DISCOVERY AND AUTO-COMPOSITION OF SEMANTIC WEB SERVICES

Philippe Larvet
Alcatel CIT, Research and Innovation, Route de Nozay, 91461 Marcoussis, France

Bruno Bonnin
Alcatel CIT, Research and Innovation, Route de Nozay, 91461 Marcoussis, France

Keywords: Web service, semantic web service, orchestration, automatic web service composition, semantic service discovery.

Abstract: In order to facilitate the on-demand delivery of new services for mobile terminals as well as for fixed phones, we propose a user-centric solution based on Semantic Service-Oriented Architecture (SSOA) for instant building and delivery of new services composed with existing Web services discovered and assembled on-the-fly. This solution, based on semantic descriptions of Web services, is made of three main mechanisms: a semantic service discoverer, transparent for the user, allows to find the pertinent Web services matching with the user's original request, expressed vocally or by a SMS or a simple text ; a semantic service composer, using the semantic descriptions of the Web services, allows to combine and orchestrate the discovered services in order to build a new service fully matching the user's request, and a service deliverer makes the new service immediately accessible by the user.

1 SEMANTIC WEB SERVICES

Web services, as they are easily accessible from any point of Internet through an Application server, are suitable to build rapidly on-demand applications.

From the point of view of their internal complexity, Web services (WS) can be divided into two families : *elementary* WS and *composite* WS.

Elementary WS provide a basic service, comparable to mathematical libraries, and contain a low level of data transformation, embedded in few algorithms; for example, translation services are elementary WS.

In the contrary, composite WS provide a high level service and contain many levels of data accesses and transformations, given by the cooperation of several elementary services. Such a composite service, resulting of the composition of several *processes* logically assembled, can be called an *orchestrated* service; for example, reservation services or secured-payment services are composite orchestrated WS.

If the development of elementary WS is obviously the domain of specialized software engineers, the emerging technology of *semantics*, associated to web services, allows a end user, through an appropriate application accessible from his preferred phone (mobile or fixed), to build *transparently* and to use some new services made by an assembly of existing elementary WS, discovered and orchestrated on-the-fly. "Transparently", because the process of discovery, composition and delivery of the new service has to be totally transparent for the user whose the only demand is to get a new easy-to-use service immediately responding to his original easy-expressed request.

A second advantage of the semantic technology attached to WS is to allow keeping only the semantic descriptions of the requested services, instead of the orchestrated resulting service, because the semantic descriptions can be processed and re-processed at any time, permitting a *dynamic* re-discover, re-orchestrate and re-deliver of a "new" new service, matching better with the original request.

2 EASY-EXPRESSING NEW SERVICES

Let us develop a concrete use case to explain the functioning of our solution. For example, a user wants to get a french translation of the latest CNN news.

Larvet P. and Bonnin B. (2006).
DISCOVERY AND AUTO-COMPOSITION OF SEMANTIC WEB SERVICES.
In *Proceedings of the First International Conference on Software and Data Technologies*, pages 40-45
DOI: 10.5220/0001310500400045

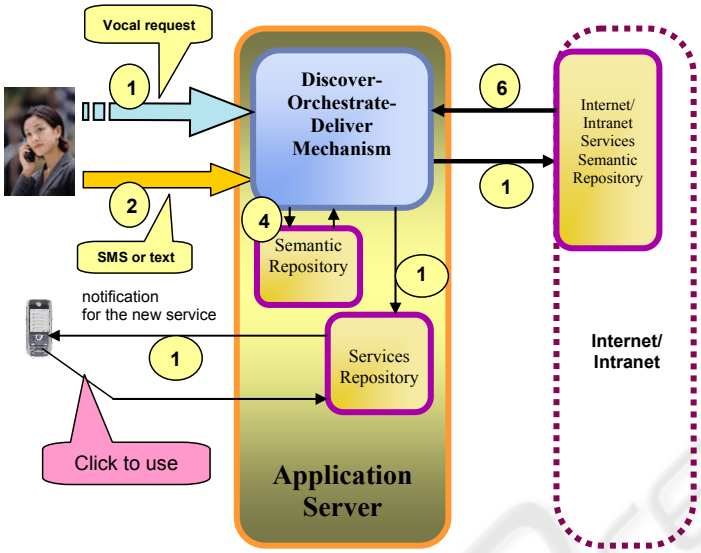

Figure 1: Processing an Instant Building and Delivery use case (global view).

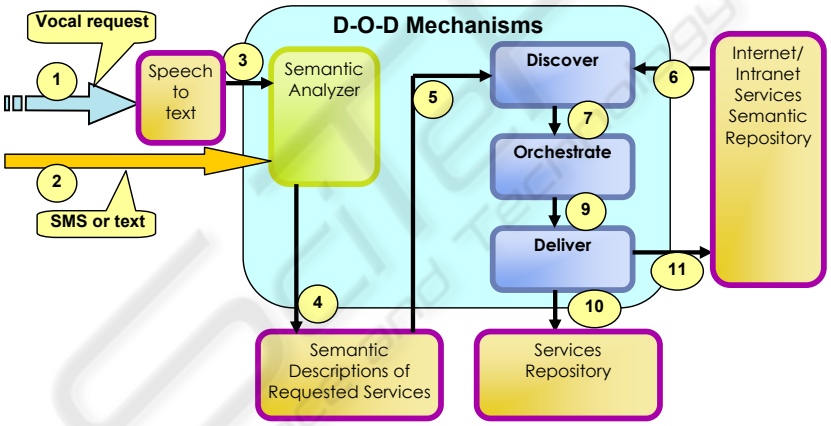

Figure 2: Processing an Instant Service Discover-Orchestrate-Deliver use case (detailed view).

The user simply activates the "instant delivery" function of his terminal, and expresses vocally his demand : "I want to get a french version of the latest CNN news". He could also send an SMS containing this short text to the address of the Instant Delivery Service. Indeed, there is no easier way to express a requirement !

The Fig. 1 allows you to follow the building and delivery of the final required service from the original request.

The vocal message (flow 1) is taken into account by the Discover-Orchestrate-Deliver Mechanism of a given Application Server.

A fisrt block, Speech-to-Text (see Fig. 2), "understands" the message and transforms it into a text, that is processed by a Semantic Analyzer. If the user has sent a text (flow 2) instead of speaking a vocal message, this text directly enters the Analyzer that extracts from the text content the needs for the intermediate elementary required services. In the case of our example, the needs are:

- french translation
- latest CNN news

These needs are kept in the Application Server under the form of "semantic descriptions" of the intermediate required services and they are stored in a specific Semantic Repository (flow 4).

The stored semantic descriptions are processed by the Discover module (see fig. 2) that searches in a public Semantic Services Repository (flow 6) some service descriptions that match with the required stored semantic descriptions.

The corresponding elementary web services (for example a web service Translator and a web service CNN News), fitting with the required semantic descriptions, are passed to the Orchestrate module (flow 7), which makes the composition of the elementary web services, using the orchestration handlers contained in their semantic descriptions.

The resulting composed service (flow 9) is taken into account by the Deliver module, that stores the new service in the Service Repository of the Application Server (flow 10) and registers in the public Intranet/Internet Semantic Services Repository the semantic description of the new service (flow 11).

The Service Repository has a local intelligence allowing to send to the user (and, occasionally, to every user who previously subscribed to this service) a notification (flow 12) informing him that the new asked service is now available on his preferred server.

Then, and this is the end of our use case story, the user can immediately activate and use the new service – and to get a french version of the latest CNN news.

Beyond the obvious benefit of this process for the user – getting a real-time and on-the-fly response to his original demand, under the form of a new service fully matching his request and immediately activable and usable – another interest of this solution is set in the notification mechanism of the Service Repository: every user who previously subscribed to the "Instant Delivery" service is informed that a new service is available.

The following paragraphs explain in detail the functioning of the three main blocks of the Discover-Orchestrate-Deliver Mechanism, whose the process is protected by an Alcatel patent.

3 THE DISCOVERY MECHANISM

From The User's Original Request To The Pertinent Elementary Web Services

Before entering the Discover module, a pre-treatment is made onto the text of the original request, in order to transform it into semantic clauses, each of them representing a precise need for an elementary web service.

This need has to be expressed in a formal way, for example under the form of a XML-based language, to allow its easier post-processing.

In the context of our solution, we imagined a simple semantic description language based onto WSDL (Web Service Description Language, a W3C's standard) and we developed such a semantic description mechanism, taking into account the fact that a WSDL description is always present for any web service. The process of making semantic descriptions from WSDL is protected by another Alcatel patent.

A "semantic description" for a given service contains:

- input data description (data category, types and semantic tags)
- output data description (same structure than input)
- semantics of operations
- "orchestration handlers", i.e. connection points expressing how this very web service could be automatically connected with another WS. This automatic connection process is also protected by an Alcatel patent.

The semantic descriptions for the required elementary services, built by a Semantic Analysis module from the text of the original request, are stored in a specific Semantic Repository, local to the Application Server (see Fig.2, flow 4).

These descriptions are used by the Discover block (flow 5) to make a search into the Internet/Intranet Semantic Service Repository (flow 6). The Discover block contains a matching module that explores the external Semantic Repository, looking for a match between the external semantic descriptions (flow 6) and the required semantic descriptions of the flow 5 that come from the user's request. This match is based upon a correlation search between the main elements of the semantic description: inputs, outputs and semantics of operations.

When a match is found for each required semantic description, the corresponding WSDL descriptions of the elementary services are sent to the Orchestration module.

4 THE COMPOSITION MECHANISM

From The Elementary Components To The Final New Service

Orchestration is the term used to describe the creation of a "business process" (or a workflow) using Web services (BPEL Editorial Team, 2005). A business process is an aggregation of services whose the

operations, i.e. the processes, are logically linked together in order to reach a given objective (Bruno, 2005).

Aggregating services to build an added-value service have many solutions depending on the chosen environment. For Web services, the orchestration is usually expressed with a specific language like BPEL for example (Business Process Execution Language (BPEL Editorial Team, 2005)), that describes the interactions between the services.

A business process is deployed itself as a service, so it can be used by other processes. A business process language describes the behaviour of business processes based on Web services, i.e.:

- Control flow (sequences, loops, conditions, parallelism, …)
- Variables, exceptions, timeout management.

By providing semantic descriptions of services, we produce in fact specifications for all the layers of the standard Web services stack, and so in the service composition layer.

The stack has been enriched with new W3C specifications, such OWL-S (OWL-S, 2004) for example, for providing support of semantics. These languages, and our "semantic descriptions"have the same goal, provide a set of XML constructs for describing the properties and capabilities of Web services in order to facilitates the automation of Web services processing, including automatic discovery and composition.

In the previous step ("Discover" mechanism), we have extracted the "semantics" from the service descriptions to find the matching services. In the current step, we automatically build the business process with some specific *semantic annotations* (that we call "semantic tags") included in the service description in order to help the composition.

For example, the *semantic annotations* give information about the input and output parameters: in a business process, output parameters of a service can be used as input parameters for another service.

The "semantic tags" are important to use the parameters in the right way. The parameter types are not sufficient to match output parameters of a service with input parameters of another service. The "semantic tags" help this composition.

In a business process, there is not only service invocation activity: there are also condition activities, loop activities, error processing, etc.

These activities are also taken into account: for example, if an activity has an array of string as output parameter and if the next service has only a string as input parameter, the business process includes a loop activity for processing all values of the array.

Once the business process has been defined and described with a business process language, it is time to deploy it.

5 THE DELIVERY MECHANISM

Serving The Newborn Service To The User

With the Service-Oriented Architecture (SOA)(Nickolaos et al., 2004), (SOA explained, http), we have a good level of dynamicity on server side. But, on client side, as the client of the services is an application that is installed on each machine, it must also be as dynamic as possible to be able to invoke any services.

There are two problems:

- How to access the new service?
- How the user knows there is a new service?

To access the new service, the user must have a service client application on his terminal. But as all the services are different, the service clients are different (service parameters, user interface, etc.).

So, the service deployment step must include:

- A service client building,
- A user notification.

The service client application (see Fig.3) must include the following functionalities:

- Business Process Server notification management.
- Service client management. Its role:
  - Download, start, stop, delete the service clients
  - Process events from the *notification manager*
  - Manage the user interface according to the authorized service clients.
- Service clients execution: their main role is to access the services. They have their own user interface.

A new business process deployment includes the following steps:

- Building of a new service client. It is built from the new service description. For each service argument, a user interface element is created.
- Deployment of the service client.
- Notification of the user that a new service is available and that he can access it.

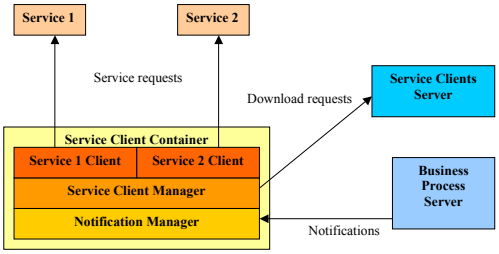

Figure 3: Detailed functional view of a Service Client Application

When the user receives the notification, he activates the service:

- The *service client manager* downloads the service client
- The service client is activated
- The service client invokes the new service

The main advantages of this kind of service client container are:

- Install once: only the Service Client Container has to be installed on each machine, the other parts of the application (service clients) are downloaded
- Context-aware: the application checks the hardware and software environment for proposing only usable applications and for using user's preferred tools such as web browser
- Dynamic upgrade: when running, if the Service Client Container receives a notification from the Business Process Server for any modification, it is immediately taken into account; this mechanism is transparent for the user and does not need any restart.

6 MANAGING THE DISCOVER-ORCHESTRATE-DELIVER CONCEPT WITH A SEMANTIC SERVICE-ORIENTED ARCHITECTURE

Service-Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among interacting software agents (SOA explained, http). In this context, a service is seen as a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners.

To achieve loose coupling among interacting software agents, SOA employs two architectural constraints:

1. A small set of simple and ubiquitous interfaces to all participating software agents. Only generic semantics are encoded at these interfaces, that are universally available for all providers and consumers.
2. Descriptive messages constrained by an extensible schema delivered through the interfaces. No, or only minimal, system behavior is prescribed by messages. A schema limits the vocabulary and structure of messages – and an extensible schema allows new versions of services to be introduced without breaking existing services.

Since we have only a few generic interfaces available, we must express application-specific semantics in messages. We can send any kind of message over our interfaces, but there are three main rules to follow before we can say that an architecture is service-oriented:

1. The messages must be descriptive, rather than instructive, because the service provider is responsible for solving the problem – when you go to a restaurant, you tell the waiter your order and preferences but you don't tell the cook how to cook your dish step by step.
2. Service providers will be unable to understand your request if your messages are not written in a format, structure, and vocabulary that is understood by all parties. Limiting the vocabulary and structure of messages is a necessity for any efficient communication.
3. Extensibility is vitally important: the world is an ever-changing place and so is any environment in which a software system lives. Those changes demand corresponding changes in the software system, service consumers, providers, and the messages they exchange. If messages are not extensible, consumers and providers will be locked into one particular version of a service.

There are numerous additional constraints one can apply on SOA in order to improve its scalability, performance and reliability. Among them, we mainly retain the stateless service concept. Each message that a consumer sends to a provider must contain all necessary information for the provider to process it.

This constraint makes a service provider more scalable because the provider does not have to store state information between requests. This is effectively "service in mass production" since each request can be treated as generic. It is also claimed that this constraint improves visibility because any monitoring software can inspect one single request and figure out its intention. There are no intermediate states to worry about, so recovery from partial failure is also relatively easy. This makes a service more reliable.

Taking into account these main particularities of SOA, we can consider the main challenges for our Discovery-Orchestrate-Deliver concept in a Semantic Service-Oriented Architecture (SSOA) are the descriptions of messages and services, the management of services and the process for facilitating the access to services.

At "Discover" level, the semantic descriptions of services – as they are made from the external descriptions of messages found in WSDL – make more easy the classification of services in the service repository, because these descriptions are structured upon a taxonomy of the concepts manipulated by the services. This taxonomy facilitates the logical ordering of services by domains, subdomains, topics and subtopics, and consequently facilitates their management. These semantic descriptions, as they can match with the service requirements contained in the user's original request, are also a good means to make more easy the discovery of corresponding services.

At "Orchestrate" level, the "composition handlers" contained in the semantic descriptions of services are a good enabler for an automatic orchestration process able to aggregate several convenient services.

The semantic descriptions of a new service, and mainly their taxonomic abilities, are still used at "Deliver" level, when the Discover-Orchestrate-Deliver mechanism has to store them as logically as possible and to insert them in the existing structure of the semantic repository.

7 CONCLUSION

Within a wide-communicating world where services become more and more sophisticated and are on the beginning to expose their semantic dimension, we consider it is important to propose solutions in which anybody could be able to create his own services in some simple clicks. This paper reveals one of the reflexion axis we currently have in this direction, and we hope we could convince the reader that a semantic approach, through the use of semantic descriptions of services, helps in an important way the mechanisms for building, holding and processing any kind of Web service.

Moreover, we tried to show how Semantic SOA-based mechanisms, if possible leant on IMS development platforms, can provide high-value capabilities for new generation service creation and delivery: a simpler way to express a service, a faster way to get and use it, a cheaper solution for the end-user because it is simpler and faster.

We are convinced that IMS services are the good base to develop such kind of rich services and use interaction, including dynamic search, discovery and publishing of web information associated to a multi-media call session, and we think semantic Web services are the good means to do it.

REFERENCES

BPEL Editorial Team, *BPEL Learning Guide,* February 2005, http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci880731,00.html

OWL-S, *Semantic Markup for Web services,* W3C Member Submission, November 2004, http://www.w3.org/Submission/OWL-S/

Bonnin Bruno, Larvet Philippe, Fontaine Patrick, Ferres Lamia, *A Multi-Actor Agnostic Platform for Web Services Development and Deployment,* June 2005, Article published in W3C Workshop on Frameworks for Semantics in Web Services, Innsbruck (Austria)

Kavantzas Nickolaos, *WS-CDL, Web Service Choreography Description Language,* December 2004, http://www.ebpml.org/ws_-_cdl.htm and http://www.w3.org/TR/ws-cdl-10/

Kavantzas Nickolaos & al., *Process-centric realization of SOA : BPEL moves into the limelight,* Web Services Journal, Nov.2004, http://www.findarticles.com/p/articles/mi_m0MLV/is_11_4/ai_n7071401

Nanda Mangala & al., *Decentralized Orchestration of Composite Web Services,* IBM Research Computer Science, Innovation Matters, November 2004, http://www.research.ibm.com/compsci/project_spotlight/distributed/

Peltz Chris, *Web services orchestration, a review of emerging technologies, tools, and standards,* Hewlett-Packard Co, January 2003, http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf

Smith Howard, *BPM and MDA, Competitors, Alternatives or Complementary,* Business Process Trends, White Paper, July 2003, http://www.bptrends.com/publicationfiles/07%2D03%20WP%20BPM%20and%20MDA%20Reply%20%2D%20Smith%2Epdf

SOA explained: http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html