A facilitator to discover and compose services

Oussama Kassem Zein
ENST Bretagne, Dept. LUSSI
Technopôle Brest Iroise, BP 832, 29285 Brest Cedex, France
Email : Oussama.Zein@enst-bretagne.fr
Tel : + 33 2 29 00 12 85 Fax : + 33 2 29 00 10 30

Yvon Kermarrec
ENST Bretagne, Dept. LUSSI
Technopôle Brest Iroise, BP 832, 29285 Brest Cedex, France
Email : Yvon.Kermarrec@enst-bretagne.fr
Tel : + 33 2 29 00 12 85 Fax : + 33 2 29 00 10 30

Abstract

In this paper, we present our approach of facilitator that allows automatic service composition in distributed systems. We propose a facilitator based on ontologies and knowledge representation that allows clients to discover a service and to get the result of the service execution without direct interactions with their providers. We extend the functionality of the facilitator to be a composition engine and we show how it composes services to satisfy the client's requests.

KEYWORDS: composition of services, facilitator, ontology, service description.

1 Introduction and Context

Facilitator is an approach of service registry used in multi-agents systems [8]. It is a medium that ensures the communication between agents. Agents may register their services to the facilitator with all the information required so that a call can be made automatically by a machine, or query it to find out what services are offered by other agents. Agent is responsible to provide information related to service description like service type and so on. An agent can also deregister or update the service descriptions.

Our proposal is to bring the approach of facilitator into the context of service discovery and composition in distributed systems. Many different approaches for service discovery have been developed, like the UDDI [1] of web services, the naming service and the trading service of CORBA [10]. UDDI and the trading service are advanced directories which allow services to be discovered at run-time via an attribute based (or yellow page) style of search. They play a major role in distributed systems and service oriented architecture (SOA) since they enable to link clients to servers.

We showed in [13], [14] that the service description plays a central role and that it was necessary to take into account various levels of description in order to retrieve the service and to access it. Once a service is properly described, it can then be indexed and users (either human or software) can look for it, and then access it. We developed a trader based on ontologies and integrated it as an OMG CORBA component in the ORBACUS platform and then as a web service in J2EE [12].

Based on our approach of service description and trader [13], we have developed an approach of service composition [15]. In this approach, we are based on SDL [7] and Interface Automata [3] to describe the behavior of the service with an automaton by inputs/outputs and the composition of several services is made by combining their inputs/outputs. This appraoch is simple and powerful because the services are easily questionable. With the trader, the client must invoke the composed service, i.e., all the components services. But the client may not be familiar with the service composition (like elderly people, disabled people, novice, etc.) and may want only the final results without dealing with the intermediate steps. So, the idea is to alleviate clients from numerous interactions (with the trader and the service providers). In this context, we have chosen to use the facilitators. They allow the indirect interactions between clients and servers. So, the client queries a service from the facilitator. The latter searches, invokes the service and returns the result to the client. In this context, the facilitator can take into account the client profile to find and propose a sequence of call of services to satisfy the client requirements. So, the idea is to extend the functionalities of the facilitator to compose services by allowing the execution of the composed services without intervention of the clients.

In the second section of this paper, we present our approach of facilitator based on ontologies allowing the service discovery and automatic call. In the third section, we extend the functionalities of the facilitator to be a composition engine that allows the service composition when a service search is unsuccessful without intervention of clients. In the fourth section, we present our approach of user profile which can be used to adapt services to the needs of different users and allows to personalize the service search. In the fifth section, we present the related work. Finally, the conclusion raises issues and presents our future work.

2 Our approach of facilitator based on ontologies

Our approach for implementing the facilitator relies on ontologies. We have selected ontologies because they make it possible to provide a shared consensus that can help share/reuse. We have selected Ontobroker [6] as our support engine for ontologies (see figure 1). It allows us to query, store, delete and modify information in the ontology. We create an ontology describing the service properties defined in the metadata model that we have proposed [12].

2.1 Tools Implementation

We present the facilitator interfaces that we have developed to manage service ontologies we have proposed. They allow clients/servers to query/add services through the ontology.

- Importing function : It is used by clients. It takes a query written in a logic language (e.g., F-Logic) [9] as the input, makes a request to Ontobroker for the service required, invokes the service and returns the result to the client. It allows client to get the result in an implicit way without direct interaction with the server.

- Exporting function : It is used by service providers. It allows a fact to be added to the ontology. A service offer is an instance of a service. It is described by a fact. The fact includes the service properties and the service address (like WSDL file address with web services, IOR address with CORBA) which is necessary to access and communicate with the service.

- A function is defined to delete a fact from an ontology.

Finally, we declare an object which implements all these functions and we start it on a server. This object represents the facilitator. The clients and the servers use this object to contact Ontobroker and to add/retrieve services through the ontology.

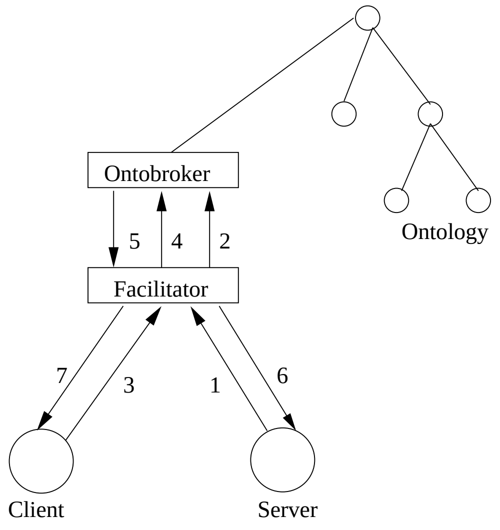

Figure 1. Interactions between the facilitator, Ontobroker, client and service provider

3 Extension of facilitator functionalities to compose services

The composition of services is critical since it provides novel services and novel functionalities to the client. By combining services, the client can obtain a desired output from a given input not provided by any service but by several combined services. In our approach, we consider that two services can be composed if an ouput of one is equal to an input of the other one. We extend the functionality of the facilitator to be a composition engine. When a client queries a service from the facilitator and if the service doesn't exist, the facilitator tries to combine two or several services based on their inputs/outputs to satisfy the client's request. Figure 2 describes an example indicating how the facilitator searches, composes and invokes services.

The client queries a service from the facilitator by indicating the desired input/output. For example, he/she queries a service that allows to transform a “tex” file into “pdf” file. The facilitator searches the required service in its repository (ontologies). If the service doesn't exist, it tries to compose services by comparing their inputs/outputs. The facilitator searches if a service has an input “tex”. For example, it obtains a service (S1) having “latex” as type which takes as input “tex” and provides as output “dvi”. It searches the services taking as input “dvi”. For example, it obtains a service (S2) having “dvips” as type. It verifies if this service provides “pdf” as output. But this service provides “ps” as output. So, the facilitator continues and tries to find services taking “ps” as input. It obtains a service (S3) having “ps2pdf” as type which provides “pdf” as output. So, the facilitator can compose S1, S2 and S3 to satisfy the client request. It invokes S1 by giving the “tex” file of the client as input. Then, it invokes S2 by giving the “dvi” obtained

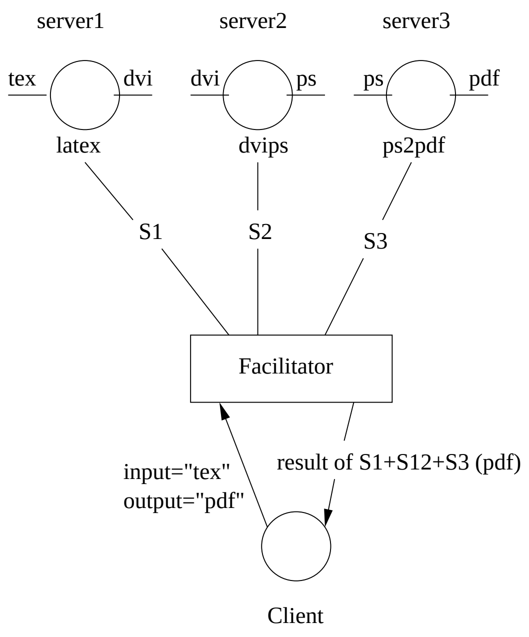

Figure 2. A facilitator as a composition engine

from S1 as input. Finally, it invokes S3 by giving the "ps" file obtained by S2 as input and it returns the "pdf" file to the client.

The client doesn't interact directly with the servers to invoke the services and doesn't interact with the facilitator to find the services to be composed, but he obtains the final result. This does not require clients to be familiar with the service composition like the elderly and disabled people. In this context, we can define a profile for each client that includes his desired and required service properties to personalize the service research by the client (see next section). These profiles can be taken into account to find, compose and invoke the services that satisfy the client requirements. So, due to the use of the facilitator, the composition becomes implicit to the clients.

In a previous paper [15], we have proposed three models of service composition : static that allows to combine service offers defined at compile-time, semi-dynamic that allows to combine service types (a service offer is an instance of a service type) defined at compile-time and dynamic which allows to combine services defined at runtime. So, we can use these models in our approach of facilitator to compose services.

4 Personalization : user profile

For adapting and personalizing services to the needs of different users, so-called "user profile" are frequently defined and used. User profiles are collections of information and assumptions about behaviors and preferences of individual users: they are used in the services adaptation and personalization process. A user profile helps to adapt services to the user needs. It holds the user preferences: it can contain critical information about the user like its location, its language, its experiences in the domain, etc.

When a client calls the facilitator to query services and if there are numerous offers, the facilitator uses the client profile as an additional criterion to extend the request. This allows the client to filter the results and to get the services that satisfy its profile and its preferences.

We consider that the user profile contains properties with their default values. When we extend the service query to include the properties contained in the user profile and if some properties have values in the user request different from those in the user profile, we use the values which are required in the user request as priority to search the services. This is possible by using "or" between the properties existed in the user profile in the extended request:

initial_request and [p1->>v1 or p2->>v2 or p3->>v3 or ...].

Where p1, p2, p3 ... the properties contained in the user profile with their values v1, v2, v3 ... respectively.

In addition, we have used a scoring approach to order the results before returning them to the client. Each property in the user profile has a value and an importance degree. The sum of these importance degrees is equal to 1. For each property in the user profile, we calculate its partial score by the following expression:

$$V_i = \frac{A_i}{\max_i A_i} P_i$$

Where V_i is the partial score of the property "i", A_i its number of occurrences in the results (returned offers) and P_i its importance degree. We count the number of occurrences of a property "i" in the returned offers if only its value coincides with that in the user profile.

Each service offered has a score which is the sum of the partial scores of its properties included in the user profile such as their values coincide with those which are in the user profile. The facilitator returns the services to the client ordered by the score.

5 Related Work

Many different approaches for describing and discovering services have been developed. For example, DAML-S [2] provides a set of characteristics that allows describing a service. This description addresses only the static properties of a service. It does not describe the behavior of a service and its interface. In our approach of service indexing, we have used this description as a reference to characterize a service by static properties. Our approach is more powerful than DAML-S since it provides complementary properties describing the service by its behavior and its interface. These properties can be used by clients to query and find a service in distributed systems.

To discover a service, we can select for example UDDI [1] which, is a registry that allows a Web service to be discovered via a yellow page style of search. It allows a service to be discovered by querying only its static properties. It provides a static schema for service description and the service provider cannot modify this schema or create databases to advertise its services offers. Then, in our approach, each service provider can define its databases and can advertise its service offers by using their static properties, their behaviors and their interfaces. The clients use the same interface to query these three levels of service description. So, the service indexing and discovery become more sophisticated.

Many different approaches for service composition have been developed like [11], [5], [4] but the client must invoke all the service components of the composed service. In this context, the client must be familiar with the service composition. These approaches don't resolve the problem of the service composition complexity. Our approach for service composition using facilitator is innovative since it allows services to be composed automatically in an implicit way to the clients. It can be used by clients that are not familiar with the service composition and aren't interested by the intermediate steps of the service composition process. Our approach of semi-dynamic composition model is the first one that exceeds the performance and complexity problem of the dynamic composition model [15].

6 Perspectives and conclusions

In this paper, we have presented a facilitator we designed and developed : it is based on ontologies and knowledge representation. It allows a service provider to advertise a service offer and a client to discover a service by querying its static and dynamic (behavior) properties in distributed systems. Flexibility is introduced since the ontology and its content can be changed and adapted to usage. When a client discovers a service offer by querying its static properties, it can obtain information about the service behavior enabling it to use the service discovered and to understand how the invocation of the service operations needs to be processed.

Based on SDL and Interface Automata, we can describe the service behavior which enables the client to use the service discovered and to understand how the invocation of the service operations needs to be performed. Based on the service behavior description and facilitators, we have presented in this paper an approach that allows the composition of services in distributed systems. This approach is a powerful one since it allows the client to benefit from the functionalities of more than one service to get novel functionalities and novel services. It is an innovative one since it allows the service composition in an implicit manner and the system is automatic and clients can be general public and they don't need to be professional in the domain of web service composition.

As perspectives, we will propose an approach of facilitator federation to compose services. In this context, when a client queries a service from a facilitator and this service doesn't exist, the facilitator can interact with other facilitators to search and compose services to satisfy the client's request. So, this approach will allow the composition of services located on different facilitators and can be used in the peer-to-peer networks.

References

[1] *www.uddi.org*.

[2] *http://www.daml.org/services/*.

[3] L. D. Alfaro and T. A. Henzinger. Interface Automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*. ACM Press, 2001.

[4] D. B. et al. Automatic service composition based on behavioral description. In *IJCIS*, 2005.

[5] M. P. et al. Planning and monitoring web service composition. In *AIMS*, 2004.

[6] D. Fensel, S. Decker, M. Erdmann, and R. Stude. Ontobroker : The very high idea. *In Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, USA*, May 1998.

[7] ITU-T. ITU-T Recommendation Z.100. Specification and Description Language (SDL). 1989.

[8] T. Khedro and M. R. Genesereth. Facilitators: a networked computing infrastructure for distributed software interoperation. In *Worhshop on AI in distributed information networks (IJCAI)*, Montreal, Canada, 1995.

[9] M. Kifer, G. Lausan, and J. Wu. Logical foundations of object-oriented and frame-based language. *Journal of the ACM, 42(4) : 741 - 843*, 1995.

[10] OMG. *CORBAservices : Common Object Services Specification. OMG Document*, 1997.

[11] J. Yang, M. P. Papazoglou, and W. den Heuvel. Tackling the Challenges of Service Composition in E-Marketplaces. In *The 12th International Workshop on Research Issues in Data Engineering : Engineering e-Commerce/e-Business Systems*. RIDE 2002.

[12] O. K. Zein. Indexing/Discovering and composition of distributed services. *PhD thesis*, January 2005.

[13] O. K. Zein and Y. Kermarrec. An approach for describing, discovering services and for adapting them to the needs of users in distributed systems. In K. Sycarra and T. Payne, editors, *In the proceedings of AAAI Spring Symposium on Semantic Web Services*, Stanford, California, Mar. 2004.

[14] O. K. Zein and Y. Kermarrec. An Approach for Describing and Querying Service Behavior in Distributed Systems. In *The IEEE international Symposium on Applications and the Internet*, pages 136–139, Trento, Italy, Jan. 2005.

[15] O. K. Zein and Y. Kermarrec. Static, semi-dynamic and dynamic composition of services in distributed systems. In *IEEE International Conference on Internet and Web Applications and Services (ICIW'06)*, Guadeloupe (French Caribbean), 17-25 February 2006.