RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources

Richard Vdovjak, Geert-Jan Houben
Eindhoven University of Technology
Eindhoven, The Netherlands
{r.vdovjak, g.j.houben}@tue.nl

Abstract. The proposed integration architecture aims at exploiting data semantics in order to provide a coherent and meaningful (with respect to a given conceptual model) view of the integrated heterogeneous information sources. The architecture is split into five separate layers to assure modularization, providing description, requirements, and interfaces for each. It favors the lazy retrieval paradigm over the data warehousing approach. The novelty of the architecture lies in the combination of semantic and on-demand driven retrieval. This line of attack offers several advantages but brings also challenges, both of which we discuss with respect to RDF, the architecture's underlying model.

1 Introduction, Background, and Related Work

With the vast expansion of the World Wide Web during the last few years the integration of heterogeneous information sources has become a hot topic. A solution to this integration problem allows for the design of applications that provide a uniform access to data obtainable from different sources available through the Web. In this paper we address an architecture that combines issues regarding *on-demand retrieval* and *semantic metadata*.

1.1 On-demand Retrieval

In principle there are two paradigms for information integration: *data warehousing* and *on-demand retrieval*.

In the data warehousing (*eager*) approach all necessary data is collected in a central repository before a user query is issued; this however, brings consistency and scalability problems.

The on-demand driven (*lazy*) approach collects the data from the integrated sources dynamically during query evaluation. The MIX project [1], for example, implements a (virtual) XML view integration architecture, with a lazy approach to evaluation of queries in an XML query language specifically designed for this purpose.

1.2 Semantic Integration

XML[1] in general, has become an enormous success and is widely accepted as a standard means for serializing (semi)structured data. However, with the advent of the Semantic Web

[1] http://www.w3.org/TR/REC-xml

[2] where the data is expected to be machine readable, i.e. not just targeted at interpretation by humans, XML shows some limitations. As stated in [3] XML's (DTD's) major limitation is that it just describes grammars. In other words, the author of an XML document has the freedom to define and use tags, attributes, and other language primitives in an arbitrary way, assigning them different semantics to describe the conceptual domain model he has in mind. Since XML does not impose rules for such a description and there are many ways how to denote semantically equivalent things, it becomes hard to reconstruct the semantic meaning from an XML document.

Some documents have associated with them what is known as metadata. *Descriptive* metadata describes fields which are external to the meaning of the document (e.g. author, date, genre, etc.). *Semantic* metadata characterizes the content of the document [4]. This second kind of metadata, when standardized, could be used in machine-processing to extract the semantics of data. RDF[2] together with RDFS[3] provide standard means to describe both descriptive and semantic metadata.

On top of RDF(S), using its primitives like *subClassOf* or *subPropertyOf*, ontology languages like OIL [5] are built. With these languages one can describe domain ontologies, by identifying hierarchies of concepts and relations together with axioms that can be used to derive new facts from existing ones. An ontology can thus be seen as a semantic interface for accessing heterogeneous information sources. This introduces a new, semantic-based generation of information integration architectures. Projects On2broker [6] and On-to-knowledge [7] are involved in building ontology based tools for knowledge management providing architectures for semantic integration. However, both projects, to our understanding, are using the eager approach.

1.3 Approach

Within the context of the (related) Hera project [8], which aims at the automatic generation of multimedia presentations for ad-hoc user queries from heterogeneous information sources, our goal is to design and implement an integration architecture based on semantic integration and using on-demand information retrieval. This approach offers several advantages but brings also challenges, both of which we discuss with respect to RDF, the architecture's underlying model.

2 Architecture

The main purpose our architecture should serve is to provide a semantically unified interface for querying (selected) heterogeneous information sources. We do not aim at merging all possible sources together providing a cumulated view of all attributes. We argue that such an approach offers a very weak semantics, where the understanding of the semantic structure of all integrated sources is effectively left up to the user who is asking the query.

In our architecture, an underlying domain model consisting of hierarchies of concepts, relations, and possibly axioms is assumed to exist. This conceptual model (CM) is maintained centrally (at the schema level) but it is dynamically populated with instances during the query resolution. The CM corresponds to an ontology and represents a semantic integration of the integrated data sources. It is described directly in RDF or RDF extended with some higher level ontology language. To create such a CM beforehand, ontology engineering tools (which are currently becoming available) could be used. The main advantage of having an underlying semantic model is that the way in which the data is structured (encoded) in the sources is transparent for the user, i.e. he can ask queries and interpret the results in terms of well-understood

[2] `http://www.w3.org/TR/REC-rdf-syntax/`

[3] `http://www.w3.org/TR/rdf-schema/`

concepts (opposed to XML views, where queries are expressed more in terms of structure rather than semantics).

As shown in figure 1 the architecture is divided into five separate layers; we address each of them in the following sections.

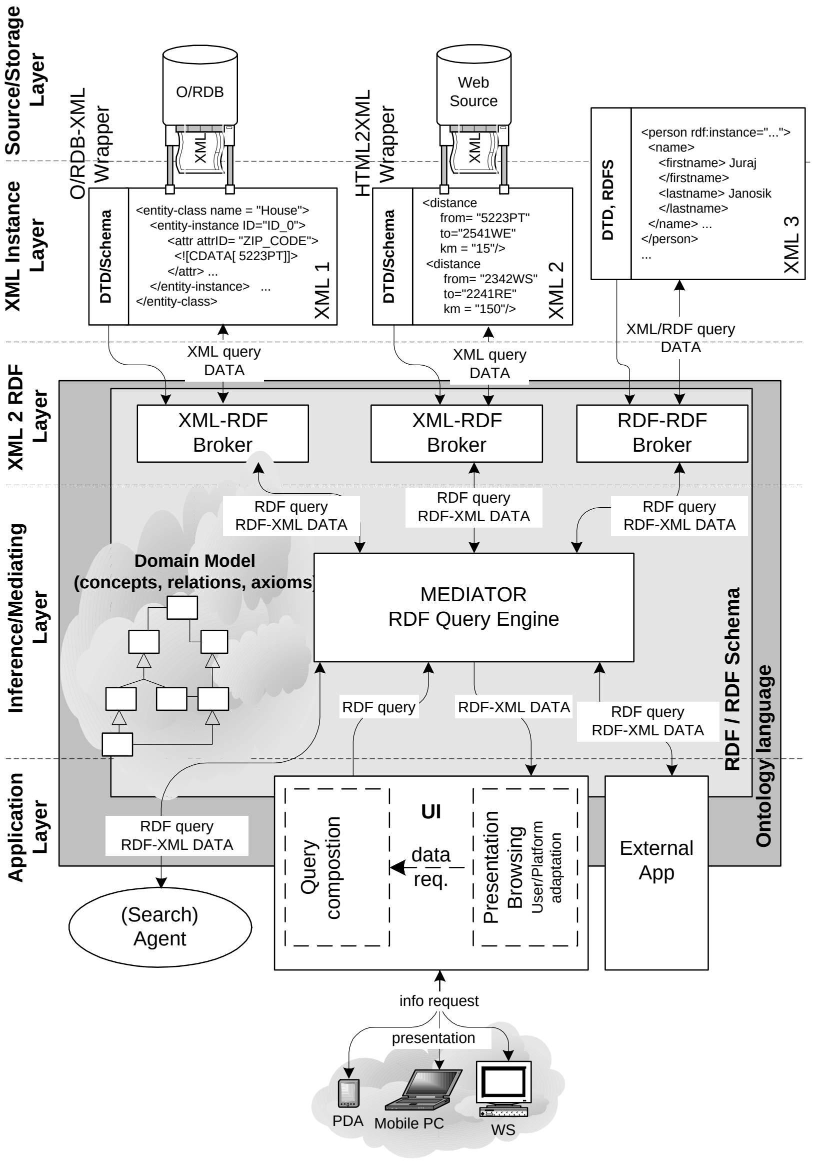


Figure 1: Architecture facilitating semantic integration of heterogeneous information sources

2.1 Source Layer

The source layer contains external data sources such as relational or object databases, HTML pages, XML repositories, or possibly RDF (ontology) based sources. Our target applications assume fairly general sources which can be distributed across the Web. The main requirement for sources is the ability to export their data in XML serialization. Some wrapping process may be needed to achieve this, but that is beyond the scope of this paper.

2.2 XML Instance Layer

This layer offers the serialized XML data that results from the previous layer. Sometimes (when no wrapping is required) the two layers can be considered as one. Note that assuming heterogeneity, we do not impose any particular structure the XML data sources should comply to. This allows us to leave the XML wrapping process for the source providers.

2.3 XML2RDF Layer

This layer consists of XML2RDF brokers which provide the bridge between the XML instance layer and the mediator. The designer tailors each XML2RDF broker to its source by specifying a mapping from XML sources to the underlying CM. This mapping is used by the XML2RDF broker while resolving a query coming from the mediator.

To establish a mapping from XML instances to the CM requires *(a)* to identify the schema i.e. to extract (parse) concepts that the source is describing and *(b)* to reconstruct their semantics in terms of the CM in other words, to relate the identified concepts to concepts from the CM.

In general, this is difficult to automate and an insight of the application designer is usually needed in both step *(a)* and *(b)*. The difficulty of *(a)* will vary based on the way the sources are encoded in XML. Sometimes the concepts of the source can be seen in its schema (DTD) as shown in figure 2 in the case of *Broker 1*. However, if the source encodes concepts as attribute values, the DTD is not enough and also the XML data has to be examined, as shown for *Broker 2*.

If the source is in RDF format (serialized in XML) or if the source's XML encoding adheres to some conventions (an implicit RDF interpretation of any XML file is proposed in [9]), step *(a)* can be to a large extent automated and tools can be employed to help the designer to accomplish the step *(b)* i.e. to relate concepts from one model to another.

Providing the actual response to a mediator's query requires the broker to poll the source for data and to create RDF statements, that is triplets (*subject*, *predicate*, *object*). Such triplets can be seen as instances or atomic facts and usually relate (*predicate*) data items (*subject*) to concepts (*object*) from the CM.

In figure 2 we provide an example of a small conceptual model, two XML sources and two mappings. The left part depicts an example of a CM together with its RDF encoding which describes the hierarchy of classes and some properties. Note that due to the space limitation we provide properties only of one class (Person). On the right we present two XML2RDF brokers, with their XML sources and the mapping rules that extract the relevant portions of information from the sources and relate it to the CM. These rules are specified in LMX[4][10], a language with a rather intuitive syntax where a rule consists of a left-hand side (interpreted as the *from* part) and the right-hand side (interpreted as the *to* part). The data which is to be transferred is specified by positioning variables denoted as *$x*. These are declared at the beginning as processing instructions to make the application aware of them.

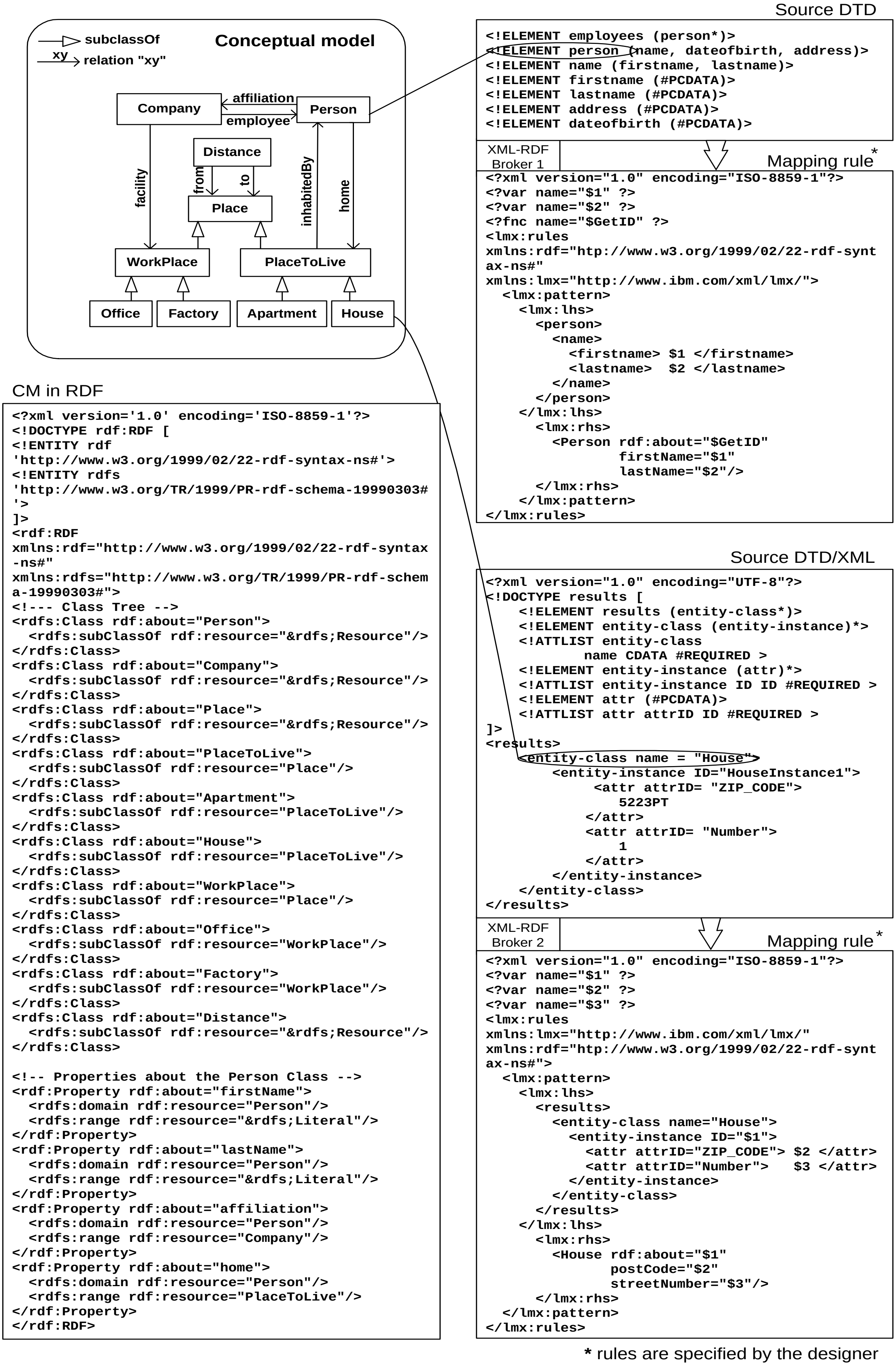
2.4 Inference and Mediating

The RDF Mediator is the central component of the architecture. It maintains the CM, provides query and inference services, and also the support for traversing the results.

The CM consists of a class (concept) hierarchy together with class properties, and a set of rules that correspond to axioms about classes or their properties. Hence by applying these rules on the set of facts which are retrieved, it is possible to infer new facts. The rules can be expressed in F-Logic [11]. For instance the following rule expresses that if a person *X* is affiliated with a company *Y*, *Y* considers *X* to be an employee.

$$\forall X, Y \quad Y[\mathit{employee} \twoheadrightarrow X] \leftarrow X[\mathit{affiliation} \twoheadrightarrow Y].$$

[4] Language for Mapping XML documents



CM in RDF

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf
'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY rdfs
'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#
'>
]>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax
-ns#"
xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schem
a-19990303#">
<!--- Class Tree -->
<rdfs:Class rdf:about="Person">
  <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="Company">
  <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="Place">
  <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="PlaceToLive">
  <rdfs:subClassOf rdf:resource="Place"/>
</rdfs:Class>
<rdfs:Class rdf:about="Apartment">
  <rdfs:subClassOf rdf:resource="PlaceToLive"/>
</rdfs:Class>
<rdfs:Class rdf:about="House">
  <rdfs:subClassOf rdf:resource="PlaceToLive"/>
</rdfs:Class>
<rdfs:Class rdf:about="WorkPlace">
  <rdfs:subClassOf rdf:resource="Place"/>
</rdfs:Class>
<rdfs:Class rdf:about="Office">
  <rdfs:subClassOf rdf:resource="WorkPlace"/>
</rdfs:Class>
<rdfs:Class rdf:about="Factory">
  <rdfs:subClassOf rdf:resource="WorkPlace"/>
</rdfs:Class>
<rdfs:Class rdf:about="Distance">
  <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>

<!-- Properties about the Person Class -->
<rdf:Property rdf:about="firstName">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="lastName">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="affiliation">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="Company"/>
</rdf:Property>
<rdf:Property rdf:about="home">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="PlaceToLive"/>
</rdf:Property>
</rdf:RDF>
```

Source DTD

```
<!ELEMENT employees (person*)>
<!ELEMENT person (name, dateofbirth, address)>
<!ELEMENT name (firstname, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT dateofbirth (#PCDATA)>
```

Mapping rule*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?var name="$1" ?>
<?var name="$2" ?>
<?fnc name="$GetID" ?>
<lmx:rules
xmlns:rdf="htp://www.w3.org/1999/02/22-rdf-synt
ax-ns#"
xmlns:lmx="http://www.ibm.com/xml/lmx/">
  <lmx:pattern>
    <lmx:lhs>
      <person>
        <name>
          <firstname> $1 </firstname>
          <lastname>  $2 </lastname>
        </name>
      </person>
    </lmx:lhs>
      <lmx:rhs>
        <Person rdf:about="$GetID"
                firstName="$1"
                lastName="$2"/>
      </lmx:rhs>
    </lmx:pattern>
</lmx:rules>
```

Source DTD/XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results [
    <!ELEMENT results (entity-class*)>
    <!ELEMENT entity-class (entity-instance)*>
    <!ATTLIST entity-class
            name CDATA #REQUIRED >
    <!ELEMENT entity-instance (attr)*>
    <!ATTLIST entity-instance ID ID #REQUIRED >
    <!ELEMENT attr (#PCDATA)>
    <!ATTLIST attr attrID ID #REQUIRED >
]>
<results>
    <entity-class name = "House">
        <entity-instance ID="HouseInstance1">
             <attr attrID= "ZIP_CODE">
                5223PT
            </attr>
            <attr attrID= "Number">
                1
            </attr>
        </entity-instance>
    </entity-class>
</results>
```

Mapping rule*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?var name="$1" ?>
<?var name="$2" ?>
<?var name="$3" ?>
<lmx:rules
xmlns:lmx="http://www.ibm.com/xml/lmx/"
xmlns:rdf="htp://www.w3.org/1999/02/22-rdf-synt
ax-ns#">
  <lmx:pattern>
    <lmx:lhs>
      <results>
        <entity-class name="House">
          <entity-instance ID="$1">
            <attr attrID="ZIP_CODE"> $2 </attr>
            <attr attrID="Number">   $3 </attr>
          </entity-instance>
        </entity-class>
      </results>
    </lmx:lhs>
      <lmx:rhs>
        <House rdf:about="$1"
               postCode="$2"
               streetNumber="$3"/>
      </lmx:rhs>
  </lmx:pattern>
</lmx:rules>
```

* rules are specified by the designer

Figure 2: Mapping source instances to the conceptual model

Note that maintaining one global CM (ontology) for all possible applications is not feasible (for scalability reasons). However, the distributed approach where one instance of the architecture (and thus one CM or ontology) serves as an input source for another architecture instance, brings scalability also in an environment like WWW.

The mediator contains an RDF parser[5], a query decomposition module, and a query engine, which uses for inferencing SiLRI[6] [12]. To support the traversal (and so the actual retrieval) of the results, the mediator also has to implement an analogy of the DOM[7] API, modified for the RDF data model (directed labelled graphs). After the mediator receives a query from the application layer it proceeds as follows.

First, it analyzes whether the query's resolution demands inference rules to be applied and if so, which are the facts that are needed to evaluate the inference rules. Note that the inference engine assumes that the facts are known beforehand, which is however, in contradiction with the on-demand retrieval approach. That's why the initial query must be enriched to retrieve also these facts that enable the inference engine to apply the rules.

Second, it decomposes the query into subqueries and distributes them among the brokers. The actual querying is triggered by a navigation request coming from the application layer.

Third, it collects the data from the brokers, applies possible inference rules, constructs the response and sends it to the application layer.

2.5 Application Layer

There are numerous applications that can take advantage of the semantically unified interface provided by the architecture. The types of applications can vary from search agents (machine processing) to hypermedia front-ends that guide a (human) user in query composition and CM exploration, and produce as a response to a query a full-featured hypermedia presentation [8, 13] supporting browsing and user/platform adaptation [14].

Another possible application is an instance of a similar architecture maintaining a different, yet similar CM, which could consider the first architecture instance as one of its data sources.

3 Conclusions

A solution to the problem of integrating heterogeneous information sources is needed in order to provide a uniform access to data gathered from different sources available through the Web. The proposed integration architecture combines semantic metadata with on-demand retrieval. It offers a semantic interface for (dynamic) access to heterogeneous information sources and also the possibility to use inference mechanisms for deriving new data, which was not (explicitly) provided by the integrated sources.

However, enabling inferencing while using on-demand retrieval introduces a possible bottleneck when the facts needed by the inference engine must be retrieved together with the requested data; here we see room for optimization and we will investigate this problem further. Currently, in the context of the HERA project, we are verifying our ideas with an implementation of the architecture's prototype.

References

[1] B. Ludscher, Y. Papakonstantinou, and P. Velikhov. A framework for navigation-driven lazy mediators. In *CM Workshop on the Web and Databases*, 1999.

[5] `http://www.w3.org/RDF/Implementations/SiRPAC/`

[6] Simple Logic-based RDF Interpreter (`http://www.ontoprise.de/co_silri.htm`)

[7] `http://www.w3.org/DOM/`

[2] Tim Berners-Lee and Mark Fischetti. *Weaving the web*, chapter Machines and the Web, pages 177 – 198. Harper, San Francisco, 1999.

[3] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of xml and rdf. *IEEE Expert*, 15(3), October 2000.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, editors. *Modern Information Retrieval*, chapter Text and Multimedia Languages and Properties, pages 142,143. ACM press, Adison Wesley, 1999.

[5] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In R. Dieng, editor, *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.

[6] D. Fensel, J. Angele, S. Decker, M. Erdmann, H. Schnurr, S. Staab, R. Studer, and A. Witt. On2broker: Semantic-Based Access to Information Sources at the WWW. In *World Conference on the WWW and Internet (WebNet 99)*, 1999.

[7] D. Fensel, F. Van Harmelen, M. Klein, H. Akkermans, J. Broekstra, C. Fluit, J. Van der Meer, H. Schnurr, R. Studer, J. Hughes, U. Krohn, J. Davies, R. Engels, B. Bremdal, F. Ygge, T. Lau, B. Novotny, U. Reimer, and I. Horrocks. On-to-knowledge: Ontology-based tools for knowledge management. In *eBusiness and eWork*, Madrid, October 2000.

[8] Geert-Jan Houben. HERA: Automatically Generating Hypermedia Front-Ends for Ad Hoc Data from Heterogeneous and Legacy Information Systems. In *Engineering Federated Information Systems*, pages 81 – 88. Aka and IOS Press, 2000.

[9] Sergey Melnik. Bridging the Gap between RDF and XML. Technical report, Stanford University, 1999. Available online from `http://WWW-DB.Stanford.EDU/˜melnik/rdf/fusion.html`.

[10] Hiroshi Maruyama, Kent Tamuran, Naohiko Uramoto, and Kento Tamura. *XML and Java*, chapter LMX: Sample Nontrivial Application, pages 97 – 142. Addison-Wesley, 1999.

[11] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1999.

[12] Stefan Decker, Dan Brickley, Janne Saarela, and Jürgen Angele. A query and Inference Services for RDF. In *QL'98 - The Query Languages Workshop*. W3C, 1998.

[13] Paul De Bra and Geert-Jan Houben. Automatic Hypermedia Generation for ad hoc Queries on Semi-Structured Data. In *ACM Digital Libraries*, pages 240 – 241. ACM, 2000.

[14] Paul De Bra, Peter Brusilovsky, and Geert-Jan Houben. Adapative Hypermedia: From Systems to Frameworks. *ACM Computing Surveys*, 31(4es), 1999.