論文
4-10-11

시멘틱 웹 서비스 합성의 요구사항 분석
Analyzing of Requirements for Semantic Web Service Composition

최병석(Byeong-Seok Choi)*홍현술(Hyeun-Sool Hong)**한성국(Sung-Kook Han)***

요 약

웹 서비스는 현재의 웹 시스템 구조 위에 한 단계 더 나은 수준의 새로운 서비스를 보장한다. 그러나 웹 서비스의 잠재력을 활용하기 위해서는 웹 서비스들에 대한 적절한 서술방법이 개발되어야 한다. 웹 서비스 합성을 서술하기 위한 최근의 연구들을 보면, 서비스의 자동적이고 선언적인 서비스의 조합을 가능하게 하는데 요구되는 중요한 요소들이 부족하다. 본 논문에서는 웹 서비스 합성분야의 현재의 기술 상태를 명확히 하고, 이것을 완성 시키는데 다음 단계를 정의해주는 지침을 제시하기 위하여 효과적인 시멘틱 웹 서비스 합성을 위한 요구사항을 제시한다. 또한 현재의 연구들이 시멘틱 웹 서비스 합성의 잠재적인 가능성을 어느 정도까지 확장시켜 개발하였는지를 서술하면서, 제시된 요구사항들을 토대로 하여, 이러한 분야의 가장 중요한 연구의 출발점이 되는 EPEL4WS, DAML-S 그리고 WSMF에 대하여 분석할 것이다.

Abstract

Web services promise a new level of service on top of current web. However, in order to employ their full potential, appropriate descriptions means for web services need to be developed. Recent efforts to describe web service composition lack important features needed to enable a real automated and declarative combination of services. In this paper a set of requirements for effective Semantic Web Service composition are presented, aiming to clarify the state of the art in the area and at providing guidelines to define next steps to be accomplished. We will provide an analysis of BEPEL4WS, DAML-S and WSMF, the most important initiatives in this direction, based on these requirements, describing to what extent current efforts have developed the potential of Semantic Web Service composition.

* 정회원 : 군장대학 컴퓨터응용학부 교수
** 정회원 : 원광보건대학 컴퓨터응용개발과 교수
***정회원:원광대학교 컴퓨터 및 정보통신공학과 교수

논문접수 : 2003. 9. 25.
심사완료 : 2003. 10. 7.

1. Introduction

"Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. ... Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service". [IBM web service tutorial].

The current web, consisting mainly of a collection of information, is being lift to a new level of service. The concept of web service and the technologies related are gaining importance, providing software programs which can be executed via the web. Such services can provide information, e.g. a TV program information service, or it can have an effect on the real world, e.g. a book buying service. Web services technologies are significantly increasing the potential of the current web, thus they are an important focus of interest from various and big IT companies.

Web services are here to stay [Sheth et al. 2003], as both business and technical considerations will provide them with the staying power and tailwind to survive the hype.

Current web service technology, centered on SOAP, WSDL and UDDI, has a very basic and non-automated interaction model. A Web service provider publishes information about the Web service to an UDDI registry and offers a programmatic access the consumer can bind to. The Web service registry contains information about location and offered Web services from all Web service providers. Then, Web service consumer interacts via XML based messages on a transport protocol with a Web service registry or a Web service provider. The dynamic binding to the consumed Web services is done using the information from Web service registry. In all this process, no real automation is achieved, and the human actor is involved in the loop.

To bring web services to its full potential and also to help to bridge the gap among current web and the next generation one (Semantic Web), semantics must be used in order to develop web services' full capabilities. The Semantic Web should enable users to locate, select, employ, compose, and monitor Web-based services automatically. Semantic Web Services combine Semantic Web and Web Service Technology.

To make use of a Semantic Web Service (SWS for short), a software agent needs a computer-interpretable description of the service, and the means by which it is accessed. An important goal for Semantic Web markup languages, then, is to develop the infrastructure over which these descriptions are made and shared. Web sites should be able to employ a set of basic classes and properties for declaring and describing services. Ontologies provide a mean to do so. They are a key enabling technology for the semantic web [Fensel, 2001]. They interweave human understanding of symbols with their machine-processability, allowing declaration and description of services.

Driving the development of markup and agent technology towards automation tasks

of Web services will enable in particular, service discovery, execution, composition and interoperation [McIlraith et al. 2001]:

- **Automatic Web service discovery** involves automatically locating Web services that provide a particular service and that adhere to requested properties. With semantic markup of services, it can be specified the information necessary for Web service discovery as computer-interpretable semantic markup at the service Web sites, and a service registry or (ontology-enhanced) search engine can automatically locate appropriate services.
- **Automatic Web service execution** involves a computer program or agent automatically executing an identified Web service. Semantic markup of Web services provides a declarative, computer-interpretable API for executing services. The markup tells the agent what input is necessary, what information will be returned, and how to execute-and potentially interact with-the service automatically.
- **Automatic Web service composition and interoperation** involves the automatic selection, composition, and interoperation of appropriate Web services to perform a task, given a high-level description of the task's objective. The information necessary to select, compose, and respond to services is encoded at the service Web sites. Software to manipulate this markup can be written, together with a specification of the task's objectives, to achieve the task automatically. Service composition and interoperation leverage automatic discovery and execution.

Of these three tasks, none is entirely realizable with today's Web, primarily because of a lack of content markup and a suitable markup language. At the moment there exist several efforts, academic as well as industrial, trying to bridge the gap among the actual web, and the semantic one. Such efforts are centered in Web service discovery [Sycara et al.] and some aspects of services execution through initiatives such as the Universal Description, Discovery, and Integration (UDDI[1]) standard service registry; the XML-based Web Service Description Language (WSDL[2]), and ebXML[3].

However, in order to achieve all the objectives depicted above, a key feature required is automatic composition, as it enables real distributed web services, transforming the Web from a distributed source of information to a distributed source of service.

This paper focuses on web service composition; one of the three tasks needed to provide real automation of web services, and defines a set of requirements for a real and effective SWS composition, analyzing the two major efforts in this direction, namely: DAML-S and WSMF.

The rest of this paper is structured as

1) www.uddi.org/

2) www.w3.org/TR/wsdl

3) www.ebxml.org/

follows: section 2 sketches the requirements for SWS composition; section 3 introduces the state of the art in the area, centered on DAML-S[4] and WSMF[5] and presenting a description and evaluation of both initiatives. Finally section 4 presents the conclusion of this work.

2. Requirements for SWS composition

Semantic web services [Fensel et al. 2002b] differ from traditional web services in that they employ semantic web technology to provide autonomous, intelligent access over a heterogeneous, distributed network environment.

However, some approaches of semantic web services extend the traditional web service by adding other stacks on top of UDDI, WDSL and SOAP. Although these approaches propose several methods such as web services choreography or orchestration based on business workflow management [Peltz 2003, Arkin et al. 2002], the current web service composition approaches consider simplistic method invocation [Zhengang et al. 2002].

In general, the composition of web services is the task of combining and linking existing web services and other components to create new business processes, or to solve a high-level task by combining atomic functionalities provided by different web services. It adds value to the collection of services by integrating them according to the requirements of business process. We have identified the following issues as necessary to achieve the efficient service composition enhanced semantically:

- **Description of integrated web service process**

The various aspects of web service must be populated to deliver the correct conceptualization so that they are semantically correct and are interpreted correctly by the service requesters and providers. This requires ontological supports to describe, select, bind, compose and mediate valid element structures for complex web services. The adaptation of ontologies can provide interoperability and reusability between service components with the same formal and consensual specifications.

- **Discovery and interoperability of services**

It is necessary to manually or automatically search for appropriate web services through the semantic supports instead of simple use of keywords. Once the desired web services have been discovered, an intelligent agent is needed to facilitate the resolution of syntactic and semantic differences between collaborated services. The agent should understand how to establish service connection between heterogeneous service interfaces for the realization of the desired complex business processes.

- **Process execution and monitoring**

The composite web services are executed according to business processes. Whether the web service is synchronous or asynchronous, a suitable scheduling technique for management of invoked service is necessary

4) www.daml.org/services/

5) swws.semanticweb.org/

in order to improve system performance. The management of process execution includes the functionalities for exception and fault handling as well as compensation activities.

The security and authentication issues are also considered for secure, reliable composition. The web services internal aspects should be hidden as far as possible from the service requester. This isolation may provoke serious problems in case of distributed, long-running business transaction and make service request feel a little left out.Consequently, in web services environment, functional components to monitor business relevant information and workflow status should be provided for more comfortable business service environment.

We will discuss the manifold requirements of semantic web services composition in terms of the following three aspects: static requirements usually required for service descriptions, dynamic requirements for operational and functional features, and web service framework requirements for the management of composite services.

2.1 Static Requirements of service composition

Semantic web service components require the obvious identity difference from those of software components. The description of the identity should specify what to have as semantic web service components and provide mutual semantic consensus to understand their functionalities.

2.1.1 Strong de-coupling of web service components

Many web service approaches do not make an explicit distinction between an internal description of a web service for service provider and its external visible interface for service requester. In other words, these approaches complicate service environment by means of shuffling private and public services. We have already experienced that the notation of a software object, which combines functions and data in encapsulated units, gave rise to new paradigm in software engineering. Furthermore, web services, are loosely-coupled, distributed service components over the global network and involve many different agents with different goals. Therefore, the clear separation between private and public descriptions is essential to provide the necessary isolation and abstraction between service providers and requesters, which mean that web services should be black boxes as independent functional service objects [Fensel et al. 2002b]. The black box components encapsulated detail business processes can provide necessary functionalities to achieve reusability, composibility, and interoperability.

Sometimes, the black box concept is required from business side. Providing web services means that service provider exposes its internal business environment to the outside world, customers, suppliers, and business partners. On account of publicity of business process, service providers consider protection and security for their own business services.

The separation of concerns provided by black box concept can achieve the realization of description, discovery, integration and mediation of semantic web services.

2.1.2 Ontology-based service description

Ontology is a specification of a representational vocabulary for a shared domain that can be communicated across web service systems. Thus, ontologies are a key enabling technology for semantic web services. Ontologies are used to define the common vocabulary that describes the specification of web service elements. For the effective composition of complex services as well as interoperability between services, ontology-based descriptions of service components are essential. On the other hand, when web services are described with ontologies, web services can be categorized according to their ontological hierarchy. This enables more effective discovery and categorization services analogous to Yahoo! in web pages [Burbeck 200], a key aspect needed for efficient SWS composition.

2.1.3 Separation of goal description and web service description

The capabilities description specifies objectives that a service requester may have on searching web service repository. The purpose of goal descriptions is for service advertisement and public process description. On the other hand, web service description specifies the internal business workflow as private process. As discussed in black box concept for the separation of concerns, the clear distinction between goal description and web service description are indispensable, because there is an n:m mapping between them and their roles and usages are inherently quite different.

The goal description contains the following declarative elements to support service discovery and composition:

1) Pre-conditions describe what a web service needs to provide its service.
2) Post-conditions describe the effects of the execution of the service.

Pre-conditions and post-conditions should be declarative and high-level descriptions which enable high-level reasoning for service composition. They are different from input and output information, as they specify high-level requirements instead of real data transformation. In a hotel booking service context, a valid pre-condition could be that a valid credit card is provided, and a post-condition could be that the requester's account is charged (effect of the service execution). On the other hand, a valid input would be the credit card number and expiration date, and a valid output a notification of the charge. Notice the different level of abstraction of pre-conditions/postconditions and input/output. The goal specification also should include several optional choices that can be chosen according to the priorities of service requester, e.g., business class in case of economy class unavailable in flight booking service, and the constraints that reveal functional attributes, e.g., only applicable to a specific region or requester. These interface optional parameters will provide parameterization and configuration of web services, which can also be specified with distinct ontological terminologies. Other important elements in the goal description are security, authentication, and privacy issues related to the exchange of information necessary for the service to be linked and legal or contractual elements between the provider and requester of the service [Burbeck 200]. These issues should be

considered in order to compose secure, authorized service components.

The web service description to specify business process is related to web service description language which defines various aspects of business workflow including fault and exception handling, input and output data, and so on. Web service description languages should require that data flow and control flow is separated from description for service composition, as we will discuss later.

2.1.4 Proof digital signature

Semantic web service components are open, distributed service components. Thus, the assurance should be provided that these components are secure and reliable. In addition, the guarantee of the quality of service (QoS) should be stipulated to service requesters. The trust of service components becomes important factor to select elements to be composed among discovered services.

2.2 Dynamic Requirements of service composition

Since web services are collaboration tasks, they should satisfy operational requirements to exchange business messages during execution. Most of dynamic requirements are related to the web service framework or architecture. We will outline requirements dealt within service components.

2.2.1 Workflow monitoring

The drawback of black box concept in web service description is that service request cannot seize the state of business processing. In case of asynchronous web services and long-running business tasks, in particular, monitoring workflow process is required for the process management as well as dynamic service composition. Service requests may want to know the current status of their services for further business decisions. Additionally, after monitoring business process state, web service can replace a certain service component in accordance with scheduling policy or business requirements. Workflow monitoring is also used to realize the interaction or conversation for the collaboration between service provider and requester. Workflow monitoring can be accomplished in terms of the explicit specification in the interface optional parameters, while keeping the clear separation between private and public processes and the black-box assumption valid.

2.2.2 Separation of process control and service ontologies

For the effective service composition, complex business workflow should be composed in the manner of servicedriven, not data-driven. As business workflow of service provider, in general, is organized as a unit of service, business process model is implemented when the execution of composite services can be controlled according to service status. On the other hand, service data are usually neutral, based on independent ontologies that support reusability. Considering the fact that the composite service is an implementation of business process model and service data are process-independent, their separation is natural.

2.2.3 Exceptions and faults handling logic

As the exceptions and faults are an unavoidable issue in web services, the exceptions and faults handling logic should be specified for the completeness of composite services. The exceptions and faults can be resolved by means of interactive conversation or business compensation strategies. To handle exceptions and faults, ontologies should provide sufficient information for compensation and the process manager should keep track of the service context.

2.3 Web Service Framework requirements of service composition

To execute and manage composite semantic web services, web service framework should provide proper architectural facilities to control the diverse service activities. As web services are inherently based on composite service components [Sollazzo et al. 2002], these requirements for service composition are the core functionalities of web service framework. We will discuss some of the functionalities closely related to service composition in brief.

2.3.1 Strong mediation facilities

For an open, loosely-coupled and distributed environment of web services, mediators are essential means to cope with the inherit heterogeneity [Fensel et al. 2002b]. The widely used mechanism in the heterogeneous, distributed processing is the broker mechanism. However, the key role of the broker may disappear [Bussler et al. 1999]. Though it may still be viable as a kind of search machine for web services, it will lose its central role, because everyone may publish semantic descriptions and crawlers may find them. For web service environment, syntax and semantics of data schema, business process logic and message exchange protocols are incoherent in spite of using ontologies. As business process and service ontologies are independent entities of web services, the mediator should distinguish them. The strong mediation is also required to guarantee service level agreement (SLA) of composite services.

2.3.2 Persistence of service context

The business context for the correct realization of the business process must be stored persistently to avoid loss of service context. For asynchronous web services and long-running business tasks, persistence of service context for workflow process is required for the process management as well as dynamic service composition. The business context for invoked services can be utilized to exchange messages between composite services. Furthermore, the persistent service context provides the fundamental management information to system manager. Upon failure of an invoked service the business process can define a compensation strategy for the failure. For the compensation, persistent storage can be used to implement the various strategies. Persistence of service context is necessary to achieve effective system management as well as collaboration between services.

2.3.3 Exceptions and faults handling

Even though business logic to handle exceptions and faults is specified in some service description languages, the autonomous handling of exceptions and

faults should be innate within web service framework. Several approaches may be considered to recover business work flow. These approaches are related to either compensation or new service generation. Dynamic service discovery and composition become fundamental functionalities to handle exceptions and faults in web service framework. That is, the issue for exceptions and faults handling is one of the principle basis of a web service framework which enables real composition of services.

2.3.4 **Automatic discovery of services**

The fast and dynamic composition of business process is an essential requirement for service provider to adapt their business strategies to the dynamic nature of web services. Business partner may be either permanent for long-term business or temporary for short-term business. In the permanent business, services are known in advance and completely defined. In the temporary business, however, services are dynamically discovered to execute the required transactions. This type of dynamic composition requires supports for automatic service discovery and fast composition [Benatallah et al. 2002]. In addition, automatic discovery of services is frequently required in case of exceptions and faults handling. The automatic service discovery and composition is the principal functionality of web service framework to realize the ultimate goal of web services composition.

2.3.5 **Management of service proxies**

Communication over the Internet is currently inherently unreliable and insecure. It is, therefore, required that service requester and service provider make sure themselves through defined protocols that messages are transmitted with the manner of at-least once semantics, atmost once semantics, early once semantics, and in-order semantics[Layman 2003]. The mediation handles defining a sequence of acknowledgement procedure in addition to related constraints such as time-outs, retry logic and upper retry limits, and low-level protocol conversion. The management of service proxies contains the proper binding of service interfaces.

2.3.6 **Maintenance of ontology repositories**

Ontologies [Fensel et al. 2002b] are key enabling technology to describe semantic web services, discover the desired services, manage workflow, and compensate broken business process. Ontologies also provide the common conceptual vocabulary that used in web service framework. The composition of web services will be accomplished under the basis of ontologies. The web service framework is required to maintain various ontology repositories as its knowledgebase.

3. Technologies related to SWS composition

In the following the technologies related with web services composition will be introduced.

3. 1 BPEL4WS

BPEL4WS (Business Process Execution Language for Web Services) provides a language for the formal specification of business processes and business interaction protocols. It is mainly an IBM, Microsoft

and BEA joint effort, trying to use web services to implement business processes. Business processes can be described in two ways [Curbera et al. 2002], namely:

- **Executable business processes** model actual behavior of a participant in a business interaction.
- **Business protocols**, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called *abstract processes*.

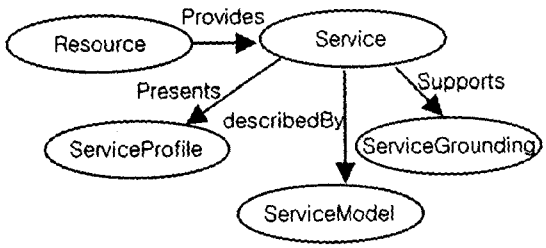
The BPEL4WS web services composition is much related to the definition of a business process. Nevertheless, as BPEL4WS doesn't provide any semantics for service description, it violates most of the requirements for an effective SWS composition and its analysis is without the scope of this paper.

3.2 DAML-S

DAML-S is an ontology of services, being developed as part of the DARPA Agent Markup Language program and currently at version 0.9. It aims to provide suitable ontologies to enable users to locate, select, employ, compose and monitor Web-based services automatically, and to enable agents to process service description and to access such services automatically [DAML Service Coalition, 2003].

DAML-s is designed to enable automatic Web service selection, composition and interoperation to perform some task, given a high-level description of an objective. It provides declarative specifications of the prerequisites and effects of an individual service that are necessary for automatic service composition and interoperation.

The ontology structure used within DAML-S is made of the following elements (Fig. 1):



[Fig.1] DAML-S upper ontology

- **Service Profile**: What does the service require of the user and provide for him?
- **Service Model**: How does the service work?
- **Service Grounding**: How is the service used?

One of the key elements for (dynamic) composition is the service profile, as it gives the type of information needed by a service-seeking agent to determine whether the service meets its needs. In addition to representing the capabilities of a service, the profile can be used to express the needs of the service-seeking agent, so that a matchmaker has a convenient dual-purpose representation upon which to base its operations.

In DAML-S, the requester relies on infrastructure

components that act like registries to find

the appropriate provider. These registries provide the matching between the request and the offers of service providers to identify which of them is the best match. Within DAML-S, the service profile provides a way to describe the services offered by the provider, and the services needed by the requesters, although it does not mandate any representation of services, being possible to create specialized representations of services that can be used as service profiles.

The profile implicitly specifies the intended purpose of the service by means of inputs and outputs (information transformation) and prerequisites and effects (state change). Considering the hotel booking example presented in the requirements section, information transformation talks about the input (credit card number and expiration date) and output (charge notification), as it transforms real data provided by the requester. On the other hand, state change refers to how the service affects to the "state of the world", and it is given by the pre-condition (valid credit card) and the post-condition (requester account charged). The service profile advertises those functionalities which the service wants to provide, while it may hide (not declare publicly) other functionalities.

The other key element for DAML-S service composition is the service model, and especially its subclass *ProcessModel*. The Process Ontology has a primary kind of entity called "process", which gives information about inputs, outputs, preconditions and effects, and which is intended to describe how the service work and to enable planning, composition and agent/service interoperation. It can strengthen or weaken the profile descriptions of preconditions, post-conditions, inputs and outputs. It can be an atomic process, a simple process or a composite process:

- Atomic processes are directly invocable, have no subprocesses and execute in a single step from the point of view of the service requester.
- Simple processes are not invocable and are not associated with any grounding, and like atomic processes they are conceived of as having single step executions. They are used as elements of abstraction, to provide a view of a specialized way of using some atomic process or as a simplified representation of some composite process (for purpose of planning and reasoning).
- Composite processes are decomposable into other (non-composite or composite) processes; their decomposition can be specified by using control constructs such as SEQUENCE and IF-THENELSE. Such decomposition normally shows how the various inputs of the processes are acceptedby particular subprocesses, and how its various outputs are returned by particular subprocesses.

Processes can be viewed at different levels of granularity, either as primitive, indecomposable or as a composite process, which are usually called "black box" and "glass box" views, respectively. A composite process must have a *composedOf* property by which is indicated his control structure, using a CONTROLCONSTRUCT.

Each CONTROLCONSTRUCT is associated with an additional property called components to indicate the ordering and conditional execution of the subprocesses (or other control constructs) from which it is composed. DAML-S process upper ontology includes a minimal set of control constructs that can be specialized to describe a variety of Web services, including Sequence, Split, Split+Join, Choice, Unordered, Condition, If-Then-Else, Iterate, Repeat-While and Repeat-Until.

Once the process composition has been specified, data flow has to be defined. As DAML+OIL does not provide for the use of variables, DAML-S use process annotations to specify how processes are to be instantiated and information shared between process elements. A specialized DAML-S reasoner is required to use this information to determine which properties should have the same value in any coherent instance of the process being defined.

A process control ontology, which provides a model to interpret process instantiations to monitor and control the execution of a process, is to be defined within DAML-S. Although it isn't defined at the current version of the framework, it is expected to provide:

- Mapping rules for the various input state properties (inputs, preconditions) to the corresponding output state properties.
- A model of the temporal state dependencies described by control constructs.
- Representations for messages about the execution state of atomic and composite processes sufficient to perform execution monitoring.

The elements presented so far describe the composition model used within DAML-S and the features still to be defined. Although it provides a quite complete composition model, it presents some caveats and some of the described requirements for an effective SWS composition are not fulfilled.

Considering the identified static requirements for composition, DAML-S presents some deficiencies concerning the de-coupling of web service components. A Web Service can be exposed as a composite process, thus giving information about how the service is internally composed. The DAML-S "glass box" view reveals too much information about the internal composition of the service, and therefore it violates the required separation between private and public aspects of the service. Although DAML-S also provides ontological support to use a black-box approach, it shouldn't allow the "glass box" level of granularity for a service.

On the other hand, DAML-S fits to our criterion concerning the ontology-based service descriptions, as well as the separation of goal description and web service description. It defines a concise but still flexible ontological description of the services, and the profile description exposing what the service offers to the user is clearly separated from the specific service description. Although DAML-S specification does not include any security and reliability feature, some work on security annotation for DAML-S services is being done [Denker et al. 2003].

About dynamic requirements for

composition, DAML-S separates process control and service ontologies. Process control ontology is still under definition, so in its current status DAML-S does not enable services monitoring.

Nevertheless, its definition is explicitly considered within DAML-S specification. One important caveat of DAML-S is that it does not offer a clearly defined approach about exception and fault handling. Process control ontology would allow services monitoring, although it gives no clue about the strategy to follow.

As DAML-S is not intended to provide a framework but an ontology for services, it can not be evaluated under our framework criteria.

Considering the overall criteria evaluation, DAML-S includes interesting features to achieve SWS composition but lacks some of the basic requirements identified to bring WS composition to its full potential. Especially interesting are its ontologies definitions, as they provide a solid base while enough flexibility to be used in the context of dynamically composed semantic web services.

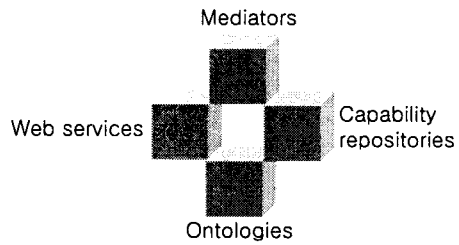
3.3 WSMF

WSMF aims to provide a comprehensive framework to achieve automatic web service discovery, selection, mediation and composition into complex services, that is, to make semantic web services a reality and to exploit their capabilities.

In order to fully enable e-commerce based on workable web services, WSMF is centered on two complementary principles [Fensel et al. 2002a]:

- **Strong *de-coupling*** of the various components that realize an e-commerce application. This decoupling includes information hiding based on the difference of internal business intelligence and public message exchange protocol interface descriptions. Coupling of processes is achieved via interfaces to keep the amount of interactions scalable.
- **Strong mediation** service enabling anybody to speak with everybody in a scalable manner. This mediation service includes the mediation of different terminologies as well as the mediation of different interaction styles.

To fulfill these essential principles, mediators which map between different document structures and different business logics as well as the ability to express the difference between publicly visible workflows (public processes) and internal business logics of a complex web service (private processes) are key elements in the framework.



[Fig.2] WSMF main elements

WSMF consists of four different main elements (Fig 2):

- **Ontologies**: Provide the terminology

used by other elements. Are used to define the terminology that is used by other elements of WSMF specifications, with the purpose of enabling reuse of terminology as well as interoperability between components referring to the same or linked terminology.

- **Capabilities repositories**: Define the problems that should be solved by web services. They are composed of two elements: pre-conditions, which specify what the web service expects in order to enable it to provide the service, and post -conditions, that show the estate after execution. Capability specifications are kept separate from actual web service descriptions because there is an n2m mapping between them, and to enhance web services discovery.
- **Web services descriptions** : Define various aspects of a web service. They are described as black boxes consisting of *Web service name, Capability reference, Pre and post-conditions,* and *input and output data.* Apart from these aspects WSMF considers other properties important for the service. Among them: Failure, concurrent execution, concurrent data input and output, dynamic service binding, data flow, control flow, exception handling, and compensation.
- **Mediators** : Bypass interoperability problems related to data structures, business logics, message exchange protocols and service invocation. They deal with the requester and provider heterogeneity, performing the necessary operations to enable fully interoperation between the requester and the provider. Mediation can follow different underlying approaches, such as a client/server, peer to peer or mediation enabled peer to peer approach. The approach preferred in WSMF is mediation enabled peer to peer, as it provides better scalability and a higher degree of transparency for both the requester and the provider. However, some limitations of this approach make still consider other interaction styles.

These main elements of the framework are put together to provide automatic web service discovery, selection, mediation and composition into complex services.

Using this framework, discovery strategies which take advantage of all these features can be implemented, as well as selection, mediation and composition strategies. Also the framework counts with capabilities to support exception handling trough a proxy component, which will take care of retrying the execution of the web service in case it fails.

There is no restriction about the model used for composition. Composition can be achieved as providerbased composition, offering ready-to-use complex web services, as client-based composition, relying on agents discovering and combining the required services, or as a declarative web service composition, where just goals tobe fulfilled are declared and services are discovered, selected and invoked dynamically.

When taking a close look to the requirements stated in section 2 for semantic web service composition, the features of WSMF happen to accomplish most of them as shown in detail in the coming paragraphs.

Static requirements:

- **Strong decoupling**: It is one of the foundation principals of WSMF, so the framework clearly presents this feature.
- **Ontology based service description**: Ontologies are used as a vocabulary to describe the specification of the web service. They enable composition of complex web services as well as interoperability. The framework also accomplishes this requirement
- **Separation of goal description and web services description**: WSMF makes a clear separation among these two elements. The service description is stored in one part of the framework and service advertisement and public process description in a different one.
- **Proof digital signature**: At the moment the framework doesn't provide support for such a characteristic, also it is supposed to be extendable.

Dynamic requirements:

- **Workflow monitoring**: From the perspective of gray box approaches WSMF provides the means to monitor the web services workflow.
- **Separation of process control and service ontologies**: The framework is not clear about this point, it provides the means to define and store ontologies, but it should be clarified whether it is supported or not
- **Exceptions and faults handling logic**: As stated in [Fensel et al. 2002a] the framework provides the means to handle exceptions and faults, by means of compensation strategies.

Web Service Framework requirements:

- **Strong mediation facilities**: Mediation is one of the foundational principles of SWSMF as stated in [Fensel et al. 2002b].
- **Persistence of service context**: It is not specified in the framework, it may be included in coming releases.
- **Exceptions and faults handling**: Supported by the framework; it is implemented through compensation mechanisms.
- **Automatic discovery of services**: This is a basic functionality of any web service framework. It is supported in WSMF.
- **Management of service proxies**: It counts with different levels of acknowledgments spread among different layers of the framework.
- **Maintenance of ontology repositories**: Not included in the framework.

Summarizing, WSMF presents the core set of features required for SWS composition. Although some of them need to be further

defined, WSMF, when developed in complete detail, presents the necessary characteristics to be a suitable framework for next generation web services.

4. Conclusions

The definition of precise requirements for semantic web service composition provides a context in which current efforts can be evaluated. And this evaluation draws conclusions and guidelines about current state of the art and future steps. BPEL4WS, one of the strongest industrial efforts in the WS composition area, is not providing any semantics, so it is difficult that it could achieve real automatic WS composition, as semantics are a key element for next generation Web and next generation Web Services.

On the other hand, DAML-S and WSMF have faced the challenge from the perspective of providing explicit and formal semantics to Web Services. They share a common area of interest, and they are focusing in slightly different but related aspects. WSMF tries to provide a framework for SWS description, discovery, composition, execution and interoperation by defining the elements needed for such a framework, while DAML-S is focusing on ontology definition to describe, locate and compose services in an automatic manner. Both approaches are fulfilling most of the requirements for automatic composition, although especially DAML-S lacks some essential characteristics to provide a full SWS composition support. Nevertheless, WSMF is providing an important context which could integrate efforts in the area, as it is addressing the key points to bring services to its full potential. As DAML-S is defining some complementary issues, as ontology support for SWS definition, it could be used in combination with WSMF to provide a comprehensive framework for SWS composition. In spite of the interest of this approach, new initiatives may appear to really take the Web to a new level of service. In any case, future steps should take into account a complete set of requirements for effective composition, as the ones presented in this paper.

References

[1] Arkin, A. et al, Web Service Choreography Interface (WSCI) 1.0, available at *http://www.w3.org/TR/2002/NOTE-wsci-20020808/*

[2] Benatallah B., Dumas M., Fauvet M.-C., and Rabhi F. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi, editors, Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, UK, Nov 2002.

[3] Burbeck, S. The Tao of e-business services - The evolution of Web applications into service oriented components with Web services. *IBM Software Group, Oct 2000. available at http://www-106.ibm.com/developerworks/webservices/library/wstao/?dwzone=webservices*

[4] Bussler C., Fensel D., Maedche A. A Conceptual Architecture for Semantic Web Enabled Web Services. SIGMOD Record 31(4): 24-29, 2002 Environments," *J. ACM SIGMOD*

Record, vol. 28, no. 1, Mar. 1999, pp. 47-53.

[5] Curbera, F., Goland, Y., Klein, J., Leyman, F., Soller, D., Thatte, S., Weerawarana, S., Business Process Execution Language for Web Services, BEA Systems & IBM Corporation & Microsoft Corporation, 2002, http://www-106.ibm.com/developerworks/library/ws-bpelwp

[6] The DAML Service Coalition, DAML-S: Semantic Markup for Web Services, 2003, available at *http://www.daml.org/services/damls/0.9/*

[7] Denker, G., Kagal, L. Security annotation for DAML Web Services, 2003

[8] D. Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag, Berlin, 2001.

[9] D. Fensel, D. Bussler C. The Web Service Modeling Framework. First International Semantic Web Conference, Sardinia, Italy, June 2002

[10] D. Fensel and C. Brussler. The Web Service Modeling Framework, In White paper and Internal Report, Vrije Universiteit, Amsterdam, 2002.

[11] F. Layman. Web Services: Distributed Applications without Limits, Proceedings Database Systems for Business, Technology and Web, Leipzig, Germany, Feb., 26-28, 2003

[12] Mcllraith, Sheila, "Semantic Enabled Web Services", XML-Web Services ONE Conference, June 7, 2002.

[13] Peltz, C. Web Services Orchestraion, HP White paper, 2003, available at vresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf

[14] Sheth et al, Web Services: Been There, Done That?, IEEE intelligent systems, Jan./Feb. 2003, pp 72-85

[15] Sollazo, T., Handschuh, S., Staab, S., Frank, M. Semantic Web Service Architecture - Evolving Web Service Standards toward the Semantic Web., Proc. of the 15th International FLAIRS conferences. Pensacola, Florida, May 16-18, 2002. AAAI Press.

[16] K. Sycara et al., "Dynamic Service Matchmaking among Agents in Open Information

[17] Zhengang Cheng, Munindar P.Singh, and Mladen A. Vouk, Composition Constraints for Semantic Web Services, *WWW2002 Workshop on RealWorld RDF and Semantic Web Applications*, May 7, 2002.

최병석

1981년 조선대학교 과학교육과 졸업(이학사)
1987년 고려대학교 과학교육과 졸업(이학석사)
2002년 원광대학교 일반대학원 컴퓨터공학과 박사과정수료
1996년 ~ 현재 군장대학 컴퓨터응용학부 조교수
관심분야 : 인공지능, Semantic Web Service, XML

홍현술

1985년 원광대학교경영학과 졸업(경영학사).
1987년 원광대학교대학원 경영학과졸업(경영학석사).
1990년 원광대학교대학원 전자계산기공학과(공학석사).
2001년 원광대학교 대학원 컴퓨터공학과 졸업(공학박사).
1989년 - 현재 원광보건대학 컴퓨터응용개발과 교수.
관심분야 : 객체지향시스템, 디자인패턴, WBI, 컴퓨터 응용, 온톨로지시스템, 시멘틱웹

한성국

1979년 인하대학교 전자공학과 (공학사).
1981년 인하대학교대학원 전자공학과 (공학석사).
1988년 인하대학교대학원 전자공학과 (공학박사).
1984년 - 현재 원광대학교 전기전자 및 정보공학부 교수.
1988년 - 현재 미르칸전자 기술고문.
1989년 - 현재 ㈜비트 컴퓨터 기술고문.
1990.12월 - 1992.2월 미국 펜실버니아 대학교 Post-Doc.
2003.2 - 현재 오스트리아 인스부르크대학 객원교수
관심분야 : 인공지능(자연언어처리), 정보공학, 인지과학, 컴파일러, XML, 컴퓨터교육, 온톨로지시스템, 웹서비스, 시멘틱웹