Semantic web services in corporate memories

Moussa Lo[1,2], Fabien Gandon[1]
[1]INRIA, ACACIA Team, 2004 rt Lucioles, BP 93 06902 Sophia Antipolis, France
[2]LANI, Université Gaston Berger – BP 234 Saint-Louis, Senegal
lom@ugb.sn, Fabien.Gandon@sophia.inria.fr

Abstract

We present our experiment in integrating semantic web services in the existing corporate semantic web server architecture we use to implement corporate memories. We rely on a semantic web search engine, to provide a semantic registry and automatically discover and invoke corporate applications wrapped into semantically annotated web services. Using rules we also demonstrate how to compose the web services with queries on the knowledge stored in the corporate memory to automatically populate the service inputs.

1. Introduction

Until the end of the 90's, enterprise modeling has been mainly used as a tool for enterprise engineering. But the new trends and the shift in the market rules led enterprises to become aware of the value of their memory and of the fact that enterprise model has a role to play in knowledge management (KM) too. Just like data-integration problems can benefit from corporate-level models, technology and application integration problems can benefit from these same models. This was recognized by practitioners of Enterprise Application Integration but it requires a programming paradigm at a level of abstraction high enough to ease its implementation.

An organizational memory is an explicit, disembodied, persistent representation and indexing of knowledge and information or their sources in an organization, in order to facilitate its access, share and reuse by members of the organization, for their individual and collective tasks [6]. In the past, we experimented with agent-based architecture for distributed KM [6][7]. At that time semantic web hadn't meet the web services while all our protocols and messages were at the knowledge level. With the emergence of frameworks to semantically annotate web services a new paradigm can be used to integrate enterprise applications in a model-based memory. A corporate semantic web is an extension of the current use of web technologies in intranets, exploiting semantic web frameworks to semantically annotate resources available on the intranet and relying on these annotations to assist corporate actors in their daily tasks. In this article we describe our first experiment in integrating semantic web services in the existing semantic web server we use to build corporate semantic webs.

In section 2 we show how semantic web services allow us to integrate dynamic resources in corporate memories. First, we summarize our previous work on corporate semantic webs; then, we introduce our needs to take into account dynamic resources which led us to integrate web services into corporate semantic webs. Finally, we survey existing frameworks of semantic web services and give our position in the semantic web service stack.

The section 3 presents our current implementation embedded in the semantic web server architecture. We describe our architecture which relies on Corese, a semantic web search engine and provides automatic discovery and invocation of annotated web services. Using production rules we also demonstrate how to facilitate service composition. Finally section 4 tackles the original issue of composing corporate web services with knowledge from the corporate memory. We explore two paths: (a) using semantic types to attach queries to service inputs and (b) turning the search engine itself into a composable service of the memory.

2. Dynamic resources in corporate memory

2.1. Corporate semantic webs

Semantically annotated information worlds are, in the actual state of the art, an effective way to make information systems smarter i.e. a little semantics can go a long way. If a corporate memory becomes an annotated world, corporate applications can use the semantics of the annotations and through inferences

help the users in their interactions with the corporate memory.

The ACACIA research team focuses on knowledge management solutions based on semantic web technologies. We use RDF Model, RDF Schema and OWL (essentially OWL Lite) to describe ontologies and implement knowledge models [6]. Organizational entities and people are annotated in RDF and its XML syntax is used to store and exchange the annotations. Using W3C recommendations, information systems benefit from all the web-based technologies for networking, display and navigation, integration, interconnections, customizing, etc.

We can summarize our approach in three stages:

1. To apply scenario-driven knowledge engineering techniques in order to capture the needed conceptual vocabulary. We then specify the corporate memory concepts and their relationships in an ontology and formalize them in RDFS/OWL.
2. To use the conceptual vocabulary of the ontology and the scenario analysis to develop corporate and user models. These models are implemented in RDF and instantiate the RDFS/OWL schema.
3. To structure the corporate memory using RDF annotations on the documents: these annotations instantiate the RDFS/OWL schema and make reference to the corporate and user models.

2.1. Corporate semantic web services

More and more often, our research team must face scenarios requiring not only knowledge access but also computation, decision, routing, transformation, etc. Until now, the corporate semantic webs we designed focused on providing a unified and integrated access to a range of knowledge sources; but there is a growing demand to get the same facility to access corporate applications and services and to integrate both worlds. Users expect Information Technology (IT) managers to get very different computing systems (desktops, mobile phone, PDA, mainframes, etc.) to talk together and to get the variety of applications that run on them to talk together. Users don't only want to get access to the needed pieces of information, they want it in a format they are used to, with some certification of quality and of provenance, with appropriate tools to analyze it, modify it, etc.

Usage scenarios are evolving from the problem of providing a unified access to information to the problem of providing a unified access to information and applications. Corporate memories, as they are specified now, not only include information mediums but more generally:

- information *storage* services including: document sources (digital libraries, mailing-lists, forums, blogs, etc.) and dedicated systems (corporate or public databases, ERP, data warehouse, etc.);
- information *creation* services including: sensors (e.g. location tracking, presence & availability), computation and inference systems (e.g. data analysis tools);
- information *flows* management services including: secured transport channels, business rule engines and workflow systems, connectivity management, privacy enforcement and trust propagation;
- information *mediation* services including: matchmaking directories, translation and mapping services, contract and service quality enforcement;
- information *presentation* services including: multimedia transformation, contextual adaptation, dynamic customization and manipulation;

All these services may be internal or external to the company yet users want them to interoperate smoothly and, even better, to automatically integrate their workflows at the business layer.

Web services allow organizations to make public a programmatic access to one of their application without exposing the internal architecture of their IT systems. However, compared to agent-based platforms we used before [6][7], these technologies had the disadvantage to remain at the syntactic level while all the resources we manipulate are described in ontology-based models enabling us to leverage the semantics of descriptions in inferences. The emergence of semantic web services provided us with a new paradigm to identify and integrate corporate or public services at a semantic level like other corporate resources.

Thus the idea was for corporate web services to rely on a semantic web server like for other KM applications we designed: to provide a portal, to include annotations to describe services like we did with other resources before and to use our semantic web search engine to retrieve them like we used it to retrieve knowledge resources. To do so, we needed to rely on schemata to annotate these new resources.

2.3. Semantic web services frameworks

Semantic web services (SWS) frameworks allow service providers to enrich the service descriptions with formal annotations of their capabilities in order to be automatically discovered, executed and composed [11]. Many frameworks have been proposed [3] among which main ones are: OWL-S, WSMO, and WSDL-S.

OWL-S [10][12] is a set of OWL ontologies for describing web services. It has been developed to

provide the building blocks for encoding rich semantic service descriptions and consists of three main upper ontologies used to describe three facets of the services:

- The Profile facet is used for describing essentially the non-functional properties (service name, category, quality of service, etc.);
- The Process facet gives a detailed description of the operation, its inputs and outputs and can even detail its internal processes and, if it is the case, it identifies the other services it is composed of;
- The Grounding facet provides details on how to interoperate with a service via messages.

The service profile provides the information needed for an application to discover a service. The service model and service grounding provide the information needed for an application to make use of a service.

Semantic Web enabled Web Services (SWWS) [2][14] aims at providing a web service description framework, a web service discovery framework and a mediation platform for web services. One result is the Web Service Modeling Framework (WSMF) [5] that provides a conceptual model for developing and describing web services and their composition. It consists of four main elements: ontologies that provide the terminology used by other elements, goal repositories that define the problems that should be solved by web services, web services definitions that define various aspects of a web service, and mediators for interoperability problems [3]. The Web Service Modeling Ontology (WSMO) [15] is an ontology that addresses two aspects: capability which is a non functional description of a Web Service (preconditions, post-conditions, assumptions, effects), and service interfaces which specify the behavior of the service to achieve its functionality by providing information about the operational competence on the web service (how a client can communicate with the service, how the overall service functionality is achieved using other services…).

OWL-S and SWWS start at the knowledge level and are then grounded in WSDL. WSDL-S [1] starts from WSDL and augments its expressivity with semantics descriptions. Using extension slots of WSDL, it provides a mechanism to add annotations in a WSDL description to semantically describe the capabilities and requirements of Web services (inputs, outputs, preconditions, effects, operations). Again, these annotations are based on external ontologies. WSDL-S is the base of the SAWSDL recommendation currently under construction at W3C.

Because OWL-S is directly expressed in OWL and because our approach to corporate semantic webs relies on OWL too, we relied on OWL-S for our experiment. However, in our current scenarios, we use only the profile and the grounding of OWL-S plus the input and output description in the process description. This corresponds in WSMO to the services capabilities and input output description and also to the semantics added by WSDL-S / SAWSDL to annotate services.

3. Corporate semantic web services

3.1. Corese as a semantic UDDI-like registry

Since our approach to corporate semantic webs relies on OWL we relied on OWL-S for our annotations of the web services; OWL-S offers the framework the closest to semantic web frameworks and thus is directly compatible with Corese. WSML or WSDL-S would have required mappings (we are considering the use of GRDDL from W3C together with SAWSDL for our next experiment). We use the profile, the process part offering input and output descriptions and the grounding of OWL-S to annotate web services wrapping corporate applications.

In this first prototype, to wrap a corporate application into an annotated service, one must: (1) write/wrap and deploy the corresponding web services; (2) annotate the web services with OWL-S. All our services were based on JWSDP 1.3 and wrapped legacy services of our intranet.

The corporate semantic webs we experimented with are based on Corese: a semantic web search engine enabling us to query the semantic web statements. As summarized in Figure 1, it relies on a mapping between RDF/S-OWL and Conceptual Graphs and thus leverages results of more than 20 years of research and implementation in that branch of knowledge-based systems including a graph projection algorithm that provides an ontology-guided search operator.

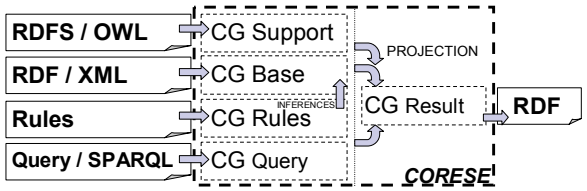

Figure 1. Corese principle

In the corporate memories developed so far, the annotations generally describe documentary resources or corporate structures, but, when relying on schemata as the ones surveyed in section 2.3, these annotations can describe web services available online (intranet, extranet, Internet). This means that Corese allows us to automate the identification of web services available to a user. Following a service-oriented architecture and a find-bind-execute schema [13] Corese fits well in the picture of semantic web services as a semantic registry.

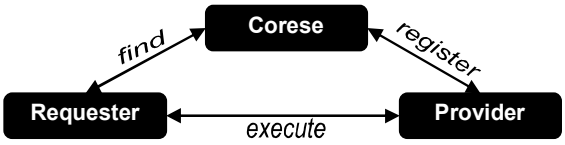

Figure 2. Corese as a semantic registry

In this new architecture, we moved from text-based UDDI search to the semantic search engine Corese to solve queries on the descriptions of the services, taking into account the ontologies used to characterize them and leveraging their semantics when solving a query. For instance let us consider the following query:

```
?s    rdf:type               ex:Directory
?s    service:describedBy    ?p
?p    proc:hasInput          ?param
```

This query will retrieve any instance of directory services (including instances of sub-types like yellow pages, LDAP, etc.) together with their input parameters. With this architecture, annotations of services corresponding to corporate applications are stored in the corporate semantic web. Then, the indexed services can be automatically discovered and dynamically invoked.

3.2. Semantic web portal to corporate services

Our current implementation is embedded in a semantic web server architecture we designed to accelerate the development of semantic web based portals. In this 3-tier architecture, we added a web application to manage semantic web services. It enables us to extend the portal with accesses to corporate (and external) services using main functionalities: automatic web service discovery, dynamic invocation of web service, and web service composition. All the components of the portal rely on Corese to access service ontologies and annotations. When a service is selected by a user, we dynamically generate a form offering an interface to call the service; on submitting the form, the inputs are used to generate a dynamic client and call the web services. The output is then formatted as a web page. Figure 3 shows two windows:

- A window in the background showing the result of a query that retrieved a service description. This service is a mail-sender with a number of inputs;
- A second window appeared when the service was selected and provides a form to specify the inputs. Once submitted, this form triggers a call to the web service which is then dynamically executed and displays the outputs.

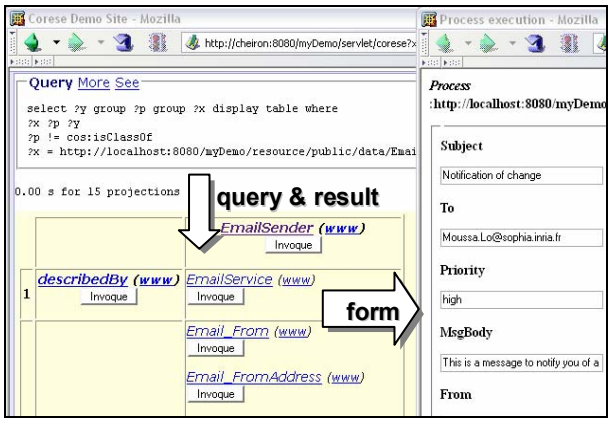

Figure 3: Discovering & invoking a service

We also give the ability to export the output as an XML document. This can be interesting for instance to integrate the results in other applications.

3.3. Discovering sequential compositions

We also introduced means to discover limited compositions of services that match a user's request expressed in terms of available inputs and desired outputs: Corese provides the possibility to search for resources linked by a path of relations. For instance the query ?x cos:Property[4] ?y looks for an oriented path between two resources with a maximum of 4 relations. This feature of Corese was designed to explore the relations between two resources in a knowledge base (e.g.: to discover acquaintance networks). Applied to web services it can be used to discover a special type of composition: sequences *i.e.* a succession of services combined one after the other through their input and output types. This is used to match users' requirements when no single service directly matches available inputs and desired outputs.

We formally defined what it means for two services to be "composable" in a sequence. This is done through a production rule encoding the sufficient condition of the "composable" relation. To represent this relation, we extended OWL-S with a property named "composable" for the Process concept. The rule defined below uses this new property and defines two services s1 and s2 as "composable" when the input of s2 and the output of s1 are ontologically compatible *i.e.* the type of the output of s1 is the same or a subtype of the type of the input of s2:

```
<cos:rule>
  <cos:if>
    ?s1 rdf:type proc:Process
    ?s2 rdf:type proc:Process
    ?s1 proc:hasInput ?input
    ?s2 proc:hasOutput ?output
```

```
        ?s1 != ?s2
        ?input proc:semanticType ?inType
        ?output proc:semanticType ?outType
        ?outType rdfs:subPropertyOf ?inType
      </cos:if>
      <cos:then>
        ?s2 proc:composable ?s1
      </cos:then>
    </cos:rule>
```

Applied to the knowledge base this rule generates the couples of "composable" services. Since this rule allows us to identify all the services that can be composed together we can then express queries like "Find all sequences of services having as input a BookName and as output a BookBuyNotification":

```
?s1 all::proc:composable[2] ?s2
?s1 proc:hasInput ?param1
?s2 proc:hasOutput ?param2
?param1 proc:semanticType c:BookName
?param2 proc:semanticType c:BookBuyNotification
```

Figure 4 shows an answer to this query with the Book services coming with the OWL-S API. We obtained the composition of the three services:

BookFinderProcess (input : BookName, output : BookInfo)

BNPriceProcess (input: BookInfo, output : BookPrice)

BookBuyProcess(input:BookPrice,output:BookBuyNotification)

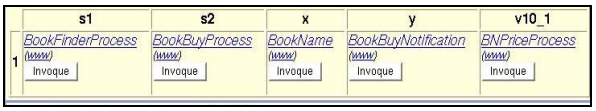

Figure 4. Example of a sequence of services

We provide the ability to save sequences of services as OWL-S composite processes. These composite services can then be retrieved and executed like any other corporate web service. This feature can be used by an IT manager to create, save and propose new services from existing ones.

4. Composing services and knowledge

4.1. Mapping input types to queries

Since we are in a semantic web environment, "knowledge is everywhere" and thus one can use the knowledge stored in the corporate memory to populate, in an automatic way, the service inputs during execution. This idea was suggested by a previous work on context-aware service invocation [7]. The implementation is in three steps:

1. We associate to service inputs a predicate from a domain ontology by means of the semanticType predicate. This means that candidate values for an input can be found from values of this predicate.
2. We formally define these predicates using rules allowing us to generate dynamically the needed information from the memory.
3. When generating dynamically an invocation form we also extract from the corporate memory the information to (pre)populate the service inputs.

Let us consider the example of the service with an input associated to the property EmployeeName. A rule allows us to generate the candidate inputs from the corporate memory annotations; it defines a sufficient condition of the predicate EmployeeName. Then, by using the generic query (select ?value where { ?x *semanticType* ?value}), we obtain the triples generated by the previous rule and we generate a dropdown box with the name of the employees to pre-populate the input of the service. To summarize we can select a service from the result of a query on the directory, then we can pre-populate the input form and finally we display the result of the invocation. Again, every step of this process (semantic rules and queries) leverages the ontological reasoning.

4.2. Corese as a semantic web service itself

By wrapping a semantic search engine in a semantic web service, one can provide new capabilities in compositions: (1) to use the result of a query over the corporate memory as a service input; (2) to use a service output to add knowledge to the memory. In order to do so, we provide the ability to compose a service which wraps a corporate application with a Corese semantic web service of two kinds:

- A Corese SWS which takes a query as input and gives the query results as output; this service can be composed with any other one to get knowledge from the memory in order to associate a Corese query and the service input;
- A Corese SWS which gets an RDF annotation as input, stores and loads the annotation into the memory; this service can be composed with another service to transform its output into knowledge for the memory. We introduced an auxiliary XSLT service to transform outputs into RDF/XML.

We tested the following scenario. When someone wants to know the email of an employee whose name he knows, he can use directly the service wrapping the LDAP application. Now, assume he is searching the email address of the assistant of a given team but he doesn't know the name of this person. He can perform a query (Find the name of the team's secretary) over the memory and give the result directly to the service

providing emails. He can also save this new knowledge (the email) in the memory.

5. Conclusion

In this article we presented an experiment in integrating enterprise applications as web services in an intranet relying on semantic web frameworks. We chose to focus on a clearly identified family of scenarios: the integration of enterprise applications in an intranet. We have used Corese, a semantic web search engine as a semantic registry. This allowed us to prototype a semantic web portal embedded in the Corese semantic web server. The portal offers (i) automatic discovery, (ii) dynamic invocation, (iii) interactive composition and (iv) discovery of sequences of corporate and public web services.

An original contribution of this experiment is the composition of corporate web services with knowledge from the corporate memory: (i) services inputs types are mapped into queries and (ii) the semantic search engine is turned into a web service to connect corporate applications with the memory knowledge. This allowed us to differentiate between needed functionalities and prospective ones and to identify the layers of the semantic web services stack that we needed first.

A typical question, for instance, is the one of offering "manual vs. semi-automatic vs. fully automatic composition and invocation of services". In our scenarios, we do need to provide high-level functionality through dynamic integration. However we have not found ergonomic ways to describe and decompose service needs to support fully automatic composition. In addition, such functionality seems to rely a lot on domain knowledge, and more over we think that, as claimed in [8], in many cases, users will want to control the composition process, influencing the service selection. In the actual state of the art, we found it more realistic to consider, for instance, the request from business managers to be able to implement business workflows in flexible (declarative) manners above the classical web services architectures.

Finally we are currently studying the interaction and integration with emerging semantic web extensions such as: SPARQL query language and protocol and SWRL rule description language. We also consider the problem of dynamically generating ergonomic user interfaces to semantic web services: web services are primarily designed for B2B programmatic interactions but the services or their compositions are called by users. Since their discovery, composition and invocation are dynamic their use will require dynamically generated *ergonomic* user interfaces.

6. References

[1] Akkiraju, R., Farrell, J., Miller, J.A., Nagarajan, M., Schmidt M-T., Sheth, A., Verma, K. Web Service Semantics - WSDL-S, Technical Note, V 1.0, April 2005

[2] Bussler, Fensel, Maedche, A Conceptual Architecture for Semantic Web Enabled Web Services, ACM 2002.

[3] Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F., Approaches to Semantic Web Services: An Overview and Comparisons, ESWS'04, 2004.

[4] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Querying the Semantic Web with the Corese search engine. In Proc. of the 16th European Conference on Artificial Intelligence ECAI'2004, IOS Press, p. 705-709.

[5] Fensel, D., Bussler, C., The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications, Vol. 1., 2002

[6] Gandon, F., Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web, PhD Thesis in Informatics, 7th of November 2002, INRIA and University of Nice - Sophia Antipolis

[7] Gandon, F. and Sadeh, N., Semantic Web Technologies to Reconcile Privacy and Context Awareness, Web Semantics Journal. Vol. 1, No. 3, 2004.

[8] Kifer, Lara, Polleres, Zhao, Keller, Lausen, Fensel, A Logical Framework for Web Service Discovery, workshop Semantic Web Services: Preparing to Meet the World of Business Applications, at ISWC, Hiroshima, 2004.

[9] Kim, J., Gil, Y., Towards Interactive Composition of Semantic Web Services, First International Semantic Web Services Symposium, AAAI, March 2004.

[10] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K., Bringing Semantics to Web Services: the OWL-S Approach, SWSWPC'04, LNCS n° 3387, 2004.

[11] McIlraith, S., Son T. C., Zeng H., Semantic Web Services, IEEE Intelligent Systems, 16(2):46-53, 2001.

[12] OWL-S Specification, http://www.daml.org/services

[13] Qusay H. M., Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), April 2005

[14] SWWS Project, http://swws.semanticweb.org

[15] WSMO at http://www.wsmo.org/2004/d2/