008 ISWC] Ravish Bhagdev, Ajay Chakravarthy, Sam Chapman, Fabio Ciravegna and Vita Lanfranchi. Creating and Using Organisational Semantic Webs in Large Networked Organisations, Proceedings of the 7th International Semantic Web Conference (ISWC 2008), Karlsruhe, Germany.

[Bhagdev 2008 ESWC] Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi and Daniela Petrelli. Hybrid Search: Effectively Combining Keywords and Ontology-based Searches In Proceedings of the 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain.

[Borgo 2009] S. Borgo, M. Carrara, P. Garbacz and P.E. Vermaas, A formal ontological perspective on the behaviors and functions of technical artifacts, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2009), 23, 3–21.

[Butters 2008] J. Butters and F. Ciravegna, Using Similarity Metrics for Terminology Recognition, In Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), 2008.

[Buza 2009] Krisztian Buza, Christine Preisach, Andre Busche, Lars Schmidt-Thieme, Wye Houn Leong, Mark Walters, Eigenmode Identification in Campbell Diagrams, in International Workshop on Machine Learning for Aerospace 2009.

[Crowder 2010] Richard Crowder, David Fowler, Quentin Reul, Derek Sleeman, Nigel Shadbolt and Gary Wills, An information system to support the engineering designer, Journal of Intelligent Manufacturing, published online September 2010.

[D'Aquin 2009] d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M., (2009) Criteria and Evaluation for Ontology Modularization Technique. In Heiner Stuckenschmidt, Christine Parent, Stefano Spaccapietra (ed) Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, Criteria and Evaluation for Ontology Modularization Technique.

[Dadzie 2009 ESWC] Aba-Sah Dadzie, José Iria, Daniela Petrelli, Lei Xia, (2009) The XMediaBox: Sensemaking through the Use of Knowledge Lenses, European Semantic Web Conference, LNCS 5554, pp.811-815.

[Dadzie 2009 ESTC] Dadzie, A & Ciravegna, F. (2009). Flexible Knowledge Generation and Re-Use in Aerospace Engineering using the XMediaBox. At the European Semantic Technology Conference (ESTC 2009) - Session: Semantics in the Enterprise.

[Davenport 1997] T. Daveport, D. DeLong and M. Beers, Building Successful Knowledge Management Projects, Sloan Management Review, Winter 1998.

[Dzbor 2006] Martin Dzbor, Enrico Motta, Carlos Buil, Jose Manuel Gomez, Olaf Goerlitz, and Holger Lewen (2006). Developing ontologies in OWL: An observational study. In: OWL: Experiences and Directions 2006, 10-11 Nov 2006, Athens, Georgia, USA.

[Epistemics] http://www.epistemics.co.uk/

[Fensel 1998] Dieter Fensel (1998) Ontologies: a silver bullet for knowledge management and electronic commerce, Springer Verlag.

[Fowler 2008] David W. Fowler, Quentin Reul, Derek Sleeman. (2008) IPAS ontology development Stefano Borgo, Leonardo Lesmo (ed), Proceedings of the 3rd International Workshop on Formal Ontology Meet Industry Workshop (FOMI 2008) (Torino, Italy): pages 120-131.

[Gangemi 2007] Aldo Gangemi (2007) DOLCE UltraLite OWL Ontology, http://www.loa-cnr.it/ontologies/DUL.owl .

[Gruber 1995] Thomas R. Gruber (1995) Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43(5-6), pp.907 – 928.

[Grau 2006] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006), 2006.

[Hendler 2003] On Beyond Ontology, Keynote talk, International Semantic Web Conference 2003.

[ISO 13407] ISO 13407:1999 Human-centred design processes for interactive systems.

[Lucier 1997] Lucier, C. and Torsiliera, J. (1997), Why knowledge programs fail, Strategy and Business, 4th quarter, pp. 14-28.

[Maedche 2003] Alexander Maedche, Boris Motik, Ljiljana Stojanovic, Rudi Studer, Ralph Volz (2003) Ontologies for enterprise knowledge management, IEEE Intelligent Systems, March/April 2003, pp.26-33.

[NeON] NeOn project website, http://www.neon-project.org/web-content/

[NeON toolkit] http://neon-toolkit.org/wiki/Main_Page

[Palma 2008] Palma, R., Haase, P., Corcho, O., Gómez-Pérez, A., Ji, Q. (2008) An editorial workflow approach for collaborative ontology development. In: ASWC 2008: Proc., 3rd Asian Semantic Web Conference, pp.227-241.

[Petrelli 2009] D. Petrelli, S. Mazumdar, A-S. Dadzie, F. Ciravegna, Multivisualization and Dynamic Query for Effective Exploration of Semantic Data, In proceedings: ISWC 2009 International Semantic Web Conference, 25-29 October 2009, pp.505-520.

[Rolls-Royce 2005] Rolls-Royce plc, The Jet Engine Book, ISBN 0 902121 2 35, 2005, Key Publishing Ltd.

[Stevens 2000] Robert Stevens, Carole A. Goble and Sean Bechhofer. Ontology-based knowledge representation for bioinformatics, Briefings in Bioinformatics, v.1, n.4, pp. 398-414, 2000.

[Suárez-Figueroa 2009] Mari Carmen Suárez-Figueroa, Mariano Fernández-López, Asunción Gómez-Pérez, Klaas Dellschaft, Holger Lewen and Martin Dzbor. Revision and Extension of the NeOn Development Process and Ontology Life Cycle. NeOn Project Deliverable, D5.3.2. 2009. http://www.neon-project.org/web-content/images/Publications/neon_2009_d532.pdf

[Uschold 1996] Mike Uschold and Micheal Gruniger, Ontologies: principle, methods and applications, Knowledge Engineering Review, v.11, n.2, pp.93-136 1996.

[Wenger 1999] Etienne Wenger, Communities of practice: learning, meaning, and identity, Cambridge University Press, 1999.

[WOMO] 1st Workshop on Modular Ontologies: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-232/; 2nd: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-315/.

APPENDIX: OWL FILES

The owl files in this appendix are not the actual owl files of the Rolls-Royce ontology. The ontology itself is the intellectual property of the company and is confidential. However, the examples presented here use the same structures as the modules of the actual ontology and are used in similar ways.

MINIlocations.owl

This example is a cut down version of the RRlocations file with fewer classes.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns="http://www.x-media-project.org/RRmod/locations.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.x-media-project.org/RRmod/locations.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="External_location">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Combustion_zone_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="High_altitude_location">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Geographic_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="HP_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Axial_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Tropical_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Geographic_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Desert_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Geographic_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Location_tag"/>
  <owl:Class rdf:about="#Geographic_location">
    <rdfs:subClassOf rdf:resource="#Location_tag"/>
  </owl:Class>
  <owl:Class rdf:ID="LP_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="IP_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Upper_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Rear_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Turbine_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Front_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Circumferential_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Compressor_location">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Assembled_location"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Assembled_location">
    <rdfs:subClassOf rdf:resource="#Location_tag"/>
  </owl:Class>
  <owl:Class rdf:ID="Radial_location">
    <rdfs:subClassOf rdf:resource="#Assembled_location"/>
  </owl:Class>
</rdf:RDF>
```

RRengines

This file is a version of the RRengines class with a single instance.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns="http://www.x-media-project.org/RRMod/engines#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.x-media-project.org/RRMod/engines">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Engine_family">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Engine_property"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Engine_configuration">
    <owl:equivalentClass>
      <owl:Class rdf:ID="Bill_of_materials"/>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Engine_property"/>
  </owl:Class>
  <owl:Class rdf:ID="Engine_type">
    <rdfs:subClassOf rdf:resource="#Engine_property"/>
  </owl:Class>
  <owl:Class rdf:ID="Engine_mark">
    <rdfs:subClassOf rdf:resource="#Engine_property"/>
  </owl:Class>
  <owl:Class rdf:ID="Engine"/>
  <owl:Class rdf:about="#Bill_of_materials">
    <owl:equivalentClass rdf:resource="#Engine_configuration"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Instances of this class should refer_to an instance of one engine build and configuration. Instances from the Commodities ontology can
be linked to it using part_of properties. The result should associate a list of commodities with an Engine.</rdfs:comment>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="has_engine_property">
    <rdfs:domain rdf:resource="#Engine"/>
    <rdfs:range rdf:resource="#Engine_property"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="has_engine_ID">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Engine"/>
  </owl:DatatypeProperty>
  <Engine rdf:ID="Engine_Example">
    <has_engine_property>
      <Bill_of_materials rdf:ID="Bill_of_materials_4"/>
    </has_engine_property>
    <has_engine_property>
      <Engine_type rdf:ID="T800"/>
    </has_engine_property>
    <has_engine_property>
      <Engine_mark rdf:ID="Mark_892"/>
    </has_engine_property>
    <has_engine_property>
      <Engine_family rdf:ID="Trent"/>
    </has_engine_property>
    <has_engine_ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ID111111111</has_engine_ID>
  </Engine>
</rdf:RDF>
```