ICHIM
PARIS 21-23 SEPT. 05
www.ichim.org

Digital Culture & Heritage
Patrimoine & Culture Numérique

Bibliothèque nationale de France, PARIS
Sept. 21st - 23rd, 2005
21 - 23 septembre 2005

FEDERATED RDF REPOSITORIES FOR INTEGRATED HYBRID MUSEUMS

Javier Jaén, Artur Boronat, José H. Canós

Information Systems and Computation Department
Polytechnic University of Valencia

Published with the sponsorship of the French Ministry of Culture and Communication

Actes publiés avec le soutien de la Mission de la Recherche et de la Technologie du Ministère de la Culture et de la Communication, France

Interprétation simultanée du colloque et traduction des actes réalisées avec le soutien de l'Agence Intergouvernementale de la Francophonie

Abstract (EN)

The Resource Description Framework (RDF) is a standard for the semantic web that allows describing web resources by means of metadata, providing “interoperability between applications that exchange machine-understandable information on the web”. Using RDF to describe resources in a distributed environment like the web or a grid needs a framework to enable distributed RDF specifications. On the one hand such specifications require mechanisms that allow the access to interrelated semantic information about resources no matter where this information is stored. In this way, this framework would avoid semantic islands creation by providing navigational support to reach in a transparent way remote metadata resources specified in a RDF model. On the other hand, it is likely that distributed RDF specifications are owned by different organizations or stakeholders which may store their RDF models in heterogeneous repositories. Thus, a framework providing the communicating layer that allows interoperability between several kinds of RDF repositories by means of a RDF models federation is also a requirement.

In this paper, we present a federation architecture for distributed RDF models supporting the two requirements listed above: transparent navigation irrespective of the locations where resources are stored and support for heterogeneous repositories. Firstly, we discuss a particular RDF repository that has been used to prove the feasibility of our approach and the correctness of our federation architecture design. Secondly, we show the advantages of using the .NET technology to implement a framework requiring integration and interoperability between heterogeneous components. Next, we discuss the communicating technology that has been used in order to guarantee remote access between distributed repositories and finally, we present the RDF models federation architecture, including the management of federation members and the distribution mechanism to establish links between remote RDF models that can be navigated transparently. The proposed models and implementation is the core persistence layer for a Hybrid Museum project known as MoMo.

Keywords: Semantic Web, Semantic Federation, Interoperability, Web Services, .NET

I. Introduction

In the last decade we have experienced several revolutions in the way we perceive and use computing services. The first revolution relates to a new global framework called “the Web” which evolved from a mere hypertext system for the management of general information about accelerators and experiments at CERN (Berners-Lee, 1990) to become a virtual place where advertising, selling, trading, socializing and many other activities take place. The second revolution is taking place by migrating computing and data sources from traditional computers to all sorts of embedded devices like personal device assistants, clothing, smart phones, cars, etc. resulting in massively interacting systems in which computing is distributed in nature. The third revolution known as the Semantic Web (http://www.w3.org/2001/sw/Activity) is transforming the Web and all sources of information into entities that are machine-understandable so that most traditional tasks can be automated and take place without human intervention.

However, as a result of this coming revolutions, new challenges associated to the integration of services that are massively present on distributed devices and take advantage of massively distributed semantic information have to be faced. Among these challenges we have to consider, firstly, how to avoid semantic islands of unrelated information so that infrastructures where information is correlated arise and may support the development of complex tasks. Secondly, avoid the centralization in the production of semantic information because this would prevent the semantic Web to be as successful as the WWW allowing every potential user to produce and publish a semantic piece of information. Thirdly, making a reality the interoperation of heterogeneous semantic repositories which are managed independently and owned by different stakeholders.

In this paper we introduce, formalize, and implement the notion of Semantic Resource Description Framework (RDF) Federation as an infrastructure that enables the correlation (linkage) and navigation of distributed semantic RDF information that is privately owned and stored on heterogeneous repositories. Semantic federations consist of individual heterogeneous semantic repositories that may be managed and queried independently but at the same time may be interconnected and navigated as if they were a single non distributed semantic space. Our solution makes use of standard technologies like XML, RDF and Web Services guaranteeing

extensibility and interoperation with other existing frameworks. Besides, we discuss an implementation based on .NET Web Services and present some interesting experimental results. The main contributions of our approach are, first, the definition of a mechanism for interconnecting distributed semantic resources that supports replication of information, second, the definition of a Web Services infrastructure for the navigation of local and distributed RDF repositories and, finally, the definition of a federation model and its associated Web Services infrastructure for the integration of semantic information that is distributed in nature. Additionally, another important contribution of this paper is the discussion about the suitability and efficiency of the .NET framework for the integration of heterogeneous systems.

The remaining sections of the paper are organized as follows: Section 2 discusses the existing related work; Section 3 presents the key concepts related to Semantic Repositories and Semantic Federations. Section 4 describes our distributed federation model, its Web Services based architecture and discusses the selected design and implementation choices; finally, Section 6 summarizes our approach, presents the obtained conclusions and establishes our future work in this area.

II. Related work

Wide acceptance and use of the Web as a platform for communicating knowledge and the consolidation of RDF as a common base for describing and sharing metadata increase the need for RDF-enabled repositories that support storage and querying in an efficient way. In (Gómez, 2002) a comparison between several tools that provide this kind of functionality is presented. It shows the tendency to provide storage and querying support for the RDF/S description base. Most of the studied tools support query languages based on a triple-based model, i.e., they use collections of statements of the form *(subject, predicate, object)* to encode RDF models. In addition to persistent storage, most of these tools support in-memory storage, in an attempt to minimize the response time. However, they are implemented following different design choices and even using different technologies, making interoperability unfeasible and hence counteracting the efforts for sharing knowledge through the semantic Web.

Heterogeneity is not a new problem, though. Researchers in the Information Systems field have worked during more than 20 years seeking acceptable and efficient solutions to the problem of the integration of heterogeneous data sources. Specifically, the notion of federation appears for the first time in the database field. A federated database system is defined in (Sheth, 1990) as "a collection of cooperating but autonomous component database systems". The components can differ in aspects like data models, query languages, etc. schema integration, negotiation, query (command) decomposition and optimization and global transaction management are part of the key issues to be solved but usually the relational or object oriented models are the ones taken for integration purposes.

More recently, Arms describes a federated digital library as "a group of organizations working together formally or informally, who agree to support some set of common services and standards, thus providing interoperability among their members" (Arms, 2000). Again, heterogeneity is hidden to provide information in an integrated way. A very interesting discussion about how to solve the interoperability problem in digital libraries using solutions ranging from standards to mediators can be found in (Paepcke, 1998). Finally, the Open Archives Initiative enforces interoperability between digital repositories by means of a metadata harvesting protocol (http://www.openarchives.org/). From a broader perspective, the use of ontologies and ontology integration as the way to interoperability has gained a lot of credit in the last years. In this trend, the problem is faced from a high abstraction level, where vocabularies for semantic information and their integration are considered. Our work, however, is focused on the lower layers of the integration problem where the repositories in which semantic information is stored are themselves heterogeneous. Closer to our approach, the Edutella project (Nejdl, 2002) provides a RDF based metadata infrastructure for P2P application, building on the JXTA framework. JXTA (http://www.jxta.org/) is a set of XML based protocols that provide typical P2P functionality, i.e. remote service access, peer discovery, peer groups, peer pipes, and peer monitors. The Edutella network connects highly heterogeneous peers making the distributed nature of the individual RDF peers completely transparent in the same way that our federation model does with its members. Our goal, however, is not to provide support for P2P applications, but allow interoperability between heterogeneous and distributed RDF repositories which are less dynamic in nature but support the definition of RDF properties across boundaries in a distributed way.

III. Semantic Federations

A Semantic Federation is a collection of heterogeneous distributed RDF repositories that can be accessed as if they were a unique local Semantic Repository and offers the following basic services

- Member management: registration and deletion of Semantic Repositories as members of the federation.
- Federation RDF collection management: mechanisms to carry out the federated model management services.
- Federated queries: global queries that are satisfied by contacting all member repositories in a federation.

To hide the underlying distributed repository architecture to a client, the federation provides the knowledge spread out among several repositories as a unique RDF collection known as the federated collection. This is achieved by means of a standard interface for Semantic Federations which abstracts the underlying repository infrastructure. Figure 1 shows the structured organization of a semantic federation modelled in UML notation.

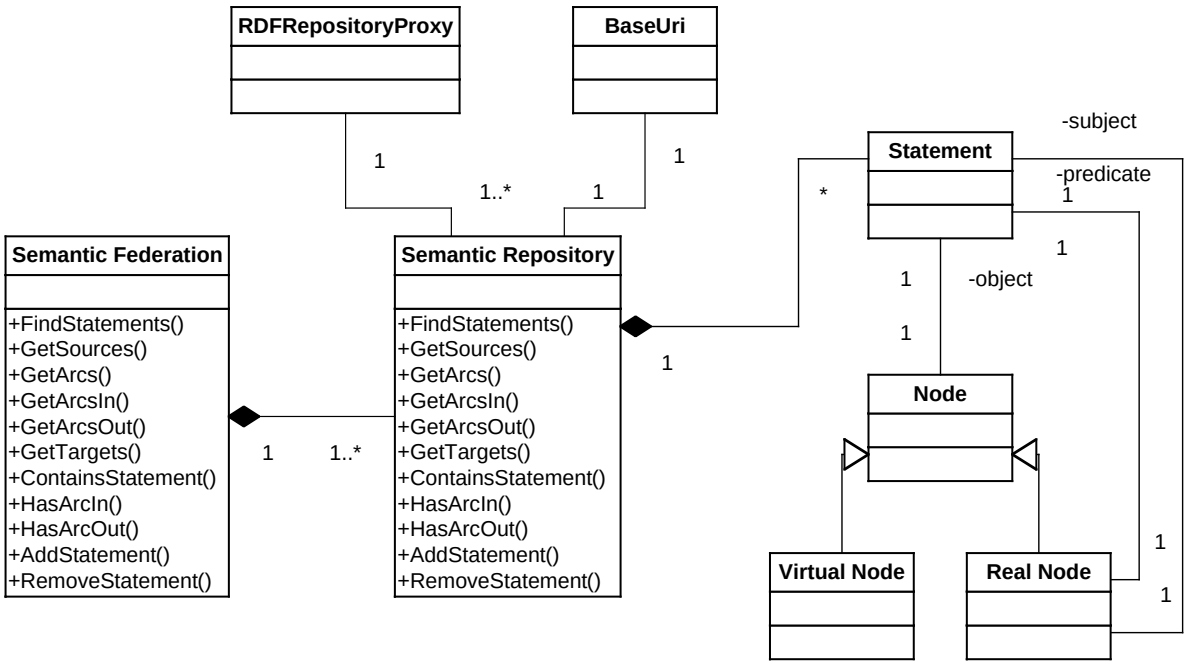

Fig. 1. Conceptual Model for RDF Semantic Federation

1. Member Management

Semantic Repositories can be added/removed to/from a Semantic Federation and stored in the same host where the federation remains or in a remote one. This structural organization can be conceptually modelled in UML notation as shown before. The classes involved in this model that allow the definition of a members infrastructure within a federation are:

- *RepositoryProxy* : a proxy that allows remote access to a Semantic Repository.
- *BaseUri* : holding the information that identifies a Semantic Repository in the federation, i.e., a url, the storage path and the storage name. Consequently, each repository must have its own BaseUri instance.
- *Semantic Repository:* wrapping a *RepositoryProxy* instance. It abstracts and simplifies the information that is needed to invoke in a correct way all the services of a *RepositoryProxy* instance. As a result, the services of the proxy are exposed in a more natural manner as if they were services of a local repository hiding the remote SOAP communication details. Furthermore, more than one *Semantic Repository* instance can share the same *RepositoryProxy* instance.

- *Semantic Federation:* consisting of several *Semantic Repository* instances representing repositories that may exist either in the same local host that holds the federation or in a remote host.

In Figure 2 we illustrate this with an example of federation consisting of four repositories, one of them is stored in the local host, two of them “r2” and “r3” are stored in the “issi.dsic.upv.es” host and the remaining “r4” is stored in the “momo.dsic.upv.es” host.

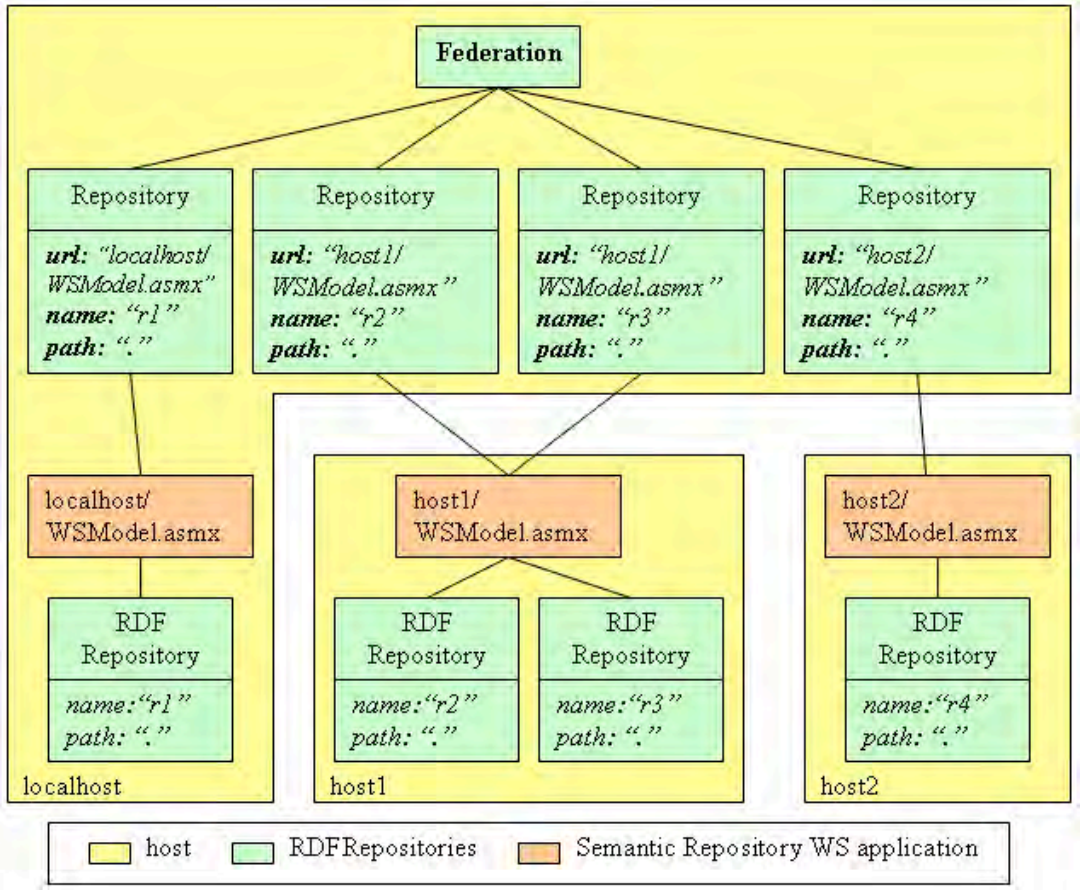

Fig. 2. Example of Distributed Federation.

Federations may add or remove semantic repositories by means of the *Semantic Repository* class. These repositories can be set in writing mode or read-only mode, so different federations may share a given member. Unfortunately, sharing members between different federations may cause collision problems. The required mechanisms for supporting transactional behaviour are out of the scope of this paper and will be an issue of future research.

2. Federation Collection Management

A Semantic Federation cannot be seen just as a collection of separate Semantic Repositories but rather as a complex Semantic Repository where RDF properties involving nodes that are stored in distributed semantic repositories (distributed statements) may be defined. This is done in our framework by distinguishing between virtual and real nodes (Figure 3). A real node is a resource node that represents a unique real web resource in the federated collection. A virtual node is a resource node that represents a real node in a remote semantic repository in order to provide navigation between distributed RDF collections. It acts as a proxy in the local host for the real node that is located in a remote host.

A virtual node is for internal use of the federation and an external client of the federation cannot directly access it. A real node may have many virtual nodes pointing to it, each one in a different repository. We distinguish a virtual node from a real one by means of two specific RDF properties: *remote* and *is_linked_by*. The subject of a statement having the *remote* predicate is a virtual resource and its object indicates the semantic repositories of the federation where the real node (or its replicas) are stored. This property only exists for a virtual node. The subject of a statement with the *is_linked_by* predicate is a real resource and its object indicates the semantic repository of the federation where a virtual node exists pointing to it. Both kinds of statements are needed in order to provide bidirectional navigation between repositories and to maintain consistency.

To illustrate our approach, let us assume that we use our federation framework to define a metadata infrastructure that provides support to different museums. The metadata infrastructure consists of several semantic repositories that contain information about art works and artists which are inter-linked by means of distributed statements. In this case, we could have a real resource node called *author_1* in a repository *spanish_authors* that is located at the *museoreinasofia.mcu.es* host and another resource named *artwork_20* in a repository *spanish_art_works* located at the *momo.dsic.upv.es* host, as it is shown in Figure 3. If we want to define an author property indicating that resource *author_1* is the author of the resource *artwork_20*, we have to define a virtual resource node called *author_1* in the *spanish_art_works* repository. Then, a *remote* property for the new virtual node is defined indicating the semantic

repository where the real node is stored. The object value of the *remote* statement is the identifier of the repository that contains the real node within the federation, the serialized fields values of the *BaseUri* instance relate to this repository. Additionally, an *is_linked_by* property is defined for the real node in the *spanish_authors* repository indicating that a virtual node stored in the *spanish_art_works* repository is pointing to this resource. Again, the object value of the *is_linked_by* statement is the identifier of the repository that contains the virtual node within the federation, and the serialized fields values of the *BaseUri* instance relate to this repository. At the end, an author property for the remote *artwork_20* resource was defined (see internal federation view in Figure 3) but the federation hides all the internal statements so that a client application may navigate such a property that relates two resources of the federation RDF collection without worrying about their physical location (external client view in Figure 3).

Once virtual and real nodes have been defined, we can take our distribution mechanism one step further by including distributed statements. A distributed statement in our context is a statement composed by nodes that belong to different semantic repositories. These repositories may exist either at the same local or at distributed hosts. The predicate node is always stored in the same repository where the subject node is because it does not represents any web resource. Working with distributed statements involves certain risks when managing the federated RDF collection. These risks are due to potential server failures that may make a Semantic Repository Web Service application unreachable.

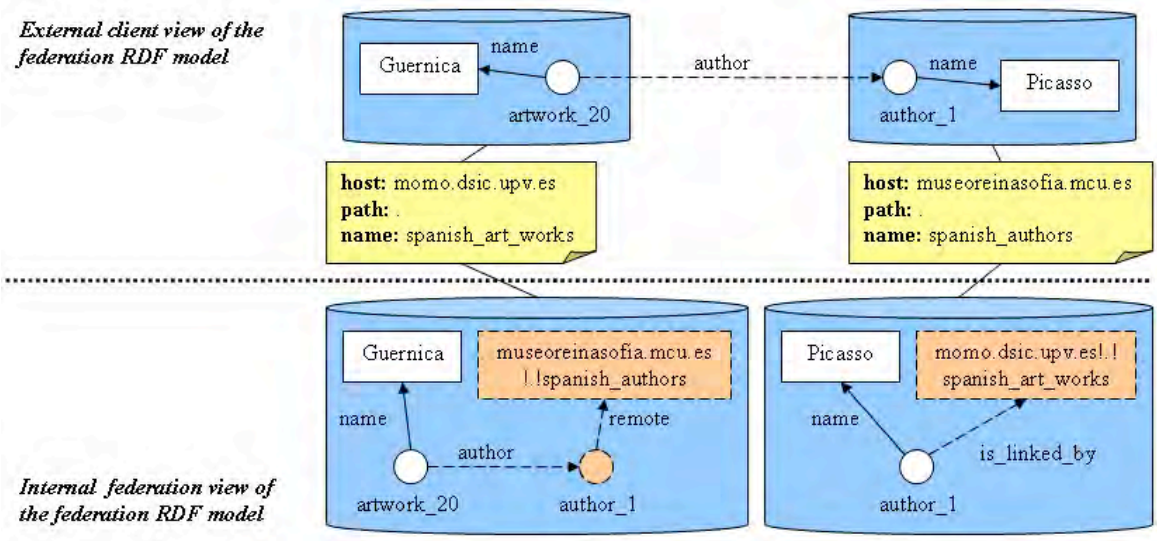

Fig. 3. External and Internal views of distributed RDF links.

Under these conditions, although a federated RDF collection is supposed to be completely queried as if it was a unique RDF collection, we might not be able to access the entire collection. To overcome this potential weakness of our framework we have implemented a Semantic Repository replication mechanism. This is done by making use of our previously defined *remote* statement so that whenever a virtual node is created it may contain references to several replicas of a real resource. Replicated real nodes have to be stored in different repositories in order to perform a correct replication. If a virtual node tries to access a real node located in a host that is down, it automatically searches a reachable replicated real node by inspecting its additional remote properties. This simple mechanism provides robustness against server failures.

The two main services for managing the federated RDF collection are the *AddStatement* and the *RemoveStatement* services of the *Semantic Federation* class. Given that these are interesting examples of distributed management services we present their singularities in the next subsections.

A) AddStatement Service

The *AddStatement* service belongs to the *Semantic Federation* class and allows the creation of new statements in a federated RDF collection. It requires three nodes as arguments, which are the subject, the predicate and the object respectively. Each one of them is identified by an instance of *BaseUri* class. From an internal point of view, the new statement might be distributed or not. The federation client can indicate each node location by means of the *BaseUri* instances that identify each of the nodes of the statement.

Let us consider that *IsReal(node)* and *IsVirtual(node, BaseUri)* are services of the *Semantic Repository* class. *IsReal(node)* determines if a node is real in a specified repository, and *IsVirtual(node, BaseUri)* determines if a node is virtual in a specified repository that points to another repository identified by the *BaseUri* instance. Making use of these services an algorithm for inserting statements in a *Semantic Federation* can be defined in a straightforward way.

Moreover, to define a replicated real node for a distributed statement just a virtual node pointing to both real nodes with the *remote* properties must be created. To achieve this, the same

distributed statement is inserted, but now changing the BaseUri instance of the object node to indicate that it is allocated in another repository.

B) RemoveStatement Service

The *RemoveStatement* service also belongs to the *Federation* class and allows deleting a statement from a federated RDF collection. It requires three nodes as arguments, which are the subject, the predicate and the object of the statement to be removed. From an internal point of view the statement may be again distributed or not.

The removal process must take into account that whenever the last statement associated to a given resource in the federated RDF model is removed the associated node in the graph must also be removed. Besides, if the statement being removed is a distributed one then all the statements that enable the distributed navigation must also be deleted to keep a consistent federation. Therefore, the *RemoveStatement* service has to consider the statements of the federated RDF model that have the subject node or the object node in common with the statement that is removed. If a distributed statement is removed then the *remote* statement of the source repository where the subject node is stored and the *is_linked_by* statement of the target repository where the object node is stored are also removed. Finally, the statement that has the same subject and predicate as the distributed statement within the source repository is removed provided there are no other replicated real nodes.

3. Federated Queries

The *Semantic Federation* class also offers services to navigate a federated collection as if it was a single centralized repository. While management services need to know the location of the nodes passed as arguments, navigation services do not need this information. Nevertheless, when the service returns a node or a statement, each node contains its physical location within the federation by means of a *BaseUri* instance. In this way, the client may decide to process this information or to discard it.

This service allows searching through nodes and arcs of the federated RDF collection, even in the case when the arcs represent the predicate of a remote statement. A query service runs the same

service for all the federation repository members and merges the returned values. The merging process filters out the internal statements used for managing distributed statements and the replicated nodes. In this way, all the returned statements contain real nodes.

IV. A Web Services Architecture for Semantic Federations

Once the models for Semantic Repositories and Federations have been presented we have to consider the most suitable design and implementation choices for the ideas presented above.
The key design goals are, firstly, supporting remote access to Semantic Repositories and, secondly, solving the heterogeneity problem that arises when different implementation technologies (Gómez, 2002) are used on different Semantic Repositories. Therefore, a communication mechanism that allows reaching RDF collections regardless of their implementation details and their physical location is needed.
The requirements stated above are not specific to Semantic Federations, but have been also considered mandatory in the context of the World Wide Web when it comes to incorporating services that can be published, found, and consumed by third parties in a standard way so that distributed computing is fully exploited on the Internet. In this field, a core number of technologies around the notion of Web Service (WS) have been proposed. These are based on open standards with multi-vendor and multi-language support and platform independence so that software elements that are written in different languages and on different platforms may interact with each other. An XML WS (Booth, 2003) can be described as a software service exposed on the Web through SOAP (Gudgin, 2003), described with a WSDL (Christensen, 2001) document and registered in UDDI (Bellwood, 2003). Not taking benefit of all the work to that has been done with WSs to support remote access and interoperability in our problem would be a great mistake. Besides, if we analyze the formal definitions given before it becomes evident that both Semantic Repositories and Federations expose a collection of services (expressed in the formal model as functions) which makes them, at least conceptually, very close to the notion of service as defined in the WS arena.
In the next sections, we describe a WS infrastructure that allows remote access to heterogeneous implementations of semantic repositories. Then, we explain how the specific and platform

dependent Semantic Repositories implementations may become transparent to the Semantic Federation and, finally, we present how a Semantic Federation provides seamless access to an RDF collection hiding the underlying architecture of distributed repositories.

1. A Web Services Based Semantic Repository

RDF collections are stored in Semantic Repository implementations that usually provide services for resource management, persistence and navigation. In our context, we have to select a suitable interface not only for navigating RDF models so that software agents may explore heterogeneous RDF repositories but also for managing RDF models so that any stakeholder is able to manipulate them without caring about implementation details. Taking the WS approach, a standard interface for performing these activities must be proposed. However, such a standard has not been yet defined as part of the ongoing W3C activities. Consequently, after studying several Semantic Repository implementations in (Gómez, 2002), we have chosen the model proposed in Redland (http://www.redland.opensource.ac.uk/) as a valid candidate standard interface to reach our goals.

2. The Redland Library

Redland is a flexible framework that provides a high-level interface allowing RDF models to be stored, queried and manipulated. It implements the RDF core layer and all the lower layers presented by Berners-Lee (Berners-Lee, 2000), including XML and URI support. Its interface allows managing RDF specifications without taking into account lower details such as the RDF-XML representation of an RDF specification. Its main features are: modular design which facilitates its maintainability and optimizes its behaviour, stable high-level interfaces that allow its integration with other applications, and multi-language support for portability purposes.

The core Redland classes that offer support for model management are:
World: handling all the startup and shutdown functions

Node: representing nodes as defined in the *RDF Model & Syntax specification* (Lassila, 1999). These include *resources* to represent web resources identified by URIs; *literals* that represent values; *blanks* as auxiliary nodes; and *li* nodes to represent collection properties.

Statement: representing graph arcs as defined in (Lassila, 1999). A Redland statement is composed by three nodes representing the subject, the predicate and the object of the property in a RDF specification or, using a different terminology, the source, the arc and the target if we consider the RDF graph.

Model: a set of statements usually held in one repository. It provides the main application programming interface for Redland.

Storage: It abstracts the physical storage of models. The storage can be in memory or persistent in a database. From the application's point of view, the interface of this class should never be visible.

The chosen interface provides several services that implement the matching and management functions of the federation model presented before.

2. The RedlandNET Library

The .NET Framework (http://www.microsoft.com/net/) is a new computing platform that simplifies application development in the highly distributed environment of the Internet. It has two main components: the common language runtime and the .NET Framework class library.

The Common Language Runtime (CLR) acts as a mediator to provide platform independence and, as a result, a program targeting the .NET platform may run everywhere. The .NET Framework class library is a collection of reusable types that are tightly integrated with the common language runtime. The class library is object-oriented, providing types from which new functionality may be derived. This not only makes the .NET Framework types easy to use, but also reduces the learning curve of its new features.

Accordingly, the .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a wide variety of programming languages. This cross-language interoperability is carried out by the existence of assemblies in the .NET Framework. Assemblies are the building blocks of .NET applications; they form the fundamental

unit of deployment, version control, reuse, activation scoping, and security permissions. They contain code that the common language runtime executes.

Essential parts of this framework, including CLR and .NET Framework base classes, have been standardized and collected under the name of Common Language Infrastructure (CLI) which has become an ECMA standard. Additionally, C# is also an ECMA standard object-oriented programming language that is part of the Visual Studio .NET programming environment. Furthermore, the Mono project (http://www.go-mono.com/) is bringing the .NET platform to Linux. Therefore, .NET is able to provide both cross-language and cross-operating system support.

Language interoperability can help maximize code reuse and, as a result, improve the efficiency of the development process. These interoperability features convinced us to compile the Redland library on the .NET platform. Our aim was to reach a .NET assembly collecting all the functionality of this library, resulting in managed code that targets the CLR. However, the Redland library targets native code and, therefore, we had to develop wrapper classes for each Redland class by means of managed extensions for the C++ programming language. We call the resulting library *RedlandNET*. RedlandNET classes benefit from the object-oriented features embedded in the .NET intermediate language, such as a better object-oriented encapsulation than the one simulated in the Redland library. The new obtained library is usable from all the languages that target the .NET platform in an easy way, such as C#, VisualBasic, etc., and even from those functional languages such as F#.

Another reason to bring the Redland library to the .NET platform is that .NET provides comprehensive support for major open-standard WS technology and Visual Studio .NET provides a development environment for writing and deploying such services in a relatively simple manner.

A) Remote Access to Semantic Repositories through XML Web Services

In this section, we explain how the services of a Semantic Repository may be reached from a remote host regardless of its implementation details by means of the interface discussed above. To do so, we have developed a WS application that may wrap any kind of RDF repository.

Given that all the repositories must agree on the same interface, the argument types for these services must be common for all of them. This collection of standard common types have been defined as SOAP serializable classes and are a fundamental building block when speaking about interoperability with heterogeneous Semantic Repository implementations.

In summary, the Semantic Repository WS application exposes methods to manage and query an RDF collection as WSs so that any client application is able to remotely access all the functionality of a specific Semantic Repository. Figure 4 illustrates this architecture where the Semantic Repository WS application constitutes the main interface and *SerializableStatement*, *SerializableNode* and *SerializableUri* are the common types that are used to exchange data between the WS and the client.

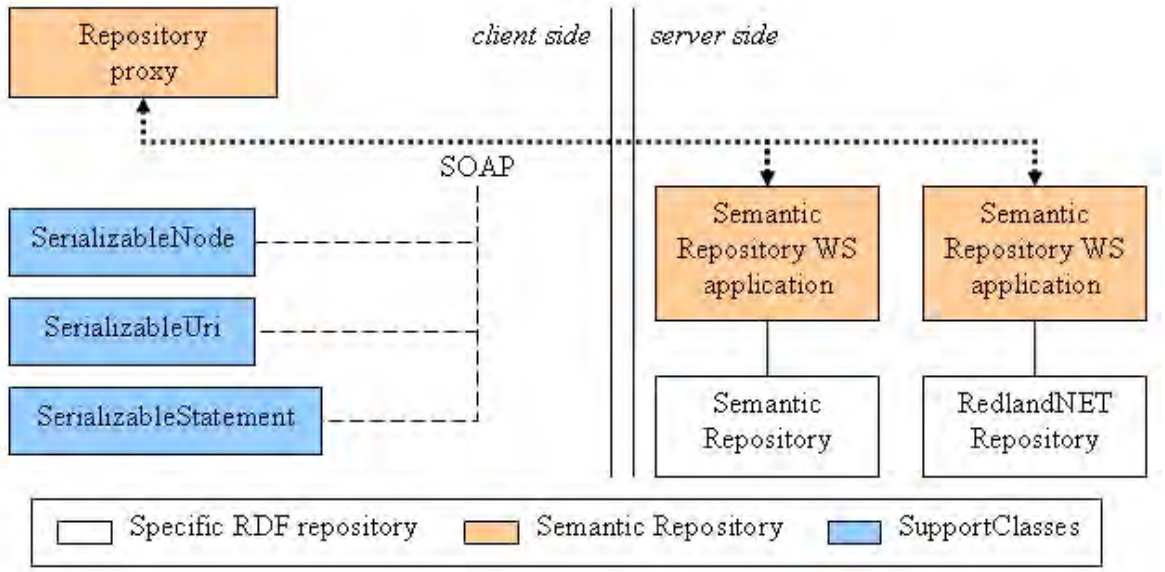

Fig. 4. Semantic Repository WS architecture.

By doing so, we are facilitating the adaptation of any existing specific Semantic Repository implementation to make it accessible as a WSs collection. Algorithmically, each one of these WSs performs the following tasks:

1. Open the storage that persistently holds the statements of the model,
2. Load the model from the persistent statements,
3. Deserialize the input arguments and create the suitable repository library classes instances associated to them.
4. Invoke the corresponding repository service,

5. Serialize the return value into serializable class instances and free the reserved resources for the return value,
6. Close the model,
7. Close the storage,
8. Return the serialized instance.

Additionally, in order to load an RDF collection, each WS needs a storage name, which is the name of the database where the model is going to be stored, and the storage path that can be either relative or absolute in the file system of the server. This information is also passed as arguments for each web service invocation.

All the necessary startup and initialization functions of a specific Semantic Repository implementation are done as soon as the WS application is started. If the WS is terminated all the resources are properly freed.

Finally, it is important to note that accessing the Semantic Repository WS requires a proxy that hides the SOAP serialization. This proxy is automatically generated by the .NET Visual Studio environment but we have slightly modified it in order to allow hot deployment of remote Semantic Repositories.

V. Conclusions and further work

Once the models for Semantic Repositories and Federations have been presented we have to consider the most suitable design and implementation choices for the ideas presented above.

The key design goals In this paper, we have introduced the notion of Semantic RDF Federation to represent distributed collections of RDF specifications. The semantic federation model is an extension of the classical RDF model based on distributed statements, that is, tuples (object, property, value) where object and value belong to different repositories. We have presented a formal model of Semantic Federations, and implemented a framework for federating RDF repositories based on that model.

Our approach uses the Redland library Model interface as the key interface for accessing any individual RDF repository and for accessing the federation itself. The framework benefits from

the cross-language and cross-platform interoperability features that the .NET platform provides, and the support that it offers to Web Services and embedded devices. The environment is being used to implement the metadata persistence layer within the MoMo project (Jaén, 2003).

Client applications using the federation WS application may be of different nature. We provide support for both desktop and embedded devices, such as Pocket PCs. However, in the latter case, the lightness needed for running applications with limited resources may become a problem. Particularly, the .NET Compact Framework does not support the standard SOAP serialization for all kind of objects. Therefore, a PocketPC client would not be able to deserialize the returned compound objects if the standard Federation WS application was used. To avoid this, we are developing our own serialization mechanism for pocket PCs using the .NET Framework System.Reflection namespace, which has the ability to dynamically create and invoke types. This namespace, also included into the .NET Compact Framework, makes the SOAP serialization for smart devices possible.

Another important issue for further work is the fact that deleting a statement from a RDF model can lead to semantic inconsistencies if dealing with the last link between two remote RDF resources. Consequently, we are developing a new version of the delete operation that takes care of the consistency problems in the same way relational databases behave. We are also studying the replication of RDF repositories so that updates in a repository will be automatically propagated to its replicas. We will also consider the performance of transactions over sets of statements, such as the statements that provide the distribution mechanism between remote repositories into the federation. Finally, we plan to study communication mechanisms between remote federations in order to provide full support for members sharing.

VI. Acknowledgments

This work has been partially funded by Microsoft Research under the “MSR Innovation Excellence Award for Embedded Systems”.

References

Berners-Lee, T. (1990) Information Management: A proposal. CERN http://www.w3.org/History/1989/proposal.html

Gómez-Pérez A. (2002) Ontoweb: Deliverable 1.3 A Survey on ontology tools http://ontoweb.aifb.uni-karlsruhe.de/Members/huro/MyPublications/OntoWeb%20Deliverable%201.3

Nejdl, W. et al. (2002) EDUTELLA: A P2P networking Infrastructure Based on RDF. *WWW2002*, May 7-11, 2002, Honolulu, Hawaii

Endig, M., Hding, M., Saake, G., Sattler, K. and Schallehn, E. (2000) Federation Services for Heterogeneous Digital Libraries Accessing Cooperative and Non-cooperative Sources. In Proceedings of Kyoto International Conference on Digital Libraries: Research and Practice. IEEE Computer Society Press

Sheth, A., Larson, L.. (1990) Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. In ACM Computing Surveys, Vol. 22, No. 3, 183-236.

Mena, E., Kashyap, V., Sheth, A. and Illarramendi, A.. (1996) OBSERVER: An approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. Proceedings of the 1 st IFCIS International Conference on Cooperative Information Systems (CoopIS '96), Brussels, Belgium.

Lassila, O., Swick, R. (1999). Resource Description Framework. (RDF) Model and Syntax Specification. W3C Recommendation 22 February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

Berners-Lee, T. (2000) Building the future. XML and the Web, XMLWorld 2000, Boston.

Arms, W. (2000) Digital Libraries. The MIT Press, Cambridge, Massachusetts.

Beckett, D. (2001) The Design and Implementation of the Redland RDF Application Framework. Tenth International World Wide Web Conference. Hong Kong. http://www10.org/cdrom/papers/490/

Jaén, J., Canos, J.H. (2003). A Grid Architecture for Building Hybrid Museums. Springer Verlag LNCS. Second International Conference on Human Society@Internet, Seoul, Korea

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard (2003). Web

Services Architecture. W3C Working Draft

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Frystyk Nielsen, H. (2003). SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001) Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

Bellwood, T., Clément, L., von Riegen, C. (2003). UDDI Version 3.0.1. UDDI Spec Technical Committee Specification. 14 October 2003. http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

Paepcke, A., Chen-Chuan, K., Chang, García-Molina, H. and Winograd, T. (1998) Interoperability for Digital Libraries Worldwide. Communications of the ACM, Vol. 41, No.4