Making Business Sense of the Semantic Web

Zavisa Bjelogrlic[1], Dirk-Willem van Gulik[2], and Alberto Reggiori[1]

[1] @Semantics. S.R.L., via Teulada 71, I00195 Rome, Italy
[2] Leiden office
{z,dirkx,albe}@asemantics.com
http://www.asemantics.com

Abstract. The Semantic Web and RDF offer a powerful platform for a wide range of applications. However, at this time, the semantic business is still in an embryonic stage. This paper analyses real-life applications: (i) A system in operation used to improve the presentation of corporate data, (ii) a prototype of a news clipping service and (iii) a feasibility analysis of a distributed, public administration database. Common to these applications is a "web database" model where resources are "URI-ed" - made available on the Web - and "RDF-ed" - described by one or multiple authorities. Descriptions are processed by RDFStore and made available to applications. The success factors of these three applications are analysed, and an application model is drawn up with requirements and desirable features for RDF Engines. In conclusion, we look at how and where the Semantic Web business can grow.

1 Introduction

The killer apps of the millenium web so far have been search engines, the deadliest of them all being Google, which tells you, as you search, what is most important to everybody. How it tells you that comes from a tiny bit of markup spread through the whole of the web, the link. The Semantic Web takes that idea a lot further: RDF provides a thread of meaning through the web, a golden thread of individualised meaning, making the web *your* web. Google is doing good business by telling us whats important to everybody. And in this paper, we show how good business can be done by telling you whats important to *you*.

The Semantic Web and its core technology RDF [1,2,3,4,5] offer a powerful platform for a wide range of applications. By "semantifying", the Web can be made searchable and browsable by computers and humans alike. This is the starting point for applications which are better and faster in presenting information to the user and which build new services by cross-correlating completely different data sources.

Talking about *the* Semantic Web can be misleading. The technology can be used for a wide variety of data resources in corporate intranets or even in standalone applications. Resources identified by a URI [22] ("URI-ed") and described in RDF ("RDF-ed"), can be searched and navigated through by following relations using so-called RDF engines. RDF engines have rapidly become an interesting alternative for a variety of database/data mining applications, knowledge

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 818–833, 2003.

```
RDFBusiness = does_application_add_value &&
      willing_to_invest(date) &&
      better_than_classic_approach &&
      does_it_work( URIfy-ing, RDFy-ing, gathering_and_ingesting,
             indexing, searching_and_browsing, application_processing);
```

Fig. 1. Evaluation of success factors in the Semantic Web business.

management and documentation management systems, and other "classic" applications. The flexibility of RDF, its ability to gradually describe the resources and to keep different descriptions of the same resource is key to this. This makes the approach feasible for a range of different applications, some of which were up to now too fluid or simply lacking the regularity needed to fit into existing relational database models. To paraphrase Tim Berners-Lee[1], we had a situation where *anyone could say anything about anything without breaking the whole system.*

We characterise this idea as the Web Database approach. Of course, this is a marketing simplification and an attempt to define in as few words as possible the business use of the Semantic Web. Different attempts to integrate the Web and the database world are described in literature [6,7,8]. Our model can be described in five steps:

1. Resources are URIed and published on the Web.
2. Resources are annotated and described using RDF. Original resources are left untouched and can be protected or even partially hidden on their sites;
3. RDF descriptors are collected and processed by RDF engines;
4. Applications use a query language to search and navigate through RDF descriptors and perform additional processing over query results.
5. If needed and authorized, applications use the URI to access the "visible" part of the original resource.

At this time, the "semantic business" is still in an embryonic stage. Inhibitors for RDF to become the mainstream technology range from business to technical reasons. In this paper we will start with these inhibitors and analyse in chapter 2 those factors that are key to overcoming them. This is illustrated using three real-life RDF applications described in chapter 3. The analysis draws up a draft of an Application Model (chapter 4). More than a strictly technical model, it is a list of requirements and nice-to have features from a customer/business development point of view. Finally in chapter 5, we identify how and where the Semantic Web business can grow.

2 Success Factors

A simple evaluation of the success factors in the Semantic Web business is made in Figure 1 with a program statement (assuming left-to-right evaluation).

`does_application_add_value:` This is the first question. In the current situation, it is not so easy to make revenues out of data and services published

on the Web. Why would somebody invest to semantify the Web? Does it really bring added value? Will an improved, semantic, Web be easier to sell? Semantifying can ease aggregation of data and, as an end result, reduce barriers between providers. Is this really wanted by the information providers? In the case of corporate use, does the semantic approach offer advantages which will pay off the investment made?

`willing_to_invest:` Supposing the first factor is True, and we have something valuable to offer, is somebody willing to invest in new technology? This may be a temporal factor: after years of the Internet boom, over-expectations and high figures in technology spending, we are not in the best situation today. Investment depends on the customer (or VC) policy on which we have little if any direct influence. There is one important point: is this investment in semantic technology something which will continue to yield results over the next years, or can it also produce just short term results? A technical characteristic of RDF is key here: RDF allows for almost complete separation of RDF descriptions from applications as URIs are used to reference the data. This implies that once RDFed the resources can be used years and years in one or in many, evolving applications, even if those applications do not have ready access to the data.

`better_than_classic_approach:` as for everything else in software, *"there are many ways to do it"* [9]. Why choose a new technology where only few adepts are experts and everybody else must learn new things? There are many academic answers, but it is not easy to find justifications for lack of programming skills and existing, mature tools, especially when you have on the other side XML technology and well-established RDBMS with JSP or ASP GUIs. We believe the right question here is not if Semantic is better but WHEN it is better.

`does_it_work:` at the end of the day, from a marketing point of view, the question is if all this stuff will work when put into real, day-to-day operation. What about the complexity of extracting metadata and defining them in RDF? Performance questions? What happens when data change daily or even faster? What happens in reality when data is distributed among networks of sites managed by different people? How appropriate is RDQL [13] for building more complex applications? How does it compare with SQL? Again, this is not a technical question but more a problem as to how fast an SQL programmer will be able to write applications in RDQL or in some other RDF query language.

3 Business Cases

Three cases are described in this paper. These are currently at different stages, ranging from a system in operation to a feasibility study.

3.1 Image Showcase

Image Showcase is a project developed for the European Space Agency ESA, ESRIN Frascati, `http://www.esa.int/`. The Agency publishes examples of satellite images acquired by different sensors of the ENVISAT satellite, the ERS-2

satellite and other satellites. These images are used as examples to explain and promote the use of earth observation techniques. The audience is the general public including youngsters and researchers not specialized in the use of remote sensing data. Images are annotated with text - which explains the background, the applications and provides some basic technical details about the images used. The image is normally published in several versions - a small thumbnail, a medium size quick-look suitable for slow downloading and a full size scientific quick-look of several megabytes in size. The main image is often augmented by as additional pictures that show annotated details of the larger images.

Cases are prepared by hand by different groups of experts following simple guidelines and using simple form templates. New case studies are added daily but, once published, they do not change. The total number of studies ranges between 500 and 1000 and grows each year. The studies are referenced or announced at different Agency portals and use different technical solutions. It is interesting to note that requests coming through Google from people searching for text contained in stories are among the *highest accesses to portals* .

The main objective of the project was to simplify and unify access to the Image Showcases for general public, in the "6-66 years range". ESAs Image Showcase is a free service open to everybody. The objective was to avoid all complex interfaces, remote sensing jargon, various specialist map interfaces, java applets and browser plug-ins and simplify the interface to the extreme to make it compatible with all browsers. The example given to follow was Google: one field to fill out and immediate fast, relevant, results on the next click.

Approach. The approach adopted starts from parsing of the original, handwritten data, and creating RDF descriptions. One RDF "sister" file is created for every case. Part of the attributes are extracted automatically from available catalogue systems [10]. The vocabularies adopted for RDF are relatively simple.

1. each original showcase image is assigned the URL of the top level page. And each of the HTML pages and images which together make up the showcase are described with simple RDF properties;
2. each RDF description is made relative to the source HTML base i.e. ENVISAT or ERS-2 catalog;
3. simple Dublin Core (DC) [23] properties are used for the title, date, description (the story);
4. extended DC Terms (DCT) [23] properties are used to express the relationships between HTML pages, images and different versions of the images including thumbnails (appropriate for result lists) created on-the-fly during ingestion;
5. the space coverage, that is latitude and longitude, and the bounding rectangle of the image are described using DCT Box (from the extended Dublin Core Terms) and some ad-hoc OpenGIS Consortiums GML [11] vocabulary is used for the four-corner polygons;
6. RDF Schema [24] seeAlso property is used to link up different RDF descriptions about different HTML pages and images;

7. each image description is augmented with simple ad-hoc properties about its media content type, size and geometry, including presence of small thumbnails.

The conversion process is run regularly through a `make` command on every resource site to parse only recently changed files. Additional RDF descriptors are used to manually configure relations to other, non-semantic, Web resources. For example, so called "guide pages" for the satellites and sensors were defined in a separate RDF file parsed at query time. This was done as a first step to extending the Semantic Web to other data sources not included directly in the project. RDF descriptors are collected regularly from all sites where resources reside. This is done by a simple RDF gathering mechanism. After gathering, all descriptors are ingested in RDFStore [12].

Perl DBI is used to query all descriptors using RDQL language [13]. Queries are simple, corresponding to the extremely simple presentation required: (i) all showcases for a given collection sorted by date, (ii) free-text search over stories, (iii) limitation of the geographic region of search, (iv) path navigation for all images related to a case.

Different Web interfaces have been built over this query to provide a simple "Google" like interface with GEO specific details details.

Conclusion. Table 1 summarizes this case study. The system is well accepted but its full impact has not yet been evaluated. The semantic solution is seen as just another search engine, perhaps more flexible. This is also due to the fact that basically only *one type of data*, even if lost in different collections, is handled. An interesting extension will be to semantify other related data and to cross-correlate this with the semantic presentation of the showcases. We expect at this point to reach the critical mass and show how, with minimal effort, more complex applications can be built over the initial description of data. See `http://demo.asemantics.com/biz/isc/` for a demonstation.

3.2 LMN – Last Minute News

A variety of news clipping services are in operation today. These range from speciallist services such as Moreover to generic systems like Google News and others. A short overview of similar systems with references to more detailed information can be found in [15].

LMN is a prototype system of a real-time news clipping service. LMN was intended mainly as a technology demonstrator and as a mock-up for testing the dynamic aspects of RDF ingestion and indexing. The objective was to offer narrow search functions for fresh daily news, trying to reduce false-positives by considering additional information about sources and by offering to the user the ability to flavour the source express his/her rating of sources.

Approach. The news sources are harvested at regular intervals and parsed to create RSS1.0 compliant RDF descriptions of the contents available. Parsing is

Table 1. Factors Key Metrics for this Case Study

Type of Application	Web site management Unified access to sparse data. Portal service with free, unlimited access.
`does_application_add_value`:	Yes, it improves the access to data already published by different groups.
`willing_to_invest`:	Yes. In an evaluation phase to test if the new solution, already made public, improves the presentation of data, and the acceptance of data by the general public.
`better_than_classic_approach`:	Yes. No appropriate alternative solution found.
`does_it_work`:	
– URIfy-ing	Resources already published on the Web.
– RDFy-ing	Persing of manually-written text. Some 5 percent of erroneous parsing corrected by rewriting the originals.
– Gathering and ingesting	A hierarchical mother-index approach is used for guide gathering
– Indexing	RDFStore
– Searching and Browsing	RDQL free-text is combined with other parameters and spatial position. Navigation through all images.
– Application processing	Sorting. GUI and map handling (by a WMS compliant server [13])
Business Model	Software development and system integrations.

done on different sources, which, not surprisingly, show similar structure. Parsing extracts description, date, date of acquisition, author (when available), title and all images related to the article. In some cases, news articles offer a see-also section with related articles.

It is interesting to note that these see-also resources either belong to the same group or are very far and thus do not pose a competition threat to the provider in either case. This is an interesting indication of an inherent submission of Semantic Web to the data when used to aggregate data coming from different sources.

The whole parsing process is currently configured manually, typically by first defining a filter configuration and then cloning it for other sources. Parsing difficulties were encountered only for see-also links which use different patterns. Also, the headlines and front-page section of the origin Web site are more prone to manual editorial intervention - making it more error-prone to automatically harvesting.

Other attributes, such as section, type of source and flavour of the sources (e.g. serious newspaper vs gossip source, right/left or liberal/conservative or Republican/Democratic ratings) were defined at parser configuration time and assigned to a group/section of news.

The classification-flavouring of news-sources is subjective and depends on the orientation of the reader. To manage such diversity, the LMN user is allowed to define *his own flavouring of sources* which will override default settings. This is collected in a separate RDF description of sources that is parsed on-the-fly at search time and used to flavour the search results to rank the results according to the users preferences. Although this is just an experiment which does not show significant advantages to the user when only 10-15 sources are clipped, it might become of particular interest for analysis and intelligence work where not only information itself counts but also the *kind* of source and its *reliability* - trustworthiness is important.

Once parsed from the news sources, RDF data are ingested using RDFStore. That part of the data relating to user preferences is left undigested and used when flavoured search is invoked (i.e. parsed on-the fly). A simple RDF gathering protocol is used in this case as the RDF creation is done, after harvesting, in a single pass. The whole mechanism is managed by a series of make commands executed regularly.

Three interesting aspects were touched during ingestion and indexing:

1. A hot-swappable pool of RDFStore indexes was used to optimize updating of data without interrupting or slowing down the search and retrieval functions;
2. Depending on the provider site, news articles may disappear from the site after a short period, while others remain archived for a long time and can be retrieved later. This may lead to a *404 not-available* error badly perceived by the user of a aggregating service and to the erroneous ordering of hits. A simple solution was adopted here by defining the expiration time attribute for every section or group of resources. Following our baseline to try to keep things simple and efficient, we used a historic time-to-live acronym – TTL – for this attribute. This was an application level hack that uses RDF structure to define the TTL. Further work goes probably in favour of thinking about a more system-level solution: a TTL attribute on a triplet which might then be used for optimization of the RDF engine.
3. A special case is the RDF definition of user preferences and flavours of sources. This is implemented as RDF parsed on the fly at the query time and used for a single query. Being a preference only, these are small files not creating particular performance problems but again raising interesting questions about the usage of TTL on these flavoring triplets.

The query functions offered are (i) Free-text query on words contained in the description or title. (ii) Navigate (show) all pictures referenced by an article (or all pictures related to a group of hits, e.g. all images related to SARS). (iii) Navigate through see-also sources referenced by an article (iv) Flavour the search, i.e. consider the users classification of sources in ranking the hits.

Table 2. Factors / Key Metrics of Case Study II

Type of application	Unified access to external sources not participating in the initiative. News clipping service.
`does_application_add_value`:	Not clear yet. Demonstrator just open. As objective, added value should be in narrow search, and in user Flavoured search
`willing_to_invest`:	Not clear yet. Positive first comments.
`better_than_classic_approach`:	Not clear to the customer.
`does_it_work`:	
– URIfy-ing	Resources already Web published
– RDFy-ing	Parsing of HTML published data in a limited domain.
– Gathering and ingesting	A spider runs at regular intervals, parsing and keeping a local buffer of RDF data.
– Indexing	RDFStore
– Searching and Browsing	RDQL free-text combined with news/news-sources attributes. Navigation through graph for All images related to this/all news found and see-also articles.
– Application processing	Sorting/ranking, also taking user preferences into consideration (Flavored search). Presentation of results.
Business Model	Aggregator service or system development

The query is packed in one simple user interface and can operate both as a pull service and as a push service.

Conclusion. Table 2 summarizes this case study. Though we received positive first comments we also perceive reluctance to invest in new Internet services, even if such services seem to offer some added value as the difference is not immediate visible. Free-text search indexes + directory structure show similar results. Two aspects are positively considered: (i) application flexibility (another specialised service could easily be cloned) and (ii) user preferences and customization of the service. Also, and though easier then expected initially the latter requires more work for see-also data and headlines/front pages

It is difficult to draw conclusions for an application just open and still in finalisation and improvement phase. We believe the demonstrator can explain well the advantages of the Semantic Web and the ability to build different applications over RDF-described data. Such is also the case when operating on data belonging to different, non-participating, providers. `http://demo.asemantics.com/biz/lmn/`

3.3 Unemployment Records Database

Internet information systems aimed at supporting and streamlining employment procedures and ease the match of job requests with available offering were developed recently in Italy [16,17]. These systems cover the formal aspects related to the employment law in Italy and serve as exchange vehicles between employers and employment agencies/companies on one side and local administration on other side. The system also covers special rules fostering the employment of disabled people and are used to rank the candidates according to rules defined by local authorities for work offerings related to public services and administrations.

Todays systems started as stand-alone database applications, evolving through distributed database applications based on simple batch exchange of data and are evolving today to Internet services where a single, centralized service is used by various local authorities. Years of database development has led to a well-established data model and a good understanding of procedures. However the established model is commonly found lacking when:

1. Rules about employment and ranking algorithms change frequently and need to be tailored or modified by local authorities. This makes difficult to maintain a central service that offers up-to-date yet regional customized service to different local authorities.
2. Unemployment profile, skills, education and work experience descriptions vary from year to year according to the evolution of technology and society. The standard classification is a precise and rich thesaurus but always lags behind significantly, e.g. computer technology experience was only added recently.
3. Inadequacy or just slow evolution of the thesauri leads to use of free text to describe the employee his profile and, consequently, to the need for a free-text search. This is similar to a problem of semi-structured databases [7].
4. Operations are highly distributed among local offices. This requires either a central service or an efficient distributed database model.
5. Once ingested in the RDMBS, data is in practice lost and cannot be restructured. Technical solutions allow exchanging data saved in the RDBMS with other players and applications, tracking of changes and reconstruction of the change history but, in practice, this is costly and difficult to manage in case of multiple instances with changing data structure. Data are visible on the Web through forms and report pages but there we have the problem of opaqueness of the URI and data cannot easily be searched using this view.

These points were challenges for thinking about an upgrade of the solution based on Semantic Web technologies. Today, this is a feasibility analysis only.

Proposed Approach. Current data exist as SQL databases. The first step is to make this data Web-available. A nave approach [18] is proposed for this: SQL data are described as RDF equivalents of unemployment records. This

can be done on-the-fly by any Web-SQL gateway. A natural primary key exists in the procedure the unique fiscal code assigned to every legal entity in Italy that can be used as URI. Published data need to be protected by authorization/authentication and proper encryption rules that guarantee full privacy of data. Publishing can also be *selective* and limited to part of the record for use by other systems (e.g. regional or central authority), keeping the rest for internal use (local authority).

Doing this only to allow cross-searches is not worth the effort and will probably be done better by a database solution or by ad-hoc applications. A point we think important is that records published are also available to other applications. Any other public administration system can access, if needed and authorised, the data published in RDF. For future use, existing SQL infrastructure can be upgraded with an intermediate

RDF (XML) step between the GUI and the SQL back-end. This will also allow to simplify the maintenance of the GUI for changes of procedures or customisation to local needs a costly process today.

In our view, RDF descriptors are published by a local office that is authoritative for a given geographic or domain area. These offices maintain the records and register all updates. The expandability of RDF allows defining of *basic data* by one authority, giving at the same time a *large flexibility to other authorities* to add additional attributes for the record involved. The schema proposed is:

1. Fiscal code used as URI to identify the unemployed person and his record;
2. A basic description is created with all formal data (full name, address/es, birth date and place, etc.). This is a short and fully standardized structure (no significant changes are expected here long term);
3. Basic data are extended by family structure records which contains all information about family status and similar information which can be used for ranking or subsidy calculation;
4. Education history and employment history is described by a sequence of related descriptors.

Descriptors are collected regularly by local instances of the RDFStore engines at the local office. Descriptors can also be collected by a regional centre to region-wide consultation services. The time-to-live attribute of the resource cannot be defined a priori the record changes when the unemployed person changes his status. There is however an expiration time - time for which the administration shall keep records or time for which subsidy is allowed. The gatherer must be able to detect updated or changed records to avoid full ingestion of data which might become, in this case, too heavy. The number of records foreseen for Rome and its suburbs, considering historic records saved, is on the order of 1m. Gathering is configured using a two-level mother-index for the domain (regional unemployment offices) and local mother-indexes. Time stamps need to be kept to select descriptions which will be ingested.

A more dynamic approach that uses on-the-fly conversion to RDF and maps RDQL queries into SQL queries could also be adopted. This will avoid gathering and is a good solution for integrating information coming from different data

Table 3. Factors / Key Metrics of Case Study III

Type of application	Improvement of a pure database application in case of a distributed/federated database.
`does_application_add_value`:	Yes, simplifies handling of records.
`willing_to_invest`:	Feasibility analysis only.
`better_than_classic_approach`:	Difficult to explain
`does_it_work`:	
– URIfy-ing	Needs a SQL2RDF or SQL to XML to RDF gateway and a mechanism for URI2URL mapping.
– RDFy-ing	Deterministic conversion, relatively simple to perform.
– Gathering and ingesting	Alternatively: a spider approach or on-the-fly conversion and RDQL2SQL gateway.
– Indexing	RDFStore. An efficient update mechanism is needed.
– Searching and Browsing	RDQL free-text combined search. Navigation through employment and education history is needed.
– Application processing	May become heavy due to complex combined queries and due to complex ranking algorithms.
Business Model	Software development.

models in SQL databases. It is too early to say now what will be the best choice. This will be a matter of query performances *versus* pre-processing load and defining what is an acceptable sync/delay with respect to real-time data.

Independently from the approach adopted, it is important to note that we need an *intermediate view* on the Web to exchange data among different administrations. The trend today seems to be XML. In our opinion, RDF or RDF described XML data will be a better solution allowing not only to *exchange* data but also to *query and navigate through information directly*, i.e. without moving data in XML from application to application and later integrating the applications. One can argue that RDF data also needs to be ingested or processed by an application. We believe that this is less costly, faster and more robust than ingesting in separate applications (1st step) and later integrating the applications (2nd step).

As a final step in the proposed approach, applications are developed over the local and regional centre mainly to provide search functions with criteria: (i) skill/education profile, both from controlled lists and free text description, (ii) work experience, both from controlled lists and free text description, (iii) age, (iv) belonging to special categories (e.g. disabled) and (v) preferences and willingness to move/relocate. A similar approach is used for creating ranked lists (e.g. for jobs in the public administration).

Conclusion. As summarized by Table 3 the approach shows intriguing advantages in data exchange, split of responsibilities between central and local authorities and offers flexible handling of semi-structured data.

The main benefit is not just simplifcation but also the improved exchange between actors, better customisation which directly causes a strong negative: management of data is more complex and technology change is too big. Part of the issue is that the same results could be obtained through improved RDBMS/Web approach.

It as a step ahead of XML data exchange: RDF will allow not only to exchange data but also to directly search and process data from third party applications. However, the approach requires a jump in a new technology and requires high training costs. And although the application brings valuable advantages, it is perceived as not yet mature. This makes it less realistic for operations in near future. It might be of interest for a more research-oriented pilot project.

4 Application Model

The cases presented show a common model which can be seen as the next, application, layer over the basic RDF Engine layer [12]. We can present the basic engine layer as a component dedicated to ingestion/update of RDF descriptors and fast, bullet search/navigation over the stored triplets. Looking at our real-life examples, the following basic components are needed to create a complete system:

1. *Compiler-gatherer system* which is able to collect RDF descriptors from controlled/cooperating sites, from external systems publishing RDF data or data from legacy applications (e.g. SQL database). Some of the gathering (or harvesting-Scutter) applications are described in [19,20,21]. The whole process of collecting and compiling data is more complex. Here we limit ourselves to feedback after building the first applications. For any real-life application, this must be optimized to reduce network traffic and engine ingestion load, e.g. by gathering only newly modified or expired resource descriptions only. The gathering system should be able to pull descriptors when expired, or to accept data pushed by an external unit (e.g. a robot collecting data and parsing them into RDF) and to immediately provide RDF data parsed on-the-fly and used for an on-going query. Gathering should be based on a protocol(s) which will simplify exchange of RDFs among different applications/providers.
2. An *Optimiser* component which is able to ignore/eliminate obsolete triplets in the engine and require new compiling when needed. This is more an internal component of the engine, but it needs information about the expiration of resources from the external world.
3. An optional application layer which uses the RDQL interface offered by the engine and provides intermediate level processing which may be used by applications. As an alternative, applications which need strong customisation

can directly use RDQL. A first example of this may be the sorting of data which is not covered today by RDQL.

4.1 Search Domain

In a more general model, complex applications may need to limit the search scope to a domain . More important, a domain must be kept up-to-date, i.e. checked for expiration by the optimiser and refreshed by compiler-gatherer. The RDFStore implementation today supports the concept of provenance as an orthogonal dimension to stored triplets [12]. To work efficiently, this context information will need to contain some time information (like TTL, date of acquisition, etc.) that is needed to check and optimise RDF compilation.

The protocol adopted for defining the domain cases described uses a simple mother-index solution. Mother-index is the starting point of the domain definition and it is defined by an URI. This top-level index contains information for the search engine and/or information for the gatherer where to start looking for RDF descriptors to collect. The mother-index contains a list of items which might be: (i) An URI of a sub-domain mother-index, (ii) An URI of a RDF description itself together with additional timing information, (iii) URI of a gateway where RDF(s) can be created on-the-fly from data in RDBMS and (iv)URI of an instance of the RDFStore engine which might accept the query and return results. Once defined, the domain can be used as part of the pattern to limit the scope of the search.

4.2 Optimiser

This component is introduced here as a missing part of the puzzle. The optimiser is still a vague definition for the next generation of RDFStore engines. A few functions are identified today, more will be added following requirements from on-going application development:

1. Check the context and mask/remove/garbage collect obsolete triplets.
2. Track the provenance of obsolete triplets and request from the compiler-gatherer to update information by checking original descriptors.
3. Perform optimization of indices and internal structures.

4.3 RDQL, SQL, and Application Layer

RDQL is defined to support a model where little if any a priori knowledge of original data is known at run time. This leads to a simple query language which offers basic functionality only and attempt to optimize as much as possible the performances.

Considering the type of applications searching and browsing the Web we believe this is a good approach. However, trying RDQL on a few real-life applications has shown a few limitations:

1. The current version of RDQL does not support empty matches. One can solve this by forcing, in RDF, empty property values, but this conflicts with the desire for basic separation from the original data structures. A modification of the syntax will help in building real-life applications without influencing the language basics.
2. Similar to this is the problem of OR-ing data in the search pattern. As defined today, the pattern allows only the conjunctive AND match criteria. As soon as we move to real applications, some kind of OR-ing is needed. Today, OR-ing is done on application level. This may be costly but, and it may lead to complex application code. Two solutions are possible here: allow OR in the search pattern of RDQL or build a higher-level layer to support these functions just below application level.
3. Free text supported search today is very primitive: only a simple word match is only. Considering the importance of free-text data (the easiest to parse and used in all cases when a controlled list is not available), a better model is needed. Alternatives could be to keep RDQL simple and do more sophisticated work in a higher, application level. Due to the strong tie with the way words are indexed and saved, it might become too costly in terms of time and memory use.
4. Sorting of hits according to some criteria is another interesting point. This is used in all applications we tried. In case of small number of matches, such is not a problem, sorting can easily be done by the application after getting the results back an RDQL query. Small means today even hundreds of thousands if not millions of hits. The problem occurs when a large number of nodes is hit. In this case, sorting outside the engine will become unacceptable. It is probably premature to discuss sorting here, but some way to instruct the RDF engine about the way results are ordered (especially if cursors or partial result returns are required) will be needed. This might be even an approximate ordering but it can help when huge amount of data is retrieved.
5. A point for future development is also related to special comparisons which might be needed (e.g. geographical position). As part of the next generation, we are thinking of ways to push low-level (i.e. inside the engine) special comparison functions.

These points also offer a first comparison with SQL. An important aspect is that RQDL, as well as the whole model, does not provide any form of the UPDATE statement. There is no way to change the RDF data, we can only search and browse through the graph. More than a *Web Database* model ours can be called a *Web Library*.

As a practical aspect, it was interesting to note that people with SQL experience and understanding of RDF were immediately able to write RDQL queries. This was straightforward. It was much more complex to understand and prepare good RDF descriptions.

5 Conclusions

After these first real application experiences, we believe that RDFBusiness is possible. It first needs to be defined well and then explained better to customers.

Unified Web management seems the most mature area of applications. This is easy to build, non-invasive and does not interfere with the providers organization. These solutions can significantly improve the usability of a portal with limited investment. Aggregation of external sources seems more complex as a business, mainly because of the business model and because of cost of RDFying external, non-cooperating sources. However, this is again a form of unified Web management and, using a proper model of business, we expect that it might become an interesting business area. The approach of *Flavouring the results* considering user ratings of sources at run-time, all done using RDF, is another characteristic which can be useful in more complex analyses or intelligence applications.

In both cases, critical mass was not reached. To explain the full potential of the Semantic Web we need different information RDFed. *The business will explode when we will be able to show how with few instructions we can link completely different data.*

Semantic Web as the next step in federated/distributed databases is probably an over-challenging area for business. However, there are some intriguing aspects to it. First, RDF is be a step ahead of XML as lingua franca for information interchange between database applications: XML+RDF annotated data will be immediately accessible by any application and can simplify and speed up the information life cycle. We are proposing in this area a pilot project aimed at governmental - public administration use in which the advantages of Semantic Web technologies can be evaluated.

For all applications, we believe RDFying is a long-term investment, however one which will generally have immediate payback which can make the initial application already of enough value to make the implementation process a win-win proposition. Once annotated in this way, data will be usable by different applications in forthcoming years thus safeguarding the investment for a long period of time. Also as RDF is gradual and allows multiple descriptions-views it is certainly not an all-or-nothing proposition commonly seen in most document-management solutions or knowledge/data mining applications. This is a promising path to take.

To make the applications really work, we need to consider context and provenance as well as dynamic aspects: many of the resources we are dealing with change frequently during the lifetime of the application. This and the application level processing are part of a more complex model which will crystallise better in the Next Generation of RDFStore [12] at it keeps up with the new application developed for our customers.

References

1. T Berners-Lee, J Hendler and O Lassila, The Semantic Web, Scientific American, May 2001, http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html
2. T Berners-Lee: Semantic Web road Map, September 1998 http://www.w3.org/DesignIssues/Semantic.html
3. D Beckett: RDF/XML Syntax Specification (Revised) W3C Working Draft 23 January 2003 http://www.w3.org/TR/rdf-syntax-grammar/
4. O Lassila, RR Swick: Resource Description Framework (RDF) Model Syntax and Specification W3C Recommendation 22 February 1999 http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
5. J Grant, D Beckett: RDF Test Cases W3C Working Draft 23 January 2003 http://www.w3.org/TR/rdf-testcases/
6. SA Dobson: Lighweight Datatabases, WWW95 Conference, www.igd.fhg.de/archive/1995_www95/proceedings/papers/54/darm.html
7. A Levy: Putting Semi-structured Data to Practice, http://www.cs.washington.edu/homes/alon/cikm98.ppt
8. D Florescu, J Freira: "Querying the Web", http://www.pbnet.com.br/openline/cefet/tutoriais_sbbd_english.htm
9. L Wall: from the Interview with Tim O'Reilly around 1990, http://technetcast.ddj.com/tnc_program.html?program_id=52
10. ODISSEO: Open Distributed Information Services for Earth Observation, http://odisseo.esrin.esa.it/
11. OpenGIS: Geography Markup Language (GML) v1.0 and v2.0 http://opengis.net/gml/00-029/GML.html http://opengis.net/gml/01-029/GML2.html
12. A Reggiori, DW van Gulik, Z Bjelogrlic: Indexing and retrieving Semantic Web resources: the RDFStore model, Submitted to ISWC2003 Conference
13. L Miller, A Seaborne, A Reggiori: Three Implementations of SquishQL, a Simple RDF Query Language, www-uk.hpl.hp.com/people/afs/Papers/ISWC%202002%20-%20SquishQL.htm
14. Opengis Web map Server Specification, http://www.opengis.org/pressrm/summaries/20010219.TS.WMS.pdf
15. http://www.searchenginewatch.com/sereport/article.php/2175281
16. National-wide system developed by Italia Lavoro, Rome, Italy, http://www.italialavoro.it/
17. ETT: Regional systems, developed by ETT, Genova, Italy, http://www.ettdemo.com/domino/
18. W3C: Relational Databases on the Semantic Web, http://www.w3.org/DesignIssues/RDB-RDF.html
19. D Brickley: ScutterSpec http://rdfWeb.org/topic/ScutterSpec
20. All your FOAF pm module, http://frot.org/rdfweb/ayf.html
21. K Apsitis, RDF Crawler, http://ontobroker.semanticweb.org/rdfcrawl/
22. T. Berners-Lee, R. Fielding, L. Masinter August 1998 Uniform Resource Identifiers (URI): Generic Syntax http://www.ietf.org/rfc/rfc2396.txt
23. DC: Dublin Core Metadata Element Set, Version 1.1 http://dublincore.org/documents/dces/ and DCMI Metadata Terms http://dublincore.org/documents/2003/03/04/dcmi-terms/
24. Dan Brickley, R.V. Guha: RDF Vocabulary Description Language 1.0: RDF Schema W3C Working Draft