Semantic Web Technologies in Technical Automotive Documentation

François-Paul Servant

Renault, EQV NOV 3 31, 13 av Paul Langevin, 92359 Le Plessis Robinson Cedex, France
francois-paul.servant [at] renault.com

Abstract. RDF is the format of choice to exchange data between software components of a corporate system. That's why we decided to use it in a recent work at Renault, in the field of technical documentation. The prototype of a new repository for repair and diagnostic information was modeled with OWL. REST web services using RDF as data format were built on this repository, to provide access to improved repair and diagnostics methods. This report describes how Semantic Web technologies were used. It is not about the "grand vision" of the Semantic Web, but it could be of interest for those eager to promote the use of these technologies in the enterprise.

Keywords: RDF in the enterprise, RDF based web services, automotive diagnostic

Introduction

Integrating the disparate applications and data sources in a large corporation is expensive, and using Semantic Web technologies could dramatically cut down these costs: this is the "*Business Model for the Semantic Web*" [1]. The lowest layer of SW specifications, RDF - an open and mature standard – has some very appealing properties in a corporate context [2]. RDF can indeed be used as a "semantic bus" allowing an easy sharing of information between systems.

Using RDF was therefore a natural choice for the author when he undertook to develop a rather complex prototype in the field of repair and diagnostic documentation: that is a field where many classes of business objects have to be shared among the many different systems that are involved (the documentation process is corporation-wide, spanning from the engineering department to the repair shops).

As these technologies are still largely ignored, this project was the opportunity the author had been waiting for to demonstrate how Renault could benefit from their use.

In this paper which briefly describes this work, we'll give you a quick overview of the project and its objectives, before we discuss some noticeable points.

Objectives and components of the prototype

During the first half of 2006, the author was busy working on a probabilistic induction tool, conceptually not very different from Bayesian Networks, and aimed at diagnostic, using a technology developed by the team he belongs to [3]. The goal was to show that, instead of having diagnostic procedures written in manuals, a "diagnostic engine" could compute them on the fly. Using a heuristic for the "test sequencing problem", it could do so in a way that minimizes the cost of diagnostic.

However, we found that current information systems were not able to provide the data that is needed. Part of it is simply not available at the required granularity, except maybe as unstructured text. Moreover, the systems do not provide API calls that would allow a program to ask questions such as "what are the preliminary steps (e.g. parts removals) to test the resistor of the air conditioning engine on a Renault Clio?" In fact, current systems are geared toward the management of documents about repair and diagnostic, and not toward the management of the corresponding concepts themselves: strangely enough, they lacks a repository of repair and diagnostic operations. Several systems talk about repairs without formally identifying them; they are thus unable to exchange information about them.

We got convinced that such a repository would have a great value for the whole IS. Our goal became then to build the prototype of such a repository, and to prove its value through the demonstration of the diagnostic engine.

That's what we did, using OWL to model the repository, and RDF for information exchange. We provided the so much needed services architecture as a REST one over HTTP, the services returning RDF or HTML depending on content negotiation. A user of the test web application can progressively discover the information about repair, navigating the repository in the same way as she would in a traditional set of web pages. In a diagnostic session, she is instructed by the application to perform tests and enter their results.

Modeling the repository with OWL

A passing remark about corporate repositories, that can be used by people eager to proselytize for SW: value of repositories is recognized, but it is sometimes only partly understood. People focus on their authoritative side, and tend to forget that their primary role is, in fact, to identify their elements, in order to be able to speak about them. We have found this a good starting point to explain the importance of URIs.

Our repository is supposed to identify and describe all the repair and diagnostic operations. An important aspect of it is the "dismantling graph" which models, for instance, that you have to take apart the gearbox if you want to take apart the engine. This graph is used extensively to compute complete operations from elementary ones. For instance, it is used to evaluate precisely the cost of a diagnostic test, which depends on the "dismantling state" of the car.

There is no space here to explain in details what was done, and we will just note one point which is related to the extensive use of Boolean expressions at Renault.

Boolean expressions are used when you want to say: “this document applies to all Renault Megane 2006 equipped with diesel engine and air conditioning”, or “this test result excludes failure of that part”. Our reasoning tools, which are based on Boolean and probability constraints compilation, are a matter of propositional calculus: their first feature is to solve Constraint Satisfaction Problems. Boolean constraints are also closely related to the “Product Diversity Specification” (cf. ISO norm: STEP-AP214, ISO 10303-214:2001), a question which is very important for the repository, whose exhaustivity and non-duality must be controlled: for any vehicle, there must be exactly one way to, say, dismantle the gearbox.

So, as we have our own reasoning tools, we were not concerned with restricting ourselves to OWL-DL. But we had Boolean expressions everywhere, and we faced a dilemma regarding their implementation. Trying to use OWL set operators seemed a bit unnatural, because we’re so used to the classical Boolean operators (NOT, AND, OR, IMPLY...). We ended up creating a BooleanExpression class (with NOT, AND, … subclasses). This turned out to be quite effective in defining “conditional links”, used to say such things as: “the time to take apart the gearbox is 60 minutes if it is an automatic one”. It also made it easy to define the constraints between tests results and failure causes. And it was used in representing conditional probabilities.

A REST web services architecture based on RDF

So, we promoted repair and diagnostic procedures to the rank of first-class objects identified by URIs. Dereferencing those URI returns information about them, in RDF or HTML, depending on content negotiation. We implemented the guidelines concerning URIs that identify concepts, as defined in the solution of “HTTP-range 14” question [4]. This yields a REST web services architecture. Although in the current web application, HTML pages are generated by the server, it would be possible to use a generic RDF browser to connect to the service and get, almost, the same result. The only missing feature of a RDF browser such as Tabulator is the handling of forms. Forms cannot be avoided. For instance, the service may have to ask for technical characteristics of the car being repaired, in order to filter accordingly the list of documents to be returned. Forms are also needed for diagnostic tests. That’s why the question of a standard RDF vocabulary for forms was raised among the community concerned by the linking of data on the Semantic Web [5]. Forms are an important feature of the web of documents: there is no reason it should not be the same of the web of data.

Anyway, the architecture we have demonstrated here is in phase with the specification defined by OASIS Automotive Repair TC (cf. EEC Directive 70/220/ EEC) [6]. We nevertheless propose a slight improvement to the protocol described, that would allow for easier and better handling of the set of cars to which a given document applies: it is to systematically include the VIN of the car being repaired in request for documents. (VIN stands for “Vehicle Identification Number” – kind of URI for cars). In this way the service can fully filter the returned list to match the characteristics of the car being repaired.

Conclusion and future work

Despite their potential, Semantic Web technologies are still largely ignored in the enterprise: very often overlooked, or mistaken for text-analysis techniques by an amusing misunderstanding of the term “semantic”. Not being advertised by solution providers, they are at best considered as “promising”, but not ready to be used right now. They may even suffer from a negative prejudice, being perceived as yet another technological hype, whose promises, such as the easy exchange of information, have already been heard many times before. Also, current focus is about “Web Services”, and people do not understand what SW technologies add to the picture. In this context, people aware of the potential of semantic web technologies face a difficult challenge.

We got the opportunity to highlight some of the benefits of RDF and OWL. Some will shine even more brilliantly in the future: the benefit of a flexible information bus for data exchange fully manifests itself in the number of connected applications.

Thanks to RDF clear and powerful data model, which gives a high level of a priori understanding, we were able to build services that do not require client applications to include code dedicated to their use. OWL allowed to put a lot of business knowledge into data. Better than have it hard-coded in software programs! We found it to be a very effective way to describe the I/O of a system.

Open standards come with robust free tools. Protégé proved to be a very useful one during the work of conception, and provided us instantly with an input tool to key in instance data. We found it easy, powerful and pleasant to use the Jena API. The “Dynamic Object Model” pattern seemed natural and allowed for quick development. Despite our lack of experience working with OWL, we were able to solve the problems we faced in a pragmatic way.

Feed-back inside Renault was positive. Our goal now is to prove that Renault organization will be able to produce the data. We’re very confident about that and plan to connect to the actual sources of information in the engineering department. We have no doubt that Semantic Web formats, and the capacity to easily provide an RDF view of a database, will be useful in this task.

References

1. Berners-Lee, T., “Business Model for the Semantic Web", 2001
   http://www.w3.org/DesignIssues/Business
2. Feigenbaum, L., “Semantic Web Technologies in the Enterprise”, 2006
   http://www.thefigtrees.net/lee/blog/2006/11/semantic_web_technologies_in_t.html
3. Pargamin, B., “Vehicle Sales Configuration: the Cluster Tree Approach”. ECAI 2002 Configuration Workshop (2002) 35–40
4. Fielding, R., “[httpRange-14] Resolved”, 2005
   http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html
5. Community project, Bizer, B., Cyganiak, R., “Linking Open Data on the Semantic Web”
   http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
6. OASIS Automotive Repair Information TC
   http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=autorepair