WSDir: a Federated Directory System of Semantic Web Services

Michael Schumacher
University of Applied Sciences Western Switzerland, Institute of Business Information Systems, CH-3960 Sierre, Switzerland
michael.schumacher@hevs.ch

Tim van Pelt, Ion Constantinescu, Alexandre de Oliveira e Sousa and Boi Faltings
Artificial Intelligence Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
boi.faltings@epfl.ch

Abstract

This paper presents a federated directory system called WSDir, which allows registration and discovery of semantic web services. The typical use of this directory system is in a context where ubiquitous business application services should be flexibly coordinated and pervasively provided to the mobile user by intelligent agents in dynamically changing environments. The system has been modeled, designed and implemented as a backbone directory system to be searched by an infrastructure made up by such kind of agents coordinating web services. The system is modeled as a federation: directory services form its atomic units, and the federation emerges from the registration of directory services in other directory services. Directories are virtual clusters of service entries stored in one or more directory services. To create the topology, policies are defined on all possible operations to be called on directories. For instance, they allow for routed registration and selective access to directories. WSDir has been applied as a backbone in the trials of an ehealth emergency application.

1 Introduction

UDDI has become the de-facto standard to provide a general framework to describe and discover services and Web service providers. More specifically, WSDL descriptions of web services can be mapped to UDDI data structures, allowing to find web service descriptions using the standard UDDI query interface. Within the academic world, a number of approaches exist that try to build semantically enhanced discovery components on top of UDDI. [3] augments the standard UDDI registry APIs with semantic annotations. [5] uses a set of distributed UDDI registries as a storage layer, where each registry is mapped to a specific domain based on a registry ontology.

This paper presents a new federated directory (or registry) system called WSDir[1], which allows registration and discovery of OWL-S semantic web services [4]. Our directory system is used in the CASCOM platform[2] where ehealth services should be flexibly coordinated and pervasively provided to the mobile user by intelligent agents in dynamically changing environments. We modeled, designed and implemented WSDir as a backbone directory system to be searched by an infrastructure made up by such kind of agents coordinating web services. This agent infrastructure therefore is deployed on mobile users: it queries our directory system for OWL-S service descriptions, composes them to achieve a target higher functionality and executes them. WSDir has also been used in the trials of an ehealth emergency application.

We followed specific requirements for designing WSDir: i) it should have itself a Web service interface to be universally invoked; ii) it should be distributed; iii) the construction of the network should induce minimal overhead and should be scalable; also, the network should be robust to changes in topology and the number of interactions with the system; iv) the directory should allow a great number of services to be registered, and this in a very dynamic way, including lease times.

These requirements lead us to model WSDir as a federation: directory services form its atomic units, and the federation emerges from the registration of directory services in other directory services. Directories are virtual clusters of service entries stored in one or more directory services. To

[1]This work has been supported in part by the European Commission under the project grant FP6-IST-511632-CASCOM.
[2]http://www.ist-cascom.org

create the topology, policies are defined on all possible operations to be called on directories. For instance, they allow for routed registration and selective access to directories.

The paper is organized as follows. In Sec. 2, we explain the service entries of semantic web services that can be stored in WSDir. Sections 3 to 6 explain the architecture of WSDir by presenting *directories*, *directory services*, *directory operations*, and *policies*. In Sec. 7, we give a concrete network architecture example based on our federated directory system. Finally, Sec. 8 concludes the paper.

2 Service Entries

We describe services using the Web Ontology Language for web services, OWL-S [4]. The directory system stores all descriptions translating from the OWL-S description towards a representation described in SL0 [2] that will include the original service description. Additional fields contain information that is taken from the service description in OWL-S.

Internally, WSDir stores service entries in the FIPA SL0 description language. The internal representation contains a subset of the information provided in the original service description. This information can be used to find matching services in the directory. In addition, the original service description in OWL-S is stored in a separate slot. This field is used to retrieve the original description, e.g., to retrieve the grounding(s) of a service at service execution.

A service entry in WSDir contains the following information: i) *ServiceCategories*, and entry in some ontology or taxonomy of services; ii) *ServiceProfileURIs*, a set of profile URIs that point to an externally stored, but (web-)retrievable service profile; iii) *ServiceProcessURI*, a process URI defined in the service description (can be empty); iv) *ServiceGroundings*, a set of full-text service groundings (can be empty); v) and *OWLSServiceDescription*, an original OWL-S service description as a full-text entry.

3 Directories

A directory comprises a set of service entries which are managed by a collection of one or more directory services. All service entries, including directory service entries, are registered at a directory service as belonging to a specific directory. Such, directory services can form an arbitrary organisational structure (peer-to-peer, hierarchy etc.). Specifically, a directory can contain other directories, and a directory is supported by one or more directory services. This is used to characterize directories by two different types of interactions. In *Client-Directory* interactions, clients are registering, deregistering, and querying the directory interact with directory services supporting the directory. They may or may not have any idea about the internals of the directory. In *Director-Directory* interactions, directory services supporting the directories interact with one another to perform the internal management of the directory (data propagation, federated queries, managing the membership of the directory service group managing the directory).

The first interaction style (*Client-Directory*) is part of the base for the creation and maintenance of a *domain directory*. The second interaction style (*Directory-Directory*) is the base for the creation and maintenance of a network directory. As directories can be used to support other directories they are seen as organizational structures. These concepts will be elaborated and exemplified in Sec. 7.1 on network topology.

4 Directory Services

Directory services provide a Web service interface to a repository that holds service entries. The service entries in this store are all registered as belonging to a certain directory. The directory service forms the atomic unit of the directory federation.

The directory service allows clients to register, deregister, modify and search registrations in its repository. These registrations include service descriptions of services offered by clients as well as profiles of other directory service. By registering directory services in other directory service stores, the system becomes federated.

Figure 1 visually summarizes the relationship between service entries, directories and directory services. In the illustration, the directory service holds regular service entries and a directory service entry belonging to a Hospitals directory as well as entries belonging to an Insurers directory. Both directories are contained in the Body directory, which in turn is contained in the all-encompassing "." directory.

5 Directory Operations

The directory is able to handle five types of operations. The *register* operation enables a client to register a service description entry into a *directory* for a time period given by a *lease-time*. The *directory-service* can return a *redirect* message pointing to another directory where the client could try to register its object. The *deregister* operation deregisters a service that previously has been registered identified. The *modify* operation allows modifying a registered service. The *get-profile* requests meta-information on a directory service such as its name and the policies governing the operations.

The *search* operation looks in the directory for services that match a template. The request can possibly be for-

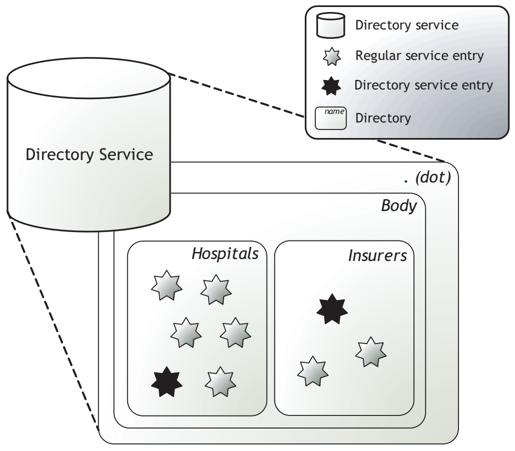

Figure 1. Directory system concepts

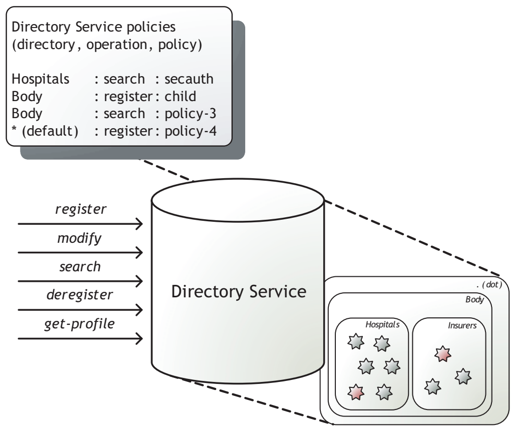

Figure 2. Example use of policies

warded to supporting directories, depending on the implemented search policy at the directory. As the internal service descriptions are expressed as SL0 expressions, the structured element used in the operation is also an SL0 expression. *Search-constraints* can be specified in order to restrain the search. A *max-time* specifies a deadline by which the constrained search should return the results. A *max-depth* specifies the maximum depth of propagation of the search to federated directories. A *max-results* element specifies the maximum number of results to be returned.

6 Policies

Directory services employ *directory policies* to regulate the operation of directories. Policies are defined per directory service in the *directory service profile* and determine the behaviour of a specific directory. Two types of policies can be distinguished. *Pro-active policies* policies are typically used for internal management of the directories. A policy may be attached to a directory to establish the number of times per hour data is propagated within the directory, how often old entries are removed etc. *Reactive policies* assign a behaviour to combinations of directories and operations. The policies are executed whenever a bound operation is called. They are defined as a triple: *(directory name, operation, policy)*.

Policies can also be applied to the default directory named ”*” which matches all directories that don’t have a policy explicitly assigned.

From the consequent application of policies, the network topology emerges. Policies can for example define how much entries can be registered per directory, which directories can be searched by which clients, and which types of services will be accepted. Each directory service can define its own policies or use one of the pre-defined policies. The only requirement on the part of a policy is that it can be executed.

A straightforward example of a pre-defined policy is the *child/sibling* policy: this policy forwards all operations to both the known children (directory services registered in the service store) as well as its known siblings. The list of siblings is obtained by querying the parent directory service at which it has registered itself.

Figure 2 shows an example of the reactive policies that are assigned to the various directories this directory service supports. In the illustration, when a client calls the search operation on the ”Hospitals” directory, a policy called ”secauth” will be applied. Such a policy could for example require the client to authenticate itself before it is allowed to query the directory.

7 An Example of a Network Architecture

WSDir allows to setup flexible distributed directory systems, especially thanks to the mechanism of policies. We present here a specific application of WSDir to build a network of directory services which are modelled as a virtual tree with multiple roots. This topology has been applied in the trial of a medical emergency usecase scenario.

7.1 Network Topology

The nodes of the tree are made up by the individual directory services. This hierarchical structure with multiple entry points effectuates no replication or data caching within the directory: each directory service is responsible for registrations made in its local store. Since there is no replication, directory services will forward queries to other directory services. Query messages are checked for duplication: a given directory service will handle only the first of several identical query messages from several sources and discard the others while returning the appropriate failure message. Furthermore, dentical results may be returned by one or more directory services for a single query. This enhances the robustness of the federation at the cost of shifting the burden of filtering the results to the client.

In the network, we distinguish two types of directories: network directories and domain directories. *Network directories* are a reserved set of directories that are used for the construction of the network. In a directory federation, we can distinguish three different network directories:

- *Hidden* network directory: the directory service that forms the root of the federation by registering the top-level nodes of the network. Neither the directory service nor its registered services will be visible to the other nodes in the federation.

- *Top* network directory: visible to the network as being one of the roots of the federation multi-rooted tree. A directory service with this role could typically serve as a bootstrap service to leaf directory services. These services constitute the Top network directory.

- *Body* network directory: regular directory services that form the body of the multi-rooted tree. These directory services provide the interface to the directories that will contain most of the service registrations in the network.

Domain directories emerge from the registrations of service descriptions at the directory services that make up the directory system. By definition, domain directories are contained in the "Body Members" directory.

Hereafter, we explain the network directories in more detail. The above-mentioned roles and their place in a WSDir federation topology are depicted in Fig. 3.

Hidden network directory The Hidden directory service node responds only to requests coming from Top directory services and exclusively regarding the "Top Members" directory. For that it holds authentication information regarding a pre-configured list of possible top-level nodes and uses this information together with information in the identification information field of the requests. Search queries are not propagated to other directory services. The location of the hidden directory service node is pre-defined.

The existence of this domain ensures that directory services belonging to the "Top Members" directory know of their respective existence and such makes sure that every node in the network can be reached if needed. The hidden directory service is only used for bootstrapping of the top member directory services. After the system has been initialized, the hidden directory service will only be used by the top member directory services to poll to see whether

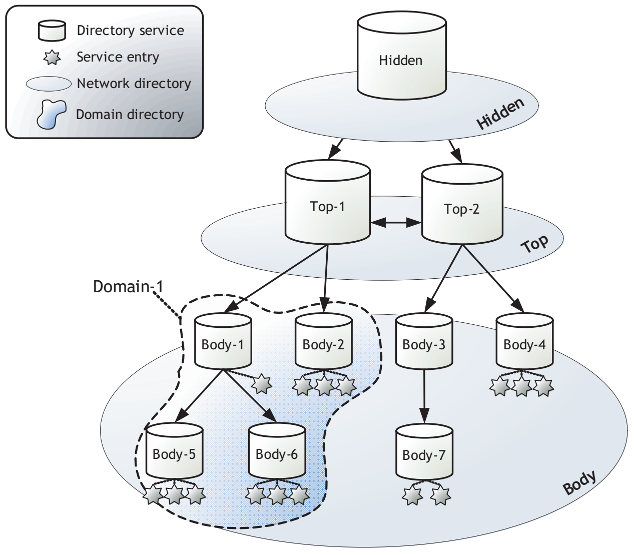

Figure 3. Network Topology

new directory services have been added to the "Top" network directory. Thus, queries, registrations and other requests do not go through the hidden directory service, but will be directed at a top or body member directory service.

Top network directory Upon start-up the Top directory services join the network by registering their directory-service-profiles inside the "Top Members" directory of the Hidden directory service. Also they keep track of other Top directory services currently members of the "Top Members" directory by continuously polling the "Top Members" directory of the Hidden directory service for directory-service-profiles entries. In the case that the Hidden directory service fails the Top directory services should continue to use the last retrieved membership information until the Hidden directory service will be back online. In terms of response to registration requests from clients, a Top directory service allows only for the registration of directory-service-profile entries inside the "Body Members" directory. Once the number of registrations that are hold locally goes over a given threshold, the Top directory service returns redirect messages pointing requestors. Normal directory services directly registered with the current directory service. For any other kind of registration requests, the Top directory services will issue redirect responses pointing at Normal directory services registered with the current directory services or other Top directory services that might be more appropriate for use. Top directory services will respond to all search requests by first trying to fulfil them locally and in the case that more results can be returned (the value of the max-results parameter in the search-constraints object has not been reached yet) it will forward the query to all other Top directory services members of the "Top Members” network directory. For determining the other Top directory services members of the ”Top Members” directory, the information from the last successful polling of the Hidden directory service will be used. Top directory services will respond with a failure to all other kinds of requests.

Body network directory At start-up, a Body directory service will try to register its directory-service-profile in the ”Body Members” directory of a Top directory service randomly picked from a pre-configured list of Top directory services. If the Top directory service cannot be reached another one is randomly picked until either the joining procedure (see next) succeeds or the list is exhausted. In the latter case the directory service will report a join failure. The directory service will follow redirect responses until the entry is successfully registered with a directory service (either Top or Body). Upon failure of the directory service used for registration the current directory service will sleep for a random time period and after than will re-initiate the initial join procedure.

For other Body directory services that try to register directory-service-profile entries inside the ”Body Members” directory a Body directory service will act as a Top directory service: once the number of registrations that are hold locally goes over a given threshold the Body directory service will return redirect messages pointing requestors to child Body directory services directly registered with the current directory service.

A Body directory service will respond positively to all other requests. In particular it will forward search queries for which it could return more results than locally available to directory services locally registered in the ”Body Members” directory.

7.2 Network Construction

At boot time, the directory makes use of a pre-defined network configuration to create a network topology. The configuration specifies management and data relations between members of the network.

Some of the network nodes might have fixed well-known addresses in order to serve as bootstrap hosts for other directory services. Depending on their role, different parts of the network are visible to bootstrapping directory services.

As mentioned before, in a typical setting, the node at the highest level will be hidden to all nodes not belonging to the ”Top Members” directory.

The process of directory service registration is equivalent to the process of registering regular service entries. Directory services are registered invoking the same register method as is used for registering regular services.

7.3 Used Directory Policies

WSDir employs a set of pre-defined pro-active policies, mainly for routing purposes.

For example, a registration request directed at a directory belonging to the ”Hidden” network directory triggers the application of the *ChildRegisterPolicy*. The service registration request is forwarded to its known children, which themselves apply (by default) the *ChildSiblingRegisterPolicy*. This in turn selects the least loaded directory service among its children and among its siblings to put the service entry in its store.

The procedure for a *search* operation is similar. Requests directed at a directory service either belonging to the ”Hidden” network directory or to the ”Top” network directory will forward the search request to its registered children and its known siblings. The directory service instances belonging to the ”Body” network directory apply the *DefaultSearchPolicy*, only searching their local store.

For the *modify*, *deregister* and *get-profile* operation, the policies that are assigned to the operations also depend on the network directory the directory service instance belongs to.

7.4 Examples of Network Interactions

The following section describes one example of the policy-governed query operation of the network of directories as presented previously. The visible network is formed from a number of ”well-known” top nodes (Top-1, Top-2, Top-3) with a fixed name and transport address (but which can possibly fail) and an arbitrary number of leaf nodes which are organized in a tree topology with one of the top nodes as root.

We illustrate the process of query resolution in Fig. 4 and 5: a client issues a search request for service profiles matching a template in the *Hospital* domain directory. i) First, the client issuing the query randomly selects one of the top level nodes (*Top-1, Top-2, Top-3*) (in this example, *Top-2* is picked). ii) The *Top-2* directory service forwards then the query to its siblings. iii) This allows directory services that have an entry for the *Hospital* domain directory in their service profile to propagate the query down. iv) Then, to guarantee full query resolution, the query is forwarded to all directory services that are known to store entries for the *Hospital* domain directory. v) Finally, upon finding results, the nodes holding the results will send a message back (depicted by ”R=x” in the figure, where x denotes the number of matched services) to the directory service it was queried by, until it reaches the original requester. In this case, the matching service profiles in the directory services supporting the *Hospital* domain directory will be returned.

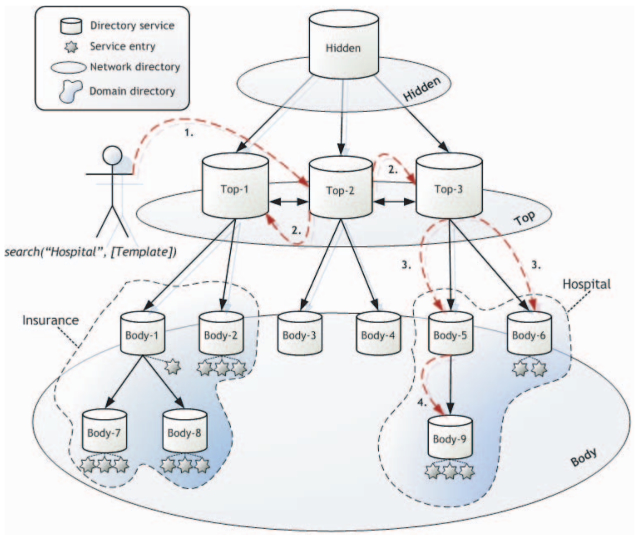

Figure 4. Query Resolution (1)

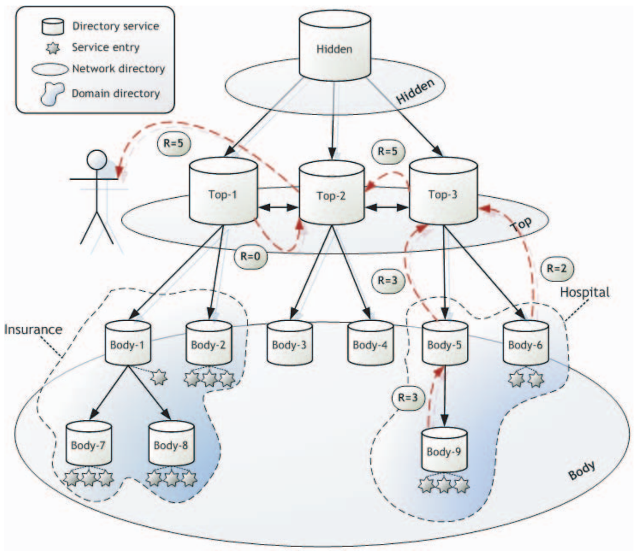

Figure 5. Query Resolution (2)

8 Conclusion

WSDir has been tested thoroughly in a real distributed setting spread over different countries. The system has proven to be scalable and very stable. We have integrated the system in a use case scenario in the pervasive ehealth domain uses WSDir as its backbone. Future work could enhance security.

From the security and privacy-awareness point of view, we currently employ standard security mechanisms for accessing the directory services. In particular, if a directory service requires protecting messaging from overhearing or if it would require privacy sensible data as parameters, the access to this web service will be based on HTTPS. In cases where no HTTPS is available, we could couple WSDir with Guarantor agents [1] spread in the architecture in order to provide a secure tunnelling between secure ACL messages and HTTPS.

Another improvement could be to define security measures directly within the directory system by defining specific policies. A policy can be employed to restrict the right to perform a certain operation on a directory to only those clients that can provide the right credentials. Using this method, registration of services to a directory and search operations on directories can be restricted. For example, a directory service that does not forward any queries pertaining to a "Hospital" domain directory will simply return its entries for the domain and nothing more. This would be completely transparent to the requestor, as its view of the network topology is determined by the application of policies of the directory services underneath it.

References

[1] R. Bianchi, A. Fontana, and F. Bergenti. A real-world approach to secure and trusted negotiation in mass. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1163–1164, New York, NY, USA, 2005. ACM Press.

[2] Foundation for Intelligent Physical Agents. Fipa sl content language specification, December 2002.

[3] T. Kawamura, J.-A. D. Blasio, T. Hasegawa, M. Paolucci, and K. P. Sycara. Preliminary report of public experiment of semantic service matchmaker with uddi business registry. In *ICSOC*, pages 208–224, 2003.

[4] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The owl-s approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004.

[5] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Journal of Information Technology and Management*, 2004.