Warehousing Web Resources with the WebContent Platform

Serge Abiteboul* Gaël de Chalendar† Ioana Manolescu* Emmanuel Pietriga*

Bernd Amann‡ Patrice Buche§ Juliette Dibie-Barthélemy§

Patrick Giroux¶ Bruno Grilhères¶ Benjamin Nguyen‖

ABSTRACT

We describe the WebContent platform for the management of content from the Web. The platform is based on a service-oriented architecture and Web standards (notably, Web services, XML and RDF). An enterprise service bus (following the JBI specification) and BEPL may be used to orchestrate service invocations. A peer-to-peer architecture may also be used to facilitate cooperation between independent partners as well as provide scaling.

We briefly describe services that were developed for supporting the main functions of the platform: acquisition, e.g., Web crawling, semantic enrichment, e.g., concept annotations, high-scale XML storage and querying (in a centralized or P2P architecture) and exploitation (including Web-based interfaces). Ontologies are pervasive in WebContent applications, supporting the description of the harvested and derived information as well as that of applications.

WebContent brings together a large number of groups from industry and academia. The core of the platform is open-source. A large toolkit of both open-source and commercial services is already available. WebContent is being tested on different Web surveillance applications. In the paper, we use a strategic watch application in aeronautics that has been developed for Airbus to illustrate various aspects of the platform. WebContent is now available for research and development outside the original group of participants.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering; H.3.4 [**Information Systems**]: Systems and Software; C2.4 [**Computer-communication Networks**]: Distributed Systems

General Terms

Design, Performance, Human Factors

Keywords

Information extraction, Distributed data management, Web services, Semantic Web, Peer-to-peer systems

*INRIA Saclay–Île-de-France and LRI, Univ. Paris XI, France
†CEA LIST, France
‡LIP 6 – Univ. Pierre et Marie Curie & CNRS, France
§INRA Met@risk/AgroParisTech, France
¶EADS DS/SDC/IPCC, France
‖Univ. Versailles Saint-Quentin-en-Yvelines & CNRS, France

WWW2009, April 20-24, 2009, Madrid, Spain.

1. INTRODUCTION

The main data sources available on the Internet today and even internally in companies are in very large part textual. Furthermore, the portions that are more controled (structured) are often heterogeneous, relying on different formats and different ontologies. As a consequence, the full exploitation of the vast quantity of available Web resources by end-users such as experts performing watch activities is difficult at best, and often impossible. Furthermore, the development of Web applications to assist them is extremely complex and often requires unavailable programming skills or budget. In this paper, we introduce the WebContent platform that has been developed to ease the development of Web content applications [44].

The platform warehouses content relying on a rich set of services. These services are used for acquiring data, cleaning them, storing and indexing, semantically enriching them (with machine-processable annotations), and exploiting them notably via Web-enabled visualization tools. The platform is based on the sharing of XML documents. An Enterprise Service Bus (ESB) and BPEL may be used to orchestrate service invocations. We also consider performing tasks in a peer-to-peer network.

Observe that WebContent is in the spirit of the Semantic Web, building of its technologies [8, 33] , notably stuctured data, linked and semantically-enriched in a machine-processable manner, exchanged between applications and services.

In this paper, we describe the various components of the platform. First, we introduce the WebContent data model that facilitates describing the content of Web documents that are relevant to a particular application. Next, we consider the core services of the platform, e.g., storage, and application services, e.g., ontology-based classification. It should be observed that a large part of the effort in the WebContent project comes from partners tailoring specific tools to the WebContent context.

Ontologies are pervasive in WebContent applications as they support the description of the harvested and derived information as well as that of the applications. This knowledge is stored in an XML database and efficient and scaling query processing is a central aspect of the platform. Besides data and knowledge management, the tools that equip the platform cover a wide range of domains, from linguistic analysis to machine learning and vizualization. We will discuss some of them and describe guidelines for developing WebContent applications.

The WebContent platform was designed to facilitate Web content processing at large. Since this is too broad a target to fully grasp, we chose to focus our attention first on Web watch applications. The platform is now up and running. We are testing its functionality

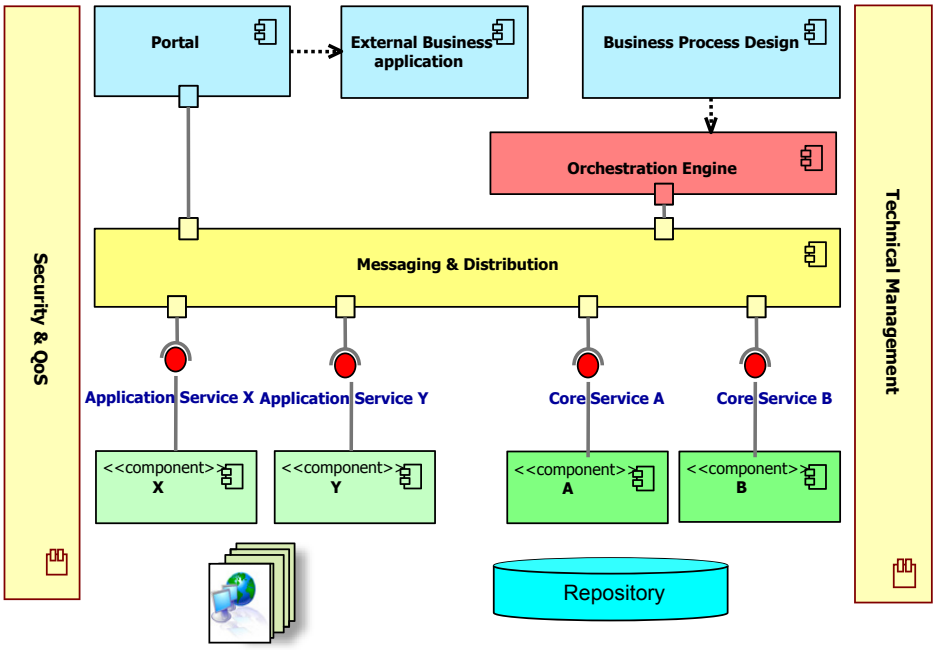

Figure 1: Architecture of a WebContent application.

with a number of real-case applications: (i) technological watch in the domain of food risk analysis, (ii) strategic watch in aeronautics, (iii) intelligence watch, and (iv) seismic watch. The second one that has been developed for Airbus is used throughout the paper to illustrate the various aspects of the platform with a real-world use case.

WebContent is a project from the Agence Nationale de la Recherche bringing together a large number of groups from industry and academia. The platform has been a testbed for the integration of a number of XML, Web service and Semantic Web technologies. It has also been used for validating prototypes issued from several research projects, including peer-to-peer (P2P) management of Semantic Web data, ontology merging tools, machine learning algorithms for structured content extraction, and Web-enabled data visualization widgets.

The core of the platform is open-source. A large toolkit of both open-source and commercial services is already available. WebContent is now available for R&D outside the original group of participants. See `http://www.webcontent.fr`

The paper is organized as follows. Section 2 gives an overview of the platform. Section 3 provides a detailed description of its architecture and data model. Section 4 discusses related work. Section 5 concludes the paper.

2. PLATFORM OVERVIEW

As mentioned in the introduction, the platform has primarily been tested with Web watch applications, and we use one such application as an illustration in this paper. Web-based watch applications are facing problems such as acquiring all kinds of documents in various formats, analyzing them and extracting knowledge relevant to the task at hand, performing a synthesis of the derived information and alerting users of relevant events. Watch applications are thus very typical of a wide range of Web applications. Beyond, Web processing applications can benefit from WebContent in a wide range of other domains, including economic intelligence, open source intelligence, Enterprise Information Portals (EIP), Content Management Systems (CMS), (multimedia) archiving, or knowledge management at large.

We describe next the platform functionalities and present the Airbus application.

2.1 Functionalities

Watch applications require the use of a wide variety of information processing modules. While building our original functional map, we identified thirty such modules, including: data collection, acquisition, storage, indexing, transformation, normalization, description, ontology-based annotation, visualization, structuring, ordering, dissemination, sharing. Robust and efficient solutions are available for some of these modules, whereas others are only partially solved in research prototypes. Solutions come from various vendors and research laboratories, handle documents and metadata in various formats, and can take different forms: stand-alone applications, libraries, and Web applications. Integration is thus a serious problem. To overcome this issue, the WebContent platform is built around a *Semantically-Driven Service-Oriented Architecture* that:

- provides an open and extensible model for data sharing and an (extensible) set of service definitions based on that model;
- provides application building guidelines and open-source development toolkits in Java and C++;
- is designed with scalability in mind; and
- is fully compliant with the main Web standards.

Figure 1 illustrates the overall architecture of a WebContent application. Each application is built around a set of services. Services are currently provided as components supplied by the nineteen partners involved in the project. The platform presents a service integration infrastructure: each service is thoroughly defined and can exchange data with all other services in a format defined and adapted to the specific type of targeted applications. WebContent applications are typically designed and implemented as Web

portal applications, using portlets to build rich and configurable front-end graphical user interfaces.

Each service is independent from the others. It is defined by a contract specifying its interface and conditions of use; it is normalized, currently through a process internal to the WebContent consortium. The intention is to make this process open to a broader community. Similar components, providing the same kind of services, possibly based on different technologies, can be implemented by different suppliers. We consider two broad classes of services: core services are a common base for applications development; application services are used to realize functions related to actual information extraction and processing.

To accomplish a particular business objective through a complex process, several atomic services may have to be combined. For that, we use a WS-BPEL (Business Process Execution Language) [37] engine. BPEL enables the description of a complex business process using standard operations such as service invocations, conditional blocks, loops, variable affectations, etc. In particular, service calls may be "pipelined" with a service response directly used as input to the next service. High flexibility is achieved with multi-level service composition and the capability to use runtime endpoint selection. A runnable process may be specified using a graphical BPEL editor. Adding, updating or removing a service implementation does not have any functional impact on the whole business process. The platform only needs to update its composite process using BPEL capability to dynamically assign the endpoint reference before invocation.

The WebContent platform uses Web service and Semantic Web technologies to provide a set of application services to collect, process and exploit structured and unstructured content; a set of core services for the management of available resources and in particular storage and querying; a middleware and a connector model enabling services to communicate and insuring their *technical* interoperability; a reference data model to normalize exchanges between services and insure their *semantic* interoperability; and a suite of tools to realize, integrate, deploy, orchestrate and test the services.

2.2 The Airbus application

In this section, we briefly describe one of the first four applications built using the WebContent platform. This application is developed for the Documentation Department of Airbus, an EADS company manufacturing aircrafts. It is targeting users performing technical and economic watch activities in aeronautics. Its objective is to collect data from public sources such as headline news, press releases, economical data and technical notes. The collected data have to involve Airbus, its competitors or its subcontractors.

In this use case, two different kinds of users interact with the application: *administrators* and *watchers* (Figure 2). Documents are collected from the Internet and information relevant to the domain and tasks is extracted from them. This information is stored in a knowledge repository where it can be retrieved and exploited. Only administrators are involved in these preprocessing and analysis phases that are performed automatically as service orchestrations. Administrators select the sources to be crawled and configure the services to be invoked. When documents have been processed, analyzed and indexed, the watcher can exploit the extracted knowledge by asking queries to the knowledge repository.

The knowledge that is extracted by the platform is about the products and companies that are involved in the engineering process, the events that occur in the aeronautic business world, and problems detected such as faulty parts, etc. One role of the analysis is to detect relationships between concepts, e.g., `the Airbus A350` *uses* `the V35 engine` *produced by* `Roll's Royce`.

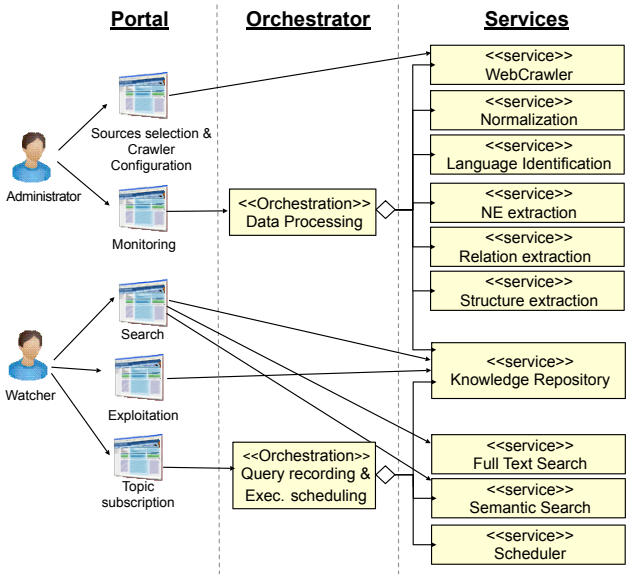

Figure 2: Architecture of the Airbus application for EADS

The watcher must be able to identify real and misleading information regarding Airbus and annotate the knowledge base, e.g., validate or invalidate. To develop an adequate tool to support watchers in their tasks, we had to use a large panel of services from the WebContent platform, as detailed further.

The scenario begins with the collection phase that involves the Web crawling service. This first service is based on Exalead's crawling engine [16] wrapped in a WebContent compatible interface. Crawled documents are then converted into the WebContent exchange format using the normalization service in order to be processable by other services: the original documents are converted into WebContent Media Units (Figure 3). These media units are semantically annotated using an OWL [26] ontology pertaining to aeronautics and designed for this application. Several annotation services are invoked sequentially, each of them gradually enriching the media units. Finally, the annotated media units are sent to two distinct indexing services: a full-text indexer and a semantic indexer. The semantic index takes into account, knowledge that has been previously extracted by annotation services (i.e., concepts and relationships between them defined in terms of the ontology). The watcher can then access the documents through two modes:

- An interactive search mode to define textual or semantic queries and then browses through results.
- A monitoring mode to specify feeds about specific topics, and be notified (based on a specified frequency) when alerts are detected.

Sophisticate visualization widgets are provided for browsing resulting documents, including semantic network and timeline (Figure 5) features. In both modes, the documents are displayed with highlighted text emphasizing annotations.

3. DETAILED PLATFORM DESCRIPTION

3.1 The WebContent model

To guarantee the full interoperability of services, it does not suffice to normalize protocols and service interfaces. It is also nec-

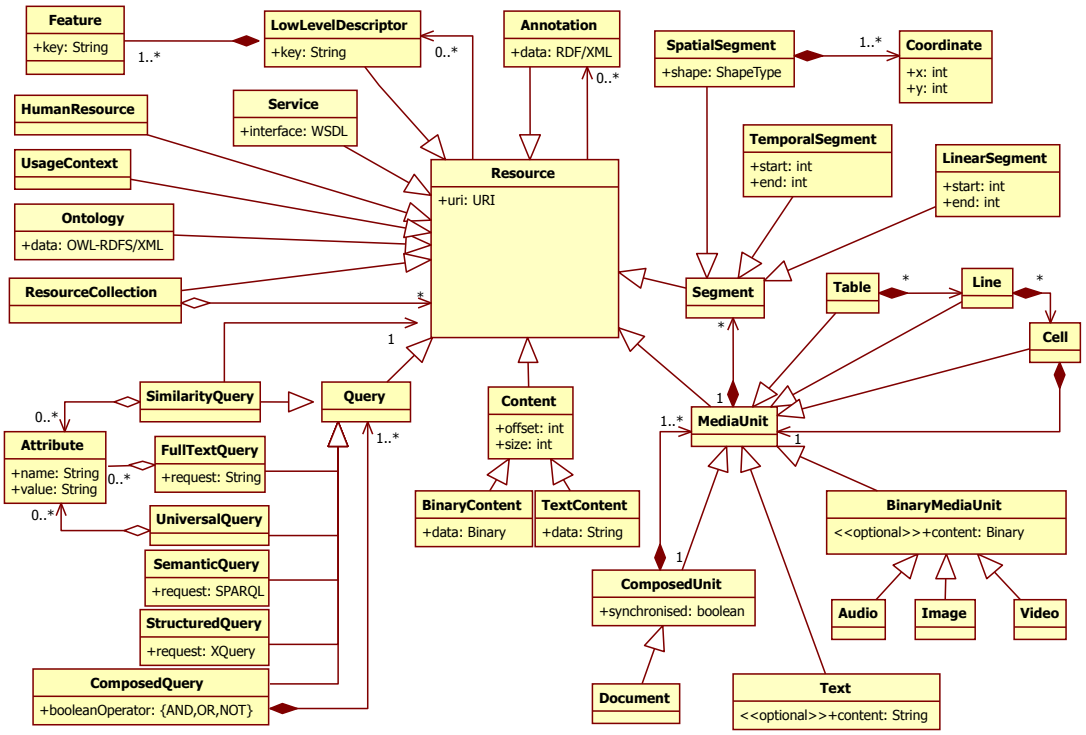

Figure 3: Elements of the WebContent model.

essary to define the structure and semantics of data shared or exchanged by services. The WebContent data model must be able to describe all kinds of source documents that go through the textual information extraction process and be extensible to other kinds of media. To reach this goal, all nineteen partners of the initial project have contributed to the elaboration of the model by confronting its evolutions to the specific needs of the provided services and to those of the built applications. The model has since been extended through other R&D projects, and other academic and industrial organizations are now using it.

The model defines an XML format. WebContent resources can thus be stored into, and queried via, an XML repository. Their structures closely matches that of the associated source documents. Annotations are expressed in RDF [25] and serialized in RDF/XML for easy handling and association of semantic data with each unit of the document.

Figure 3 shows a UML class diagram displaying the elements of the model. Each class inherits from the `Resource` class and is thus identified by a URI. RDF annotations use this URI to refer to specific units in the documents. Documents are composed of `MediaUnit`s. Recursive structures can be captured using `ComposedMediaUnit`s. Other elements include `Text` for leaf textual segments and elements to describe tables (with a terminology inspired by XHTML). Multimedia documents may also be represented using different types of `BinaryMediaUnit`s A mechanism allowing the asynchronous transfer of binary data between services is supported. The different kinds of `Segment`s are used to annotate parts of media units.

The combination of data and annotations in the data model allows for a better decoupling between services, each service receiving all the data it needs to do its processing in a single call. It also closely follows the way data are processed in the applications. This model does not impose any restriction on the way the data is effectively stored and processed by each service. In fact, some WebContent applications use an XML/RDF store and SPARQL [31] to query the data, while others use the annotations as pure metadata attached to documents in a full-text index. For binary or large data, annotations can be received separately.

An XML implementation of the model, examples, documentation and software utilities is available to the community in a free development kit [45].

3.2 WebContent deployment architectures

The WebContent platform can be deployed in different architectures depending on the constraints and requirements of the specific applications and organizations considered.

The first architecture is built around an *Enterprise Service Bus*. In this context, the warehouse's Web services are connected to the bus, and the bus manages their interaction (or composition) according to a high-level imperative specification expressed in a subset of BPEL [37]. The bus is responsible for routing messages to and from the Web services, and for ensuring *orchestration*, i.e., that interactions between services take place in the desired order. The bus-centered architecture is well suited to applications in which the Web services to be used are precisely identified, and interconnected via the bus, prior to the actual activation of the application.

To facilitate service interoperability with controlled quality of service, the platform includes an Enterprise Service Bus that allows the integration of heterogeneous components as service providers. We use Java Business Integration (JBI) [23], a commonly used specification to implement ESBs. It makes it possible to compose heterogeneous components using normalized deployment, requesting, and messaging paradigms. JBI defines two classes of internal components for an ESB:

- *Service Engines* that implement core services such as XSLT transformations, orchestration, scheduling, load balancing, etc.

- *Binding Components (BC)* that are used to interoperate with external components using a specific protocol. For example, a SOAP BC, a mail BC, an XMPP BC, a CORBA BC, etc.

The ESB can thus be extended to support any messaging protocol and implement a gateway to other platforms or middleware technologies. Currently, the SOAP [18] binding component is commonly used in WebContent applications since integrated components are Web Services compatible.

Each supported service is exposed on the bus through an endpoint that a consumer can use to reach it without knowing where it is hosted. The current version of the platform uses the open-source ESB "PetALS" [15]. But because of the compliance to JBI, other JBI buses may be prefered.

A second deployment architecture makes it possible to exploit WebContent services in a *peer-to-peer* (P2P) network. In this context, following the standard Web service model, one or several endpoints (on concrete peers) may exist for each Web service. Specific to the P2P architecture, however, is the fact that some Web services may be implemented *cooperatively* by a set of network peers. Moreover, the exact set of peers collaborating to answer a given service call may be determined dynamically at runtime, i.e., once for each service call that is made, based on the P2P network. This requires much less centralization and is more in the style of Web service choreography [46], where one specifies *how* a set of services should interact in order to implement together a given functionality.

The P2P deployment architecture is more complex but different aspects of the application may make it preferable to the bus-oriented one. First, several peers may together achieve more scaling up than a single one, both in terms of the quantity of information that is managed and of the processing on it. Then, determining at runtime the peer(s) that provide a given functionality allows balancing the load among the peers. Finally and most importantly, each participant in the warehouse may, in a P2P context, keep control over its own information and accommodate a particular *viewpoint*, e.g., prefer a particular classification module. This last aspect is particularly useful when the application is a collaboration between independent entities that are willing to combine resources but would not accept changing their operation mode to align it to a common one.

Observe that *hybrid* architectures may also be envisioned. For instance, one can use a network of peers to efficiently implement some Web services (to scaleup), then interact with this network via a fixed unique endpoint, connected to other services via a bus. The choice of an architecture for a given application should take into account the application needs in term of quality of service, control, scalability, and load balancing, as previously sketched.

3.3 Core Services

We next discuss the core services of the WebContent platform. They are grouped into four main categories, namely, catalog, storage, query processing, and semantic query reformulation.

Service catalog. The WebContent service catalog [7] fulfills two main roles in the WebContent platform. First, it provides a standard *catalog service*, in the style of UDDI. It allows storing and managing WebContent processes as well as their instantiations in form of workflows. Its second role, beyond standard UDDI-style catalogs, is that a *workflow selection and tuning assistant*. Before being executed, WebContent process may have to be instantiated into a workflow by choosing a component for each service it uses. The choices may lead to workflows with results of varying quality. For instance, some search or translation components may produce better results for certain languages than others, and some formatting services are better adapted to certain document formats (plain text, HTML, PDF, ...) than others. The instantiation of workflows may then become a tedious task, since it theoretically implies testing an exponential number of combinations of appropriate components. The WebContent service catalog assists workflow designers when instantiating WebContent based on the quality of previous workflow executions and specific user-defined requirements.

To be highly integrable, extensible and reusable, the service catalog follows the WebContent architecture and is divided into three loosely-coupled layers: storage, selection and presentation. The *service-storage* module uses an XML repository. (It is following the WebContent interface for storage, so can be implemented for instance on top of the *MonetDB system* [39] or on the P2P storage service of WebContent.) The most important module is the *service-selection* module. It exposes a public interface proposing several methods for interacting with the catalog, i.e., adding, editing and deleting processes, services, components and workflows as well as the recommendation features previously mentioned. Finally, the user interacts with the catalog through Web portlets deployed in a Web portal.

Storage services. The platform defines an interface for a *storage* service and consequently a *query* service, to access the data that is stored. These interfaces are generic, and the actual physical storage is not specified in advance. To illustrate this flexibility, WebContent already releases several instances of these services: A P2P storage and query services (developed by members of the consortium) and also such services developed on two different centralized database engines, MonetDB [39] and MS SQL Server (2008) [42].

Centralized storage service: For storage on a single machine, we can use either MonetDB or MS SQL Server. In both cases, the WebContent documents are stored in their native XML format. Querying these documents is possible using the full capacities of query languages for XML. An issue when running such queries is that they may return results of any type. We have also developed a particular WebContent query interface that only allows queries returning WebContent resources. Results of such queries may be placed in the warehouse.

A main role of each *storage service* is the indexing of the documents it stores, so that XML queries may then be performed efficiently.

P2P storage service: This service complies with the same interface as the centralized one. From a functional viewpoint, it is totally transparent for the user of the WebContent platform to switch from a centralized storage service to the P2P one. With the P2P storage service, many peers may participate to the same logical storage service. In this distributed XML database, the exact location of a piece of data is transparent to the user. The P2P storage service also supports indexing facilities. A *DHT service* is implemented on top of a distributed hash table (or DHT, for short). The DHT, a distributed software running on all peers, provides the connectivity of the network. It assigns unique identifiers to peers and allows them to easily join and leave the network[1]. Indexing is supported using a distributed data structure based on the simple abstraction of (key,value) pairs (with two services, namely *put(k,v)* and *get(k)*). Without delving into the details, the DHT stores all values associated to a given key k, on a particular peer in charge of that key.

Different DHTs may have different algorithmic properties, interesting from different performance viewpoints. For instance, a

[1]Remember that in a hybrid architecture, all participants need not be part of the P2P network.

DHT may guarantee that two keys k_1 and k_2, "close" by some distance measure, are managed by peers that are "close" in some sense. To take advantage of the good properties of distinct DHTs, several DHTs may coexist in a WebContent deployment architecture. Thus, a peer p belonging to the DHTs $dht_1, dht_2, \ldots$ is an endpoint for the services $join_1, leave_1, put_1, get_1$, but also for $join_2, leave_2, put_2, get_2$ etc. We have successfully integrated so far two DHTs [1]: FreePastry [32] from MIT, including our own extensions for robust scalable XML indexing [3]; and PathFinder [14], specially tuned to support interval search queries (which FreePastry does not support).

Query services. WebContent resources are serialized in XML and their exploitation relies on advanced XML query processing capabilities. The XML Query processor is of course closely tied to the store service that is used. MonetDB and MS SQL Server both propose an implementation of XQuery. We do not detail them here since we simply packaged them as WebContent services. In the case of the P2P storage system, we have integrated an *XQuery algebraic analyzer service*, based on the TGV XQuery optimizer [35]. When invoked with an XQuery query as input, this service brings the query to an algebraic form that allows reasoning on the query to identify the largest subsets that can be handled by the DHT index services. XML querying in P2P also uses a distributed query optimizer. We use a P2P optimizer service, namely OptimAX [5], that we developed. The distributed query plans we consider are specified in ActiveXML [36], i.e., XML document including calls to Web services. The optimizer's goal is (in a standard manner) to reduce the total work involved in the processing of the query.

Note that although we focus on XML queries, the WebContent platform offers a generic query interface that could in principle be used with other kinds of data.

Semantic Query reformulation service. This service allows transforming semantic queries based on the particular *viewpoint* of a specific peer. Different peers may use different ontologies to model the (aspects of an) application that they are interested in. In our example Airbus application, peer p_1 may be interested in Europe sales of Airbus and its competitors, while p_2 may focus on press reports concerning Airbus. The useful ontological concepts for p_1 may thus be *AirplaneManufacturer*, *salesByCountry* etc., whereas p_2 would use concepts such as *newsAgency*, *publishedSaleFigures*, *newsSource*. If p_1 is aware of the existence of p_2, p_1 may create a *mapping* specifying, e.g., how p_2's concept *publishedSaleFigures* relates to p_1's *salesByCountry*. Based on this mapping, a semantic query posed at p_1 and involving *salesByCountry* can be reformulated in terms of *publishedSaleFigures* and sent to p_2. This mechanism allows collecting answers from several peers, without requiring them to adhere to a unified ontology. Indeed, the presence of a mapping does not entail in any way that p_1 or p_2 use the same ontology.

The usage of the services for P2P indexing, XQuery analysis, P2P optimization and semantic reformulation will become clear in Section 3.4, when we discuss P2P querying.

3.4 Application services

Beyond the core services of the platform, the partners have developed a rich set of application services. WebContent applications are typically created by combining and configuring a subset of these services. In this section, we describe services involved in the construction of the Airbus application that we previously introduced. They will illustrate services the WebContent platform already supports and will in the future.

Acquisition services. The first step in a Web-based watch application is to gather documents available from public sources using a crawling service. Exalead provides a Web search engine and entreprise-level deployments [16]. The crawler is encapsulated in a WebContent service. It can be configured to crawl from a given set of URLs and up to a certain depth for a given list of file types. It also publishes a normalization service API that extracts the text stream from HTML and PDF documents. This stream is used for indexing by the search engine and as such produces a single `Text` media unit. The original document can also be retrieved for use by more sophisticated services, e.g., a service extracting news reports from pages including "noise" such as advertisements.

After the crawling phase, the documents are sent to a language recognition service that is used to keep only documents in some target languages. We are currently developing more sophisticated services to classify documents and filter them based on thematic criteria and specific profiles.

Content enrichment services. In the Airbus application (as in many other applications), enrichment consists primarily in searching documents for references to concepts and relations from a domain ontology. The corresponding text segments are then annotated with RDF statements referring to these instances. Examples of concepts include *plane* and *company*, events such as *launch* and *retirement of products*, and *incident*. Examples of relations include *manufacture* between a *plane* and the*company* that manufactured it.

We use the semantic annotation service developed by CEA LIST based on the LIMA linguistic analyzer [9] that handles all steps from tokenization to syntactic analysis and named entity extraction. Note that all these steps could be performed independently by different, more specialized WebContent services. The analyzer currently supports English, French, German and Spanish requested by the Airbus application (as well as Chinese and Arabic that are not). The analyzer extracts entities using manually-written or learned regular expressions. Other rules extract relations between entities. After the linguistic analysis step, the semantic annotator uses correspondence rules between entities and concepts of the domain ontology. In addition, relations between entities can mark object or datatype properties. Matched concepts, relations and text labels are used to try to find corresponding instances of concepts and relations in the ontology stored in a Protégé knowledge base [27]. If no such instances are found, they are created. Afterwards, annotations referencing those instances are created and associated with the corresponding text segments. Two other services based on similar technologies are available, developed by EADS and Thalès respectively. In our application, the three services are called sequentially and the results of the three analyses accumulated. A voting approach could be used to select the most relevant annotations.

Besides plain text paragraphs, HTML documents often include tables containing information about products (components, delivery date, ...), e.g., *ten A340 aircrafts have been delivered in 2006*. Those tables typically provide important and valuable information for the Airbus application. A semantic annotation web service [21] is used to annotate tables using a domain ontology. The latter defines the numeric and symbolic types of the domain and the semantic relations of interest, which link those types and which must be extracted from the tables. For instance, the relation *manufacturer* associates a *company* with an *aircraft*. *Company* and *aircraft* are two symbolic types whose domain of values (list of companies and list of aircrafts) are also stored in the ontology. RDF annotations corresponding to extracted instances of relation are associated with lines of the table in the WebContent document. This association

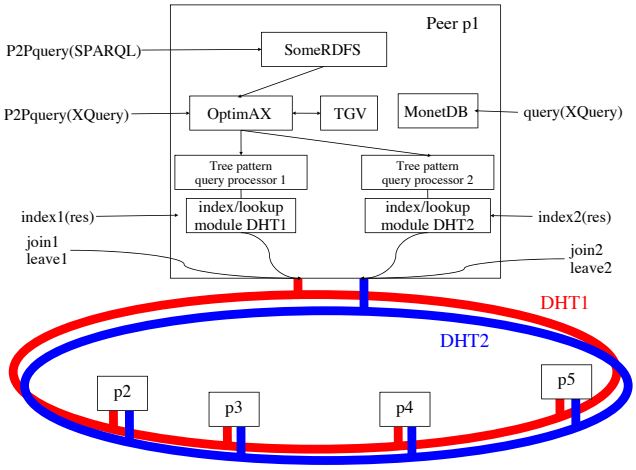

Figure 4: Outline of P2P services.

is weighted according to a pertinence degree. Consequently, it is possible to define SPARQL queries using the domain ontology to scan the annotated documents using the semantic search described in the next paragraph.

Exploitation services. Resources stored in a WebContent warehouse may be exploited in several ways. For our sample Airbus scenario, we have identified two main exploitation services: *querying* the warehouse based on Full Text, XML or semantic criteria; and *visualizing* a given set of resources, possibly obtained as a result of a query.

The full text indexing and search service is implemented on top of a cross-lingual search engine [10] that indexes all nouns, verbs, adjectives and complex terms of the `Text` media units and associates them to the unit's URI. Full text queries in one of the supported languages are processed to find simple and complex terms in all languages. Media unit URIs containing these terms are retrieved and sent back as query results. These URIs can then be used by other services to retrieve the text or other data from the XML repository, and prepare them for display to the user.

In this particular application, we use the MonetDB [39] XML query service with, MonetDB deployed on a single server. An additional flexible query Web service [12] has been developed to query WebContent XML documents that have been annotated according to a domain ontology. This service accepts as input a query built from a relation of the domain ontology and allows users to express preferences on their selection criteria [19]. This query is automatically translated into SPARQL before scanning the database. Related to the semantic annotation service for tables described earlier, this query service returns a list of XML resources matching the query, ordered according to the pertinence degrees associated with the corresponding RDF annotations.

In contrast, in the P2P architecture, a separate *P2P XML query service* is able to process queries over *all* the WebContent resources available in the P2P network. This service leverages: the DHT indexing services, the XQuery algebraic services, and the OptimAX distributed optimization services described in Section 3.3; and, the (single-site) XML query service previously outlined in this section. The roles of core and application services in the P2P architecture are outlined in Figure 4. The P2P XML query service is first handled by OptimAX which, with the help of the TGV XQuery analyzer, decomposes the query into tree pattern queries that are handled based on the DHT indexing service. For illustration, Figure 4 shows two DHTs, and accordingly, the detailed structure of peer p_1 features a tree pattern query processor for each of the DHTs. The parts of the original XQuery that could not be pushed to the tree pattern query processors are applied as post-processing by the local XQuery processor before returning the results to the user.

At a more complex level, a *P2P semantic query service* accepts as input queries expressed in a subset of the SPARQL semantic query language. Such a query received, e.g., at peer p_1 in Figure 4 may be reformulated via mappings, as explained in Section 3.3, leading to a union of SPARQL queries. These are then translated to XQuery and processed just like an incoming P2P XML query.

Front-end User Interface Components. The user interface of WebContent applications is typically built as a Web application composed of portlet components [24] assembled in the framework of Web portals such as Liferay[2] or eXo[3]. The WebContent platform provides a set of generic, reusable portlets for visualizing data such as annotations contained in WebContent documents. Inter-portlet communication is achieved through the Portlet Event interface [24]. Figure 5 shows three such portlets used to build part of the interface of the EADS strategic watch application for Airbus. The portlet in the upper part of the interface is based on an extended version of the Simile Timeline widget[4]. It displays all *events*, identified as such based on the domain ontology and obtained through the content enrichment services described earlier. These events are represented as a set of RDF annotations fed to the portlet, which transforms them into JSON (JavaScript Object Notation) data that can then be displayed by the timeline widget. This extended version features additional filtering and highlighting capabilities to help the user identify events of a particular type in the timeline, as well as overlay capabilities to display links between related events on demand (annotations also obtained from the content enrichment services).

The second portlet, in the lower-left part of Figure 5, is used here to generate charts that give an overview of the same events by type in different time intervals (years, months). The widget currently supports two representations: bar charts and line charts, and features an interface to specify which event types to visualize. While these two portlets rely solely on HTML, CSS and JavaScript to render the visualizations, other portlets can offer more advanced interactive visualizations based on rich graphics APIs such as Java2D. For instance, the third portlet in the lower right corner of Figure 5 is a generic visualization tool for RDF graphs represented as node-link diagrams. The portlet takes as input a set of annotations forming a graph, generates on-the-fly a GraphML representation of that graph, which is then sent to the Java applet served by the portlet. This Java applet computes a layout for the received GraphML data that is displayed to the user in a zoomable user interface that allows for smooth and efficient navigation (zooming and panning) in large networks, as well as for the dynamic reconfiguration of these networks (filtering by type, incremental layout modifications). In the context of the Airbus application, this portlet/applet pair is used to visualize the network formed by companies involved in the manufacturing of aeronautic parts and products, such as engines and planes. The network thus contains various types of nodes (*aircraft*, *engine*, *manufacturer*, etc.) and arcs (*is made of*, *manufactures*, etc.). When generating the GraphML representation of the graph, the portlet can optionally apply a Graph Style Sheet (GSS [28]) to the RDF graph if one is available for the underlying ontology(-ies). GSS is strongly inspired by CSS and makes it possible to declar-

[2] `http://www.liferay.com`
[3] `http://www.exoplatform.com`
[4] `http://simile.mit.edu/timeline`

Figure 5: Sample visualization portlets used in the Airbus application: event timeline, event statistics by year, interactive semantic network involving companies, products and components.

atively specify visual styling rules for the nodes and arcs of the graph based on their type. In Figure 5, the rendering of the network is driven by a GSS style sheet that assigns domain-specific icons to the various elements in the graph based on their class in the domain ontology (*aircraft*, *manufacturer*, etc.) and renders arcs with different styles depending on the type of relation in the ontology. This styling process improves the legibility of the graph compared to a generic, abstract representation in which all nodes and arcs would look the same.

As each WebContent application relies on domain-specific ontologies for the annotation of documents, generic portlet components provided with the platform must be easily customizable, as they have to adapt to the ontologies used in the considered domain(s). This is typically achieved by the WebContent application developer in portlet configuration files through the declarative specification of presentation and styling rules expressed with languages such as the above-mentioned GSS style sheet language and the Fresnel RDF presentation vocabulary [29].

4. RELATED WORK

Ideas and technologies both used and developed in the WebContent project are drawn from many domains of computer science. We focus here on work in the domains of software architectures, XML warehousing and information extraction.

The architectural approach that we chose in WebContent is resolutely *service oriented*. Today, a big majority of WebContent services are based on the Web Service technology as it was found to be one of the most optimal to implement. Other technologies have been considered, including CORBA [38] and UIMA (Unstructured Information Management Architecture [17]). The latter is close to WebContent in terms of goals and extent. It is based on a similar approach, with annotation tools working on data transmitted from one tool to another. The main difference with WebContent is related to its history: UIMA was first developed by IBM for its own needs and then released as free sofware. As such, it defines its own language and relies on non-standard technologies. On the contrary, WebContent started as a collaborative project between numerous academic and industrial partners. It was thus designed based on standards and open specifications from the beginning. UIMA has been adopted by many actors, and we plan to develop bridge components that will enable WebContent applications to use UIMA components and conversely enable UIMA applications to call WebContent services. As an open platform, we expect WebContent to be able to host any useful components.

The management of content leads to a number of systems sometimes called (Web) content management systems or enterprise content management systems [22]. XML is often used in such systems as a standard format to facilitate interoperabilty between applications. Borrowing the idea of warehouses from OLAP (OnLine Analytical Processing), a content warehouse aggregates and enriches information from many sources [2]. A similar idea is followed in dataspace platforms [20]. The idea to manage a content warehouse, i.e., a logically centralized repository, in a distributed manner has recently been promoted in [4, 6].

Service Oriented Architecture (SOA) and Semantic Web technologies are gaining momentum, and many initiatives on these topics have appeared recently. The TAO project [43] aims at defining an approach to migrate legacy applications to semantics-based SOA. The NeOn project aims at promoting the use of ontologies for large-scale semantic applications in distributed organisations [40]. The objectives are close to the ones of the WebContent P2P repository. SMILA [41] is a framework for building search solutions to access unstructured information. It was advertised as the base technology for the German government project called THESEUS. As WebContent, it is a service oriented platform that deals with integration issues. It is however not yet available.

Regarding information extraction, Gate [13] is one of the most successful open source text extraction frameworks. It however covers only part of the needs covered by WebContent, which also handles steps such as Web crawling and source documents conversion. Some Gate tools have actually been made compatible with WebContent and incorporated as services in the platform.

KIM [30] is designed for tasks similar to those of WebContent but is a more centralized, monolithic solution. It is based on well known components such as the above-mentioned Gate, Sesame[5] and Lucene[6]. But it is not possible to easily replace one component by another one that would offer the same type of functionalities. KIM does not explicitly address the problems of acquiring and transforming content, or the problem of scalability when indexing millions of (possibly multimedia) documents.

5. CONCLUSION

The WebContent platform provides means for opening existing tools to the Web Service world and to develop applications based on the warehousing of Web resources. The project is both a platform for building real-world, industrial applications, and a testbed for research projects involving peer-to-peer technologies, ontologies or machine learning techniques to name a few (the results of these projects are being published separately).

The Airbus application described in this paper is the first complete application (from Web crawling to exploitation by the end-user through the portal's front-end) implemented with the platform, based on service orchestration through an enterprise service bus (ESB). Besides this strategic watch application, other applications are currently being developed by the WebContent consortium, including the following two in the domains of food risk analysis and seismic event monitoring.

The purpose of the first application is to allow watching open sources on the Web concerning microbiological and chemical contamination of foods. It is designed for a network of food industry companies [34] and food agencies (AFSSA and EFSA) dealing with risk assessment. The application is also used to semi-automatically transform information found on the Web to feed statistical models in order to evaluate the exposure of a given population to a given chemical contaminant or to evaluate the behavior (growth, death or survival) of a microbial contaminant in a given food product.

The purpose of the second application is to watch news feeds like Google or Yahoo! News for notable seismic events and report them to the authorities with all known details. The application watches the feeds, selects documents about seismic events, and fills a dashboard with lines corresponding to the events described in the news reports, identifying important characteristics, such as the location and time of the event, its magnitude and the damages caused. Some of the challenges are (i) to differentiate references to old events inside a news report describing a recent event, and (ii) to group news reports about the same event even if the details change over time due to estimations being refined.

A development kit for the platform is available under a BSD license. It features the APIs consisting of the WSDL [11] service and XSD exchange model definitions; tutorials to implement services and portlets, invoke them from the ESB, and orchestrate them with BPEL programs. A C++ framework is also available, allowing the easy encapsulation of any C++ application in a SOAP Web service and more specifically a WebContent-compatible service. This C++ framework is available under the LGPL license. All software and documentation is made available on the WebContent Web site [44].

While the platform is open and freely available, different service providers can have different release policies, ranging from free services released as open source software to on-line restricted-access services with different licenses, e.g., for research or commercial use. The platform is open to new participants willing to build applications, provide WebContent compatible services or test their research prototypes in large-scale applications.

[5] http://openrdf.org

[6] http://lucene.apache.org

6. ACKNOWLEDGMENTS

This research was supported by the French National Research Agency (ANR) through the RNTL program, and the System@tic Paris-Région cluster.

7. REFERENCES

[1] S. Abiteboul, T. Allard, P. Chatalic, G. Gardarin, A. Ghitescu, F. Goasdoue, I. Manolescu, B. Nguyen, M. Ouazara, A. Somani, N. Travers, G. Vasile, and S. Zoupanos. WebContent: Efficient P2P warehousing of Web data (demo). In *VLDB*, 2008.

[2] S. Abiteboul, S. Cluet, G. Ferran, and M.-C. Rousset. The xyleme project. *Computer Networks*, 39(3):225–238, 2002.

[3] S. Abiteboul, I. Manolescu, N. Polyzotis, N. Preda, and C. Sun. XML processing in DHT networks. In *ICDE*, 2008.

[4] S. Abiteboul, I. Manolescu, and N. Preda. Constructing and querying peer-to-peer warehouses of xml resources. In *International Conference on Data Engineering*, 2005.

[5] S. Abiteboul, I. Manolescu, and S. Zoupanos. OptimAX: Optimizing distributed AXML applications. In *ICWE*, 2008.

[6] S. Abiteboul and N. Polyzotis. The data ring: Community content sharing. In *Conference on Innovative Data Systems Research*, 2007.

[7] B. Amann, C. Constantin, S. Jeanne, and L. Touraille. Recommandation et calibrage de processus WebContent avec πTunes (demo). In *Bases de Données Avancées (BDA)*, Guilherand-Granges, France, Oct. 2008.

[8] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[9] R. Besançon and G. Chalendar (de). L'analyseur syntaxique de lima dans la campagne d'évaluation easy. In *Actes de la 12e conférence annuelle sur le Traitement Automatique des Langues Naturelles, TALN 2005*, june 2005.

[10] R. Besançon, G. Chalendar (de), O. Ferret, C. Fluhr, O. Mesnard, and H. Naets. Concept-Based Searching and Merging for Multilingual Information Retrieval: First Experiments at CLEF 2003. In *Lecture Notes in Computer Science (LNCS 3237), Comparative Evaluation of Multilingual Information Access Systems*, pages 174–184. Springer, November 2004.

[11] D. Booth and C. K. Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, June 2007. http://www.w3.org/TR/wsdl20-primer.

[12] P. Buche, J. Dibie-Barthélemy, and G. Hignette. Flexible querying of fuzzy rdf annotations using fuzzy conceptual graphs. In *ICCS*, volume 5113 of *Lecture Notes in Computer Science*, pages 133–146, 2008.

[13] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[14] F. Dragan, G. Gardarin, and L. Yeh. Pathfinder: Indexing and querying XML data in a P2P system. In *WTAS*, 2006.

[15] EBM Websourcing. The petals esb website, October 2008. http://petals.objectweb.org/.

[16] The exalead crawler and search engine, 2008. http://www.exalead.com/.

[17] D. Ferrucci and A. Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, 2004.

[18] M. G. M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, and A. K. Y. Lafon. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), April 2007. http://www.w3.org/TR/soap12-part1/.

[19] O. Haemmerlé, P. Buche, and R. Thomopoulos. The miel system: Uniform interrogation of structured and weakly-structured imprecise data. *J. Intell. Inf. Syst.*, 29(3):279–304, 2007.

[20] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–9. ACM, 2006.

[21] G. Hignette, P. Buche, J. Dibie-Barthélemy, and O. Haemmerlé. An ontology-driven annotation of data tables. In *Web Information Systems Engineering - WISE 2007 Workshops*, volume 4832 of *Lecture Notes in Computer Science*, pages 29–40, 2007.

[22] Interleaf. Achieving competitive advantage with enterprise content management and truexml, 1999. http://www.interleaf.com.

[23] Java Community Process, JSR 208: Java Business Integration (JBI), August 2005. http://jcp.org/en/jsr/detail?id=208.

[24] Java Community Process, JSR 286: Portlet Specification Version 2.0, June 2008. http://www.jcp.org/en/jsr/detail?id=286.

[25] F. Manola and E. Miller. RDF primer, W3C recommendation, February 2004. http://www.w3.org/TR/rdf-primer/.

[26] D. L. McGuinness and F. van Harmelen. OWL web ontology language overview, W3C recommendation, February 2004. http://www.w3.org/TR/owl-features/.

[27] N. F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 2(16):60–71, 2001.

[28] E. Pietriga. Semantic web data visualization with graph style sheets. In *SoftVis '06: Proceedings of the 2006 ACM symposium on Software visualization*, pages 177–178. ACM Press, 2006.

[29] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel - A Browser-Independent Presentation Vocabulary for RDF. In *Lecture Notes in Computer Science (LNCS 4273), Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, pages 158–171. Springer, November 2006.

[30] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov. Kim - a semantic platform for information extraction and retrieval. *Journal of Natural Language Engineering*, 10(3-4):375–392, 2004.

[31] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF, April 2005. http://www.w3.org/TR/rdf-sparql-query/.

[32] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Int.'l Middleware Conf.*, 2001.

[33] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[34] Sym'Previus Platform, 2008. http://www.symprevius.org.

[35] N. Travers, T. Dang-Ngoc, and T. Liu. TGV: A tree graph view for modeling untyped XQuery. In *DASFAA*, pages 1001–1006, 2007.

[36] ActiveXML home page. http://www.activexml.net.

[37] Business Process Execution Language for Web Services (v1.1), 2003. http://www.bpelsource.com/bpel_info/spec.html.

[38] CORBA - Common Object Request Broker Architecture, 2008. http://www.omg.org/gettingstarted/corbafaq.htm.

[39] MonetDB database system with XQuery front-end, 2007. http://monetdb.cwi.nl/XQuery.

[40] NeOn - Networked Ontologies, 2008. http://www.neon-project.org/web-content/.

[41] SMILA - SeMantic Information Logistics Architecture, 2008. http://www.eclipse.org/smila/.

[42] Microsoft sql server 2008, 2008. http://www.microsoft.com/france/sql/sql2008/default.mspx.

[43] TAO - Transitioning Applications to Ontologies, 2008. http://www.tao-project.eu/.

[44] The webcontent platform web site, 2008. http://www.webcontent-project.org/.

[45] WebLab: An open platform for processing multimedia documents, 2008. http://weblab-project.org/.

[46] Web Services Choreography Description Language (v1.0), 2005. http://www.w3.org/TR/ws-cdl-10/.