Semantic Web in the Automotive Industry: a case study

Domenico MACRINI, Aldo MAGGIORE, Antonino SANTAGATA, *Elasis S.C.p.A.*[1]

Abstract. This paper describes an ongoing case study on management of engineering data to evaluate the use of Semantic Web technologies in automotive industry. This case study explores specifically the theme of data modeling, navigation and retrieval using Semantic Web, data presentation in html+svg and data analysis using neural networks. It implements a web crawler to automatically collect data of interest for modeling, too.

INTRODUCTION

Semantic Web technologies promise to allow a step forward to data management capabilities of ICT technologies and satisfy demands previously ignored or only partially answered, especially regarding engineering data and procedural knowledge.

Up to now, in fact, in these fields the most handily solution are offered by RDBMS based systems. The requirement of its object model (tables + relations) to specify the domain predefining the data structure prevents the management of data resulting from engineering activities, such as the one used by the automotive industry. As a matter of fact, those data are recorded in files and documents of several different types, including design documentation, manufacturing documentation, and test documentation. Each set has usually a hierarchical structure, but structures differ between the sets. There is also a set of implied axes which cross-link documentation sets: for example, in automotive design documents, items like ECU might appear in each.

A possibility to improve the management of procedural knowledge and engineering data is nowadays proposed by Semantic Web related technologies because (http://www.w3.org/TR/webont-req/)

1. it is specifically designed to manage data

[1] D. Macrini (e-mail: domenico.macrini@elasis.it).
A. Maggiore (e-mail: aldo.maggiore@elasis.it).
A. Santagata (e-mail: antonino.santagata@elasis.it).
Elasis S.C.p.A. is an advanced engineering Fiat Group company based in Pomigliano d'Arco (ITALY) and involved in planning, product & process development and competitive improvement of vehicle and engines. It also develops Research Projects to innovate the product and/or process. Its professional competence, testing tools, validation methods as well as the integration allowed by its information systems, places Elasis in a competitive position within the area of vehicle development and experimentation.

2. one of its reference use case considers the management of "*a large body of engineering documentation… including design documentation, manufacturing documentation, and testing documentation*"
3. it has, among the other goals, the ontology's sharing, evolution and interoperability .

In its institutional efforts to promote innovations in automotive industry, at the Elasis' ICT Dep. is ongoing a project to evaluate the maturity of methodologies and tools related to Semantic Web.

Semantic Web technologies are actually tried out in Elasis to build up ontologies, i.e. searchable information models of engineering data.

Scenario: Testing in the automotive product development

The automotive product development

One of the primary problem faced by engineering activities is improving the use of available engineering data and procedural knowledge, a well recognized value whose management (modelling, integration, sharing and re-use) is one of the most popular topic in computer science.

This is especially true for manufacturing enterprises involved in the development of durable, complex, highly technological and large consumption goods (e.g. Automotive) with life cycles lasting years, whose control and management requires a great number of (different types of) documents.

Today this process is supported by a set of heterogeneous systems able to fulfil only the requirements of specific parts of the whole process and not of the process as a whole.

Testing in the automotive product development

One of the most critical activities in the automotive lifecycle development is testing, whose requirements prove challenging for every IT methodology, e.g.:

1. each development stage has its own types of tests: software simulation during design; workbench, road and crash test when the whole product or its parts are available
2. a complete test description requires many different documents: norms describing how to perform it (algorithms to use, sensors to install, parameters to record, driving maneuvers to execute, reports to compile), product's technical specifications, environmental (whether, road conditions,…) and any other information useful to a correct understanding of the results and its replica

Modeling: Relational vs. Semantic (Web)

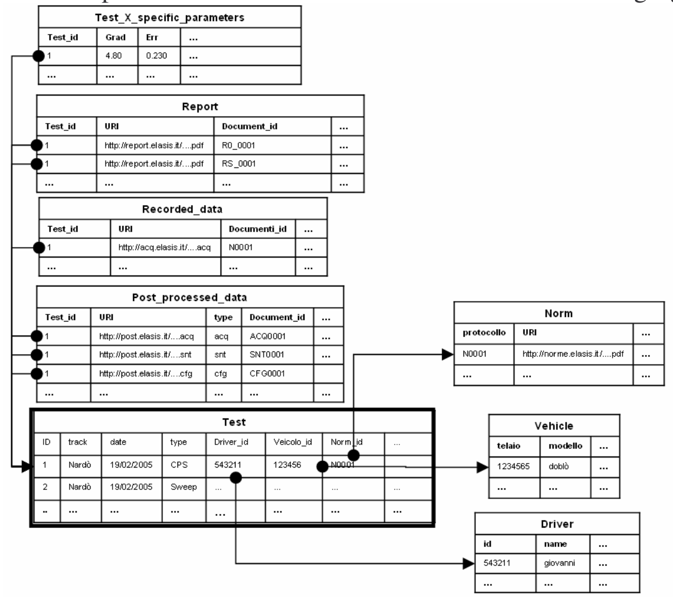
One of the most popular applications of IT is managing data archives re-designed to fit in the RDBMS model. But the RDBMS model proves scarce with engineering data (e.g. CAD drawings or parameters measured during a test). In this case, in fact, they are usually processed as an atom like entity (data file or written document), ignoring its contents so that they are classified, searched and retrieved only using metadata describing them (document management).

Lets consider, for example, road tests in the automotive industry. Some of the parameters' categories describing them are:

1. Environment: track used, whether conditions,…
2. Vehicle: type, model and configuration, tires,…
3. Driver: id, name,…
4. Type of test and norms specifying it
5. List of parameters of interest to record during test
6. List of raw data files recording test parameters
7. Numerical synthesis of the test (a short list of values inferred by the recorded data)
8. Reports
9. …

With strong peculiarities between different test types: handling road tests need a different set of parameters, numerical synthesis and reports from ride-comfort ones; software simulations are different from workbench tests; some of the tests involve only one or few components, subsystems or systems (e.g. engine or breaking system) others the whole vehicle;… From this easily comes how difficult is to define a general model to manage test data and, in general, engineering data.

RDBMS, the commonly used technology for this sort of problems, proves inadequate also breaking down the domain into pieces. For example, a graphic RDBMS model for a specific road test can be outlined as in the following figure:



Test_X_specific_parameters

Test_id	Grad	Err	...
1	4.80	0.230	...
...	...	...	...

Report

Test_id	URI	Document_id	...
1	http://report.elasis.it/….pdf	R0_0001	...
1	http://report.elasis.it/….pdf	RS_0001	...
...	...	...	...

Recorded_data

Test_id	URI	Documenti_id	...
1	http://acq.elasis.it/….acq	N0001	...
...	...	...	...

Post_processed_data

Test_id	URI	type	Document_id	...
1	http://post.elasis.it/….acq	acq	ACQ0001	...
1	http://post.elasis.it/….snt	snt	SNT0001	...
1	http://post.elasis.it/….cfg	cfg	CFG0001	
...	...	...	...	...

Norm

protocollo	URI	...
N0001	http://norme.elasis.it/….pdf	...
...	...	...

Test

ID	track	date	type	Driver_id	Veicolo_id	Norm_id	...
1	Nardò	19/02/2005	CPS	543211	123456	N0001	
2	Nardò	19/02/2005	Sweep	...	...	...	...
..	...	...	...	...	...	...	...

Vehicle

telaio	modello	...
1234565	doblò	...
...	...	...

Driver

id	name	...
543211	giovanni	...
...	...	...

Fig. 1. Some of tables and relations needed to represent road tests.

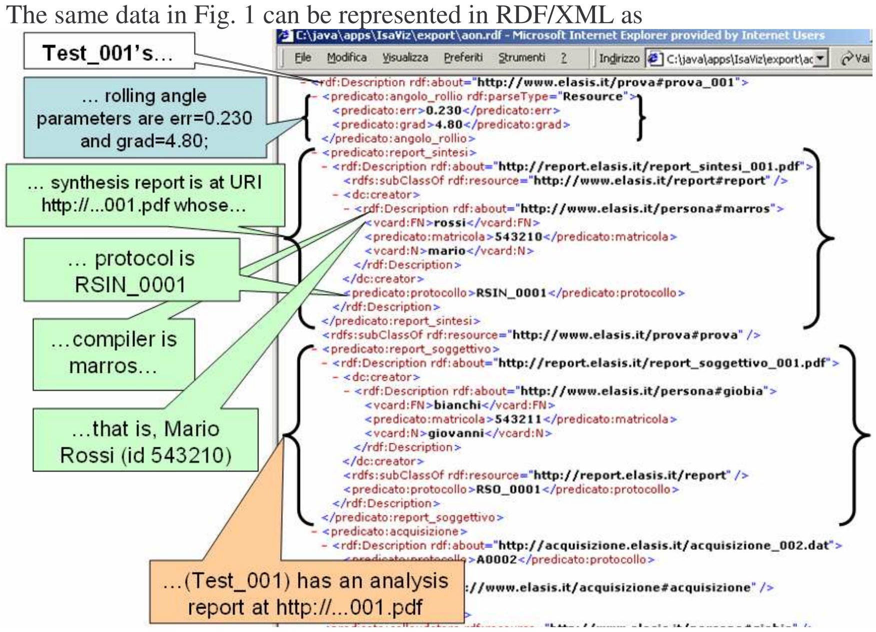
A typical schema for SQL query useful to extract information about the test with id=1 in the previous figure is:

```
Select test.*, vehicle.*, driver.*,…

From test, vehicle, driver, report,…

Where         Test.id =1

          AND test..driver_id = driver.id

          AND test.vehicle_id = vehicle.id

          AND  …
```

Fig. 2. an SQL query on Fig. 1.

As it is easily understood, queries built following the schema in the previous fig. 2 prove really weak respect to updates in the DB structure and can't consider variations on the theme, i.e. specific needs in some of the test (e.g. a new or different set of parameters or categorize a brand new type of reports).

The same data in Fig. 1 can be represented in RDF/XML as


C:\java\apps\IsaViz\export\aon.rdf - Microsoft Internet Explorer provided by Internet Users
File Modifica Visualizza Preferiti Strumenti ?
Indirizzo C:\java\apps\IsaViz\export\ac
Vai
Test_001's…
… rolling angle parameters are err=0.230 and grad=4.80;
… synthesis report is at URI http://...001.pdf whose…
… protocol is RSIN_0001
…compiler is marros…
…that is, Mario Rossi (id 543210)
…(Test_001) has an analysis report at http://...001.pdf

```
- <rdf:Description rdf:about="http://www.elasis.it/prova#prova_001">
- <predicato:angolo_rollio rdf:parseType="Resource">
    <predicato:err>0.230</predicato:err>
    <predicato:grad>4.80</predicato:grad>
  </predicato:angolo_rollio>
- <predicato:report_sintesi>
  - <rdf:Description rdf:about="http://report.elasis.it/report_sintesi_001.pdf">
      <rdfs:subClassOf rdf:resource="http://www.elasis.it/report#report" />
    - <dc:creator>
      - <rdf:Description rdf:about="http://www.elasis.it/persona#marros">
          <vcard:FN>rossi</vcard:FN>
          <predicato:matricola>543210</predicato:matricola>
          <vcard:N>mario</vcard:N>
        </rdf:Description>
      </dc:creator>
      <predicato:protocollo>RSIN_0001</predicato:protocollo>
    </rdf:Description>
  </predicato:report_sintesi>
  <rdfs:subClassOf rdf:resource="http://www.elasis.it/prova#prova" />
- <predicato:report_soggettivo>
  - <rdf:Description rdf:about="http://report.elasis.it/report_soggettivo_001.pdf">
    - <dc:creator>
      - <rdf:Description rdf:about="http://www.elasis.it/persona#giobia">
          <vcard:FN>bianchi</vcard:FN>
          <predicato:matricola>543211</predicato:matricola>
          <vcard:N>giovanni</vcard:N>
        </rdf:Description>
      </dc:creator>
      <rdfs:subClassOf rdf:resource="http://report.elasis.it/report" />
      <predicato:protocollo>RSO_0001</predicato:protocollo>
    </rdf:Description>
  </predicato:report_soggettivo>
- <predicato:acquisizione>
  - <rdf:Description rdf:about="http://acquisizione.elasis.it/acquisizione_002.dat">
      <predicato:protocollo>A0002</predicato:protocollo>
      ://www.elasis.it/acquisizione#acquisizione" />
```

Fig. 3. RDF/XML equivalence of Fig. 1.

The fragment in Fig. 3 confirms the first important benefit offered by semantic web respect to RDBMS model: it effectively integrates, in one uniform structure (the collection of RDF/XML statements), heterogeneous and not pre-defined data (URI, numeric and alphanumeric info,…).

Using SPARQL, it is easy to write a query on the set of RFD/XML statements equivalent to the one in Fig. 2:

```
Select ?predicate, ?value

Where     (<http://...test_001>, ?predicate, ?value)
```

Fig. 4. a SPARQL query on fig. 3

From the previous query follows that it is possible to manage asymmetrical adds to the base of data, i.e. it is possible to add new information/statements to a part of the resources and not to others of the very same type avoiding side effects on applications using them. It enables also to realize user interfaces flexible enough to support free browsing of the whole base of statements.

Previous considerations show that respect to design and maintainability, the data model proposed by Semantic Web (a list of RDF/XML statements) exceeds the one used by RDBMS (sets of tables + relations) in the field of road test data management and, in general, of engineering data.

Building a semantic information system

Data of interest for road test in the automotive field are highly distributed:

1. Reports, norms and other documents are usually archived in and managed by (different) document management systems
2. Test object descriptions (Vehicle, engine,…) are usually available from specialized database, but often those information must be complemented when it is modified for the test, e.g. installing a single new component to collect endurance data
3. Raw data recorded during a test are usually stored on centralized servers, post processing of raw data used by specialists to produce their reports can be stored on user workstations,…

Considering that mapping RDBMS data in RDF/XML statements is straightforward, for the project was attempt the collection of information available on user and centralized file system areas.

This problem was chosen because those are the kind of data whose modeling in a RDBMS system is arduous because

1. of their great number (a single test usually has a row file recording the parameters, tens of post-processed data produced by custom software, one or more synthesis report,…),
2. they are tedious to classify to make them searchable and then generally reusable

The solution was found in

A. the design of a specific ontology for test data
B. a spider able to search autonomously what of interest from a list of locations in the company intranet, analyze its contents and define RDF/XML statements according to an ontology.
C. a web browser to navigate the available set of RDF/XML statements

a neural network able to classify data to identify and suggest analogies between test results.

The ontology's design

To design the ontology, graphical tools and representations (such as the one proposed by IsaViz) were tried, but they proved unsatisfactory because of the quite large number of elements (about 50). For this reason a simple textual tree was used to define scalar values (leaves) and enumerable values (nodes). Predicates (arch) names are always the type of the object in the statement (e.g. tire or test type). A fragment of the defined ontology is

To design the ontology, graphical tools and representations (such as the one proposed by IsaViz) were tried, but they proved unsatisfactory because of the quite large number of elements (about 50). For this reason a simple textual tree was used to define scalar values (leaves) and enumerable values (nodes). Predicates (arch) names are always the type of the object in the statement (e.g. tire or test type). A fragment of the defined ontology is

- Test [track+date+car model+type]
  - Date [YYYYMMDD]
  - Driving maneuvers [CPS100,PIM,…->links to a resource]
  - CAR [track+date+car model+type+CAR]
    - Model [Stilo, Punto,… ->links to a resource]
    - Chassis number [an alphanumeric code]
    - Tyres [CAR+tyre type]
      - Model (Pirelli P3000,…->links to a resource)
      - Front pressure [a numeric value]
      - Rear pressure [a numeric value]
  - Post processed data [track+date+car model+PPROCESS]
      - [a list of URI]

  …

Fig. 5. a fragment of the defined ontology.

An example of actual data is

- Nardo_20050219_FIAT_PUNTO_CPS100
  - 20050219
  - ->[CPS100]
  - Nardo_20050219_FIAT_PUNTO_CPS100_CAR
    - ->[Fiat_punto]
    - 123456
    - Nardo_20050219_FIAT_PUNTO_CPS100_CAR_TYRE
      - ->[Pirelli P3000]
      - 2.3
      - 2.3
  - Nardo_20050219_FIAT_PUNTO_PPROCESS

- ->[http://…]
- ->[http://…]

…

Fig. 6. a fragment of the data.

Spiders and data collection

Of the spider were implemented 2 releases. The first one searches the file system starting from a list of paths, the other a list a web sites in the company intranet.

To work, the *file system spider* requires locally mounted disk areas to search:

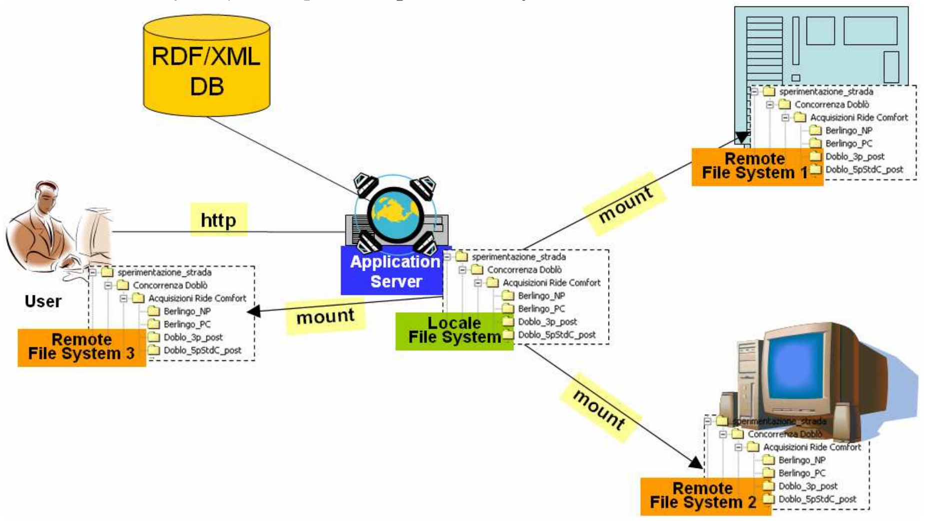


Fig. 7. the File System spider.

The *web spider* is able to search via http web sites with directory listing option enabled:

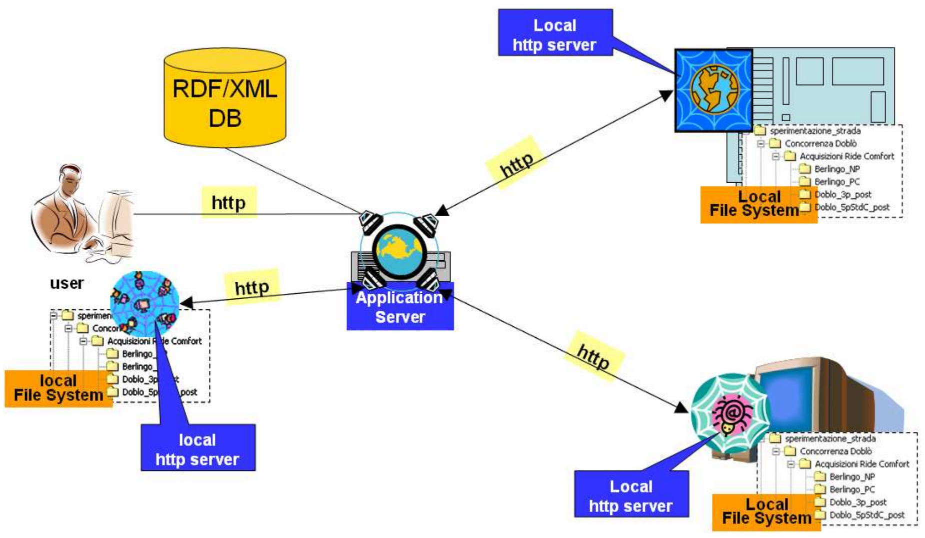


Fig. 8. the web spider

Both work well, but the second one is much more flexible then the other. While crawling the two spiders collect information both from the path and the contents of (some of the) files to define statements according to the designed ontology.

A great part of the numerical and descriptive information are extracted from textual files. A study was conducted to extract information from reports written in some of the office file format when they are available in XML. In this case to understand and find data of interest in an office document, very important is to consider the reference standard template used to write it. In fact, great part of the written technical documents requested during a product lifecycle is written according to well defined and known templates to allow to anyone who knows it to easily find what of interest. These reference templates specify document index, how to name titles, the content of each chapter and its structure, unit of measure,… simplifying data retrieval in XSL-T when the document is available in XML.

It was realized also an on the fly translation of office documents in xml using OpenOffice.org running in server mode:

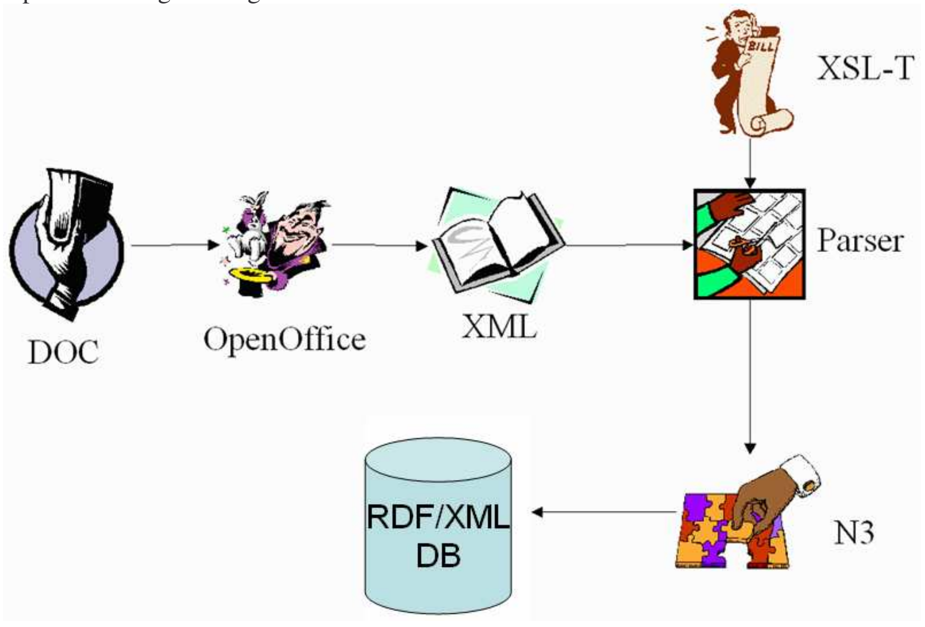


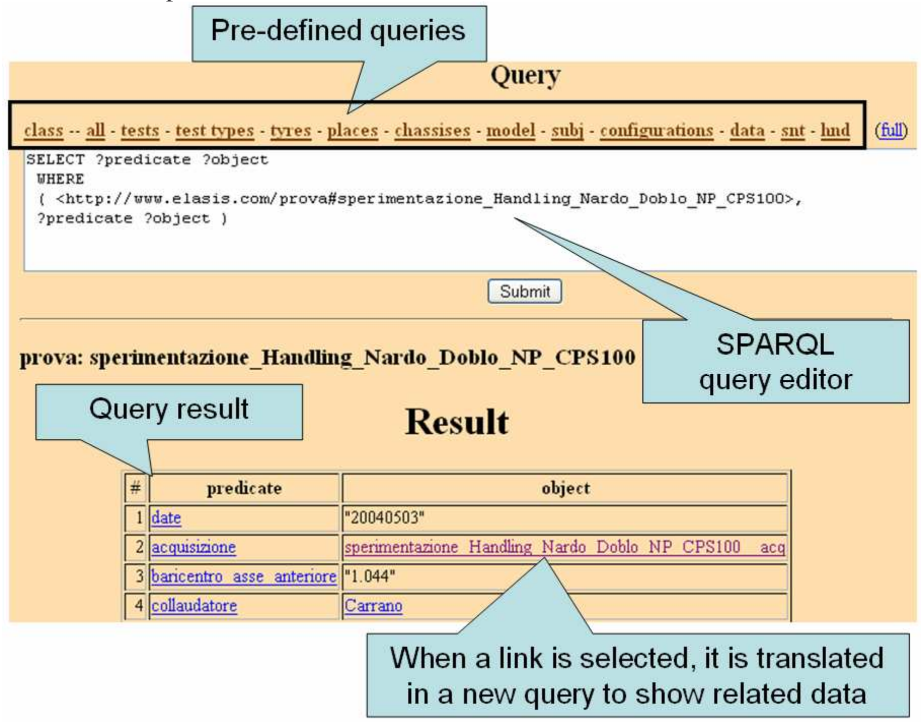
Fig. 9. from office docs to RDF/XML

In any case, considering that reports are based on information collected post-processing the raw data recorded during test and that the result of post-processing is saved in well formatted ASCII files, from these files can be easily extracted all the information so the use of OpenOffice.org/XML/XSL was put apart.

Data retrieval: the web navigator

Semantic web is usually represented as an oriented graph. The graph's recursive definition suggests the realization of new and friendly ways to explore data: a generic user interfaces allowing web browsing on any RDF/XML data sets as in the daily experience of world wide web.

Based on this idea a web navigator was implemented. A page of the navigator can be divided in 3 parts:


Pre-defined queries
Query
class -- all - tests - test types - tyres - places - chassises - model - subj - configurations - data - snt - hnd (full)
Submit
SPARQL query editor
Query result
When a link is selected, it is translated in a new query to show related data

```
SELECT ?predicate ?object
 WHERE
 ( <http://www.elasis.com/prova#sperimentazione_Handling_Nardo_Doblo_NP_CPS100>,
 ?predicate ?object )
```

prova: sperimentazione_Handling_Nardo_Doblo_NP_CPS100

Result

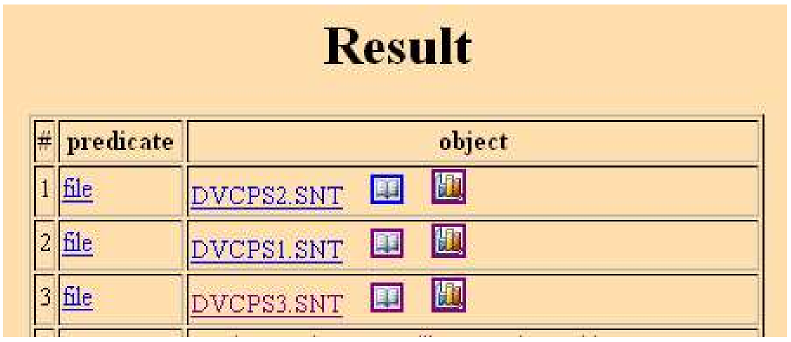
#	predicate	object
1	date	"20040503"
2	acquisizione	sperimentazione_Handling_Nardo_Doblo_NP_CPS100__acq
3	baricentro_asse_anteriore	"1.044"
4	collaudatore	Carrano

Fig. 10. the web navigator (1)

1. Pre-defined queries to retrieve well known sets of data
2. An editor to write queries using SPARQL
3. Query results where non-scalar information (the ones that can't be subject of RDF/XML statements) are shown as selectable web link. When one of these web links is clicked, a new query is defined and used to further examine the data

Even thought it is poorly designed, this interface allows a seamless exploration of all the available data confirming the advantages promised by the idea behind it.

An improvement of this functionality would be the use of template to present the result of a query according to the selected element type and show them according to users' wishes and needs. A first attempt was made managing office documents and some of the data file:

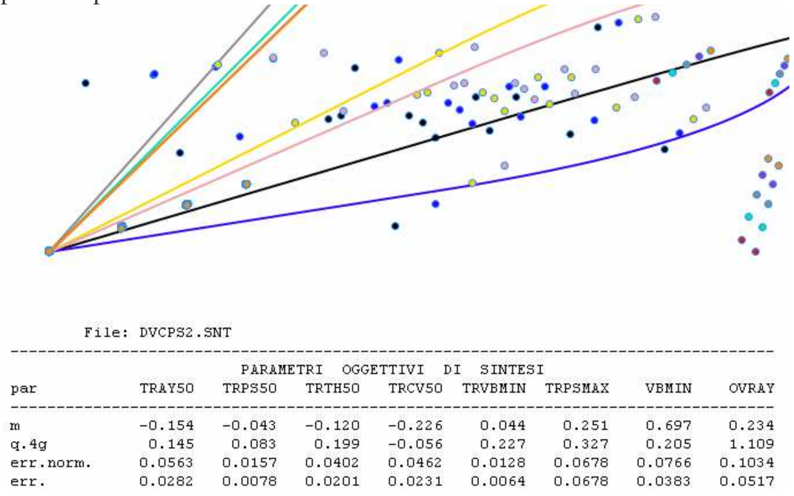


Result

#	predicate	object
1	file	DVCPS2.SNT
2	file	DVCPS1.SNT
3	file	DVCPS3.SNT

Fig. 11. the web navigator (2)

Selecting the "book" icon, in both cases it is possible to show the content in its original format. For some of the data file is also available a "bar chart" icon to request its graphical representation in SVG:



```
       File: DVCPS2.SNT
---------------------------------------------------------------------------------------
                          PARAMETRI  OGGETTIVI  DI  SINTESI
par             TRAY50    TRPS50    TRTH50    TRCV50   TRVBMIN   TRPSMAX     VBMIN     OVRAY
---------------------------------------------------------------------------------------
m               -0.154    -0.043    -0.120    -0.226     0.044     0.251     0.697     0.234
q.4g             0.145     0.083     0.199    -0.056     0.227     0.327     0.205     1.109
err.norm.       0.0563    0.0157    0.0402    0.0462    0.0128    0.0678    0.0766    0.1034
err.            0.0282    0.0078    0.0201    0.0231    0.0064    0.0678    0.0383    0.0517
```

Fig. 12. the web navigator (3): graphical rendering of numerical data

Data classification: the neural network

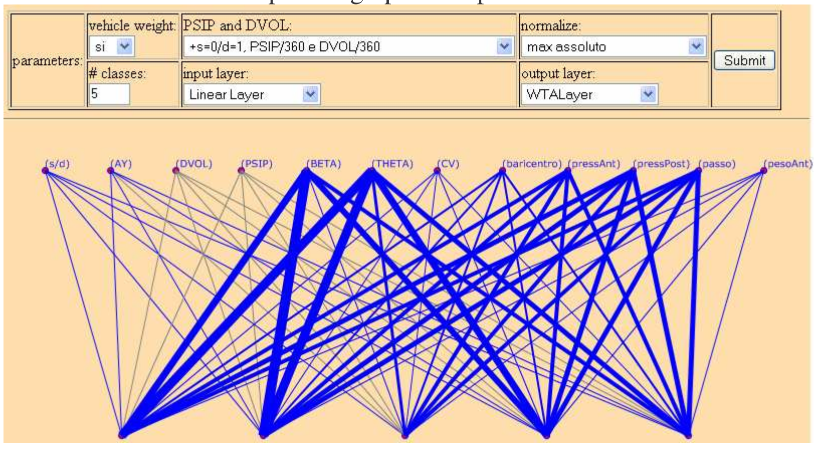
One of the major problems in managing engineering data is allowing and encouraging their reuse. But, as shown, engineering data are very complex objects and their modeling and management is arduous. Semantic web offers a good methodology to model them. It can also improve reuse via free browsing of the data, but a better solution is to be found somewhere else. For the project some statistic and AI technologies were considered.

After a first overview, inferential statistics, fuzzy logic and expert systems were excluded because they require the mapping of a skilled specialist's experience in a set of formula and rules, a process that proves long lasting and uncertain with vast and complex domains like testing. For this reason neural networks were tried and, as first application, test classification was chosen. This subject is of interest because recognizing analogies is a valuable information but proves difficult to be carried out.

The chosen architecture is the self-organized map sometimes referred to as Kohonen map. Kohonen map are interesting because they require the unsupervised learning method, i.e. they do not require the definition of pairs of input objects and desired outputs but use a list of training data to learn how to partition them in different sets.

The use of Kohonen maps requires a set of data transformation because all of the input values should be between 0 and 1. This done, its use proves effective, at least for the type of test considered: its performance were evaluated against a graphical representations of one of the main numerical synthesis commonly used by engineers and it gives comparable results.

Up to now the prototype has only a workbench to initialize the neural network and analyze its results. It also computes a graphical representation of the net:



parameters:	vehicle weight: si	PSIP and DVOL: +s=0/d=1, PSIP/360 e DVOL/360	normalize: max assoluto	Submit
	# classes: 5	input layer: Linear Layer	output layer: WTALayer	

Fig. 13. the neural network

Tools used

The software prototype is realized in java and uses (among the other libraries):

1. Struts (http://struts.apache.org/) from *Apache Software Foundation* to implement the application infrastructure according to the MVC pattern
2. Jena (http://jena.sourceforge.net/) from *HP Labs* to model and manage RDF/XML statements
3. Joone (http://www.jooneworld.com/) from Paolo Marrone to define and use the neural network

Case Study Deliverable (up to day)

Up to day the major deliverables of the project are

1. the definition of an ontology describing (some of the) road tests
2. a software prototype/technology demonstrator implementing
    - a spider capable to explore a list of web sites in the company intranet to identify and analyze files describing (or related to) tests and produce RDF/XML statements defining their Semantic Web representation according to the specified ontology
    - a web navigator to freely explore the previous ontology and visualize/access documents. To simplify the data analysis, for some of the files, the prototype is able to draw a graphic representation using SVG
    - a neural network to classify (a subset of) tests.

Conclusion

According to the experience made with this project, Semantic Web has a huge potential in engineering applications.

Among its pros, the most important is the plasticity of its data model that allows a seamless representation and integration of different kinds of information. From this point of view it exceeds the relational data model, the one ordinarily used to implement a wide range of engineering applications.

Another important advantage assured by Semantic Web regards the software robustness, especially the data management procedure, because of the possibility to add information asymmetrically, that is as they are available, avoiding updates in the data structure or re-design of query procedure.

Last but not least, tools used integrate easily with existing and well knows development solutions (architecture, DB,…), saving investments in training and infrastructure.

Among its cons, the most important is that it is jet scarcely known a part from a restricted number of advanced technologists.

For example, even thought it has great potentials and a promoter like the W3C behind, Semantic Web is comparably much less know than OOT at its beginnings: 15 years ago OOT was less mature, every scholar proposed its own approach, there were discussions on the same nature of OO but the subject was much more discussed, every technical magazine had its column, there were books and was easy to find training occasions to learn it. For this reason apprehending semantic web competences is comparably much more difficult.

Another big problem is the (very) short list of competitors in the offering of tools to implement semantic web technologies and, what is worst, no major IT player has in its catalogue tools or applications using them. For this reason it is today very difficult to propose the use of Semantic Web to develop new applications.

From this point of view would be interesting to have at least the definition of a standard API via Java Community Process.

To summarize:

1. Pros:
   - improves the data modelling process and exceeds the capabilities of the relational model
   - assures the data reuse: via RDF/XML heterogeneous contents from different sources (RDBMS, file systems, documents, web, other systems…) can be effectively integrated and ontologies are robust respect to changes
   - integrates easily with existing and well knows development solutions (tools, architectures,…)
2. Cons:
   - is still a subject for ‘gurus’ and very few are the available training events, books, tools,…

- requires new competences in designing applications (learning the best use of its peculiar object model)
- lack of competition between primary IT actors in tool development. For the java world it would be interesting the definition of a standard API via JPC.