PRACE NAUKOWE AKADEMII EKONOMICZNEJ WE WROCŁAWIU
Nr 1134 — 2006
Nowoczesne technologie informacyjne w zarządzaniu

**Witold Abramowicz, Tomasz Kaczmarek,
Marek Kowalkiewicz, Karol Wieloch**

A BRIEF SURVEY OF WEB SERVICE DESCRIPTION LANGUAGES

1. Introduction

Recently an idea to automate web service handling and managing has emerged. The automation requires detailed and precise service description as it was done with goods one electronic markets. The activities that benefit from having service description are as follows.

- To be able to execute the external service, one has to find it first. In the real world we use *yellow pages*, which have its counterpart in the software world - service registries. Service registries store searchable service descriptions. The discovery process refers to finding a service within this registry, which is most suitable for given user request. The quality of service discovery depends on the quality of service description, which is evident given recent research in [Fan, Kombhampati 2005], on the publicly available Web services. Fan and Kambhampati found that the ser-vices registered in public repositories are often not documented, not to mention proper description, which seriously impaired their discovery attempts.
- With the emergence of industry standards for developing services a new application architecture paradigm has emerged – Service Oriented Architecture (SOA). It assumes that the business applications are built of services that interact with each other. It is a common need for business applications to evolve, and using SOA this could be achieved. Services, as self-contained functionalities, could be exchanged on demand and the whole application could change its behavior even at a runtime.
- If the description is detailed enough, it is hoped to facilitate automatic service composition. This means building complex services out of simple (or atomic) services.

- With proper service descriptions, services can be invoked automatically i.e. an application that wants to execute a service could automatically check whether it can provide all the necessary input data and is capable of receiving output from the service. Further, it would be capable of starting the service with intended input.
- Interoperability, defined as the ability of a systems to work with other system without special effort required from customer, or the ability to successfully communicate (over the network), or to exchange and use information that has been exchanged, boils down for the Web services to be able to participate in the same process and interpret results from of the other services. Semantic description of inputs, outputs and service functionality greatly facilitates interoperability.
- Except from the functionalities provided by the service, it may be described with some additional, non-functional properties which are potentially relevant for the service requestor. First the party that invokes the service is interested in the result (success of failure due to some error); second it could be interested in features which are usually referred to as Quality of Service (QoS) properties. Service description should indicate what the means to communicate errors are and what the QoS properties that the service provider promises are.

In general a service may be defined as „an action performed by one entity on behalf of another" [O'Sullivan, Edmund, Hofstede 2002a]. It is often defined through features opposite to goods. These include:

- nature of the act (service is often provided, bought and consumed in one place/ /time, and its quality can not be assessed beforehand),
- number of parties involved,
- the manner in which service is requested and delivered,
- the fact that services can not be mended (like goods),
- service volatility – they can not be stored or returned,
- tight connection to the service provider,
- the type of value that is transferred (usually there is no change in the ownership as opposed to goods).

Another important point is that services are often composites – they contain other sub-services. This implies that it is possible to compose services, however at least two kinds of such compositions are possible [O'Sullivan, Edmond, Hofstede 2006]. One is aggregation – services of similar category are grouped and delivered as a single service (e.g. internet portal). The other is composition where composed services improve or add each other value (like improved security for a credit card transaction service). From the perspective of e-services (introduced shortly) this feature seems important and introduces serious complexity in handling services with hardware.

Once we move to a more technology oriented environment, we can define the electronic services as activities performed by one process on behalf of the other. It

seems however that such a definition would be too broad. In literature e-services usually refer to workflow and internet environment as means to accomplish tasks [O'Sullivan, Edmund, Hofstede 2006]. As such e-services are not services that are provided with some connection to computing hardware (like website hosting service or internet provision service). They are more specific – they are performed on the hardware, as software procedures. This view is supported in [Preist 2004] who defines e-service as „software entity able to provide something of value [to the requestor]". Nevertheless researchers are far from agreement on the use of this term (see [Preist 2004] for contradictory views).

1.1. Web services

Research community proposes the following definition of web services: „Web services are software services distributed on the internet" [Fan, Kambhampati 2005]. W3C says that „A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format [...]" [Dhesiaseelan 2004]. These and numerous other approaches have certain similarities – they characterize Web service as:

- software (electronic),
- accessible via internet protocols (like HTTP),
- invoked in a platform and language independent manner (XML plays significant role),
- interoperable.

Five years from their birth, Web services became widespread software engineering technique, useful especially for application integration and a basis for Service Oriented Architecture paradigm. The need to compare, compose and exchange Web services emerges slowly given the increasing volume of available services published in both intranet and Internet repositories. Therefore the need to compare the languages that are used for service description, because their strength influences the possible scenarios of service discovery, composition and exchange.

2. Languages

There are several languages currently used (or under development) for descrybing Web services. Each of them has slightly different focus, yet they share common goal: to enable discovery, invocation and matchmaking of Web services as software components.

WSDL (Web Service Description Language, version 1.1) [*Web Services...* 2004] provides a XML grammar to for describing network services as collections of communication endpoints capable of exchanging messages. In WSDL, the

abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings, which allows the reuse of abstract definitions. A WSDL document uses the following elements in the definition of network services: type, message, operation, port type, binding, port, service.

OWL-S [Martin i in. 2004] is an ontology (in a sense of set of concepts, also referred as a language) for describing Web services. The structure of the language is motivated by three kinds of information end user needs to know about. These are: service functionality (what it provides to its clients), service interaction model (how it issued) and service grounding (how it is accessed, what protocols it uses, etc). According to this distinction the main language concepts are: ServiceProfile, ServiceModeling and ServiceGrounding. These are formally subclasses of class Service.

WSMO (Web Services Modeling Ontology) [Roman i in. 2005] is a highly formalized conceptual model designed to describe various aspects of Semantic Web services. It uses WSML (Web Services Modeling Language) [Bruijn 2005] which with its formal semantics facilitates logical reasoning on service descriptions. Together with WSMX – Web Service Execution Environment – they constitute a framework for comprehensive service description and execution. The central concepts in WSMO are: ontologies – provide formal semantics to the information used in other components, web services – semantically described behavioral aspects of software entities, goals – express potential user objectives, mediators – elements that cope with semantic and structural inconsistencies that occur in the heterogeneous environment of Web Services.

WSRF (Web Services Resource Framework) [Czajkowski i in. 2004] is an initiative of Globus Alliance (see [O'Sullivan, Edmond, Hofstede 2002b]) as a successor of OGSA/OGSI framework. Its core concept is a WS-Resource [Graham i in. 2005] – a composition of a stateless Web service and a stateful resource. WSRF is a set of specifications of message exchanges to properly interact with WSResources. The specifications says how to manipulate WS-Resource properties, how to group them, how to mange lifetime of a stateful resource, how to model basic error messages and how to organize them for topic based subscriptions.

WSDL-S (see [Akkiraju i in. 2005]) is a language that extends existing WSDL standard with semantic description. The semantic service model is assumed to exist outside of the WSDL-S description and is referenced via WSDL extensibility elements. It is thus decoupled from the particular service description model and ontology language. The additional elements are: types and complex types that relates to an external ontology, categorized interfaces and operations (conditions, inputs, outputs) related to an external semantic model.

3. Comparison

3.1. Semantic description

Languages like WSMO, OWL-S or WSDL-S approach a service description problem from the knowledge representation and reasoning point of view. These languages aim at describing in an unambiguous way *what* a service does. Their common denominator is that they adopt ontologies for that purpose. The other family of languages – WSDL, WSRF, SSDL – is more grounding oriented. They are concerned with technical aspects of service description: defining types, protocols, formats, message exchange patterns. Therefore it is often hard to find correspon-ding features in both families. Some, as WSDL-S, aim at bridging the gap and extend standard WSDL to give at least parameters, their types and messages some semantics. Some, as WSRF, go in different direction, focusing at describing state and stateful resources, dealing with services at a level of software design patterns that are necessary and most suitable for the specific (GRID) environment.

3.2. Decoupling of provider and requestor views

Depending on the envisioned use cases, languages for service description differ in providing means to distinguish between what is provided (service capability) and what is requested (user goal/query/request) [Roman i in. 2005]. WSMO was designed with strong decoupling in mind, therefore it separates users' goals from services and provides means to mediate between the two, even if the ontologies used to describe both differ. OWL-S, on the other hand, does not go so deep into separating requested capability from provided one (everything is referred in the service profile; user request is a profile of a service that would be most suitable). Closely related is the possibility to use different ontologies to specify requests and service capabilities. It seems that WSMO goes the deepest here, allowing importing and use different ontologies, however it does not provide means to declaratively or procedurally state the transformations necessary to mediate between these ontologies. It only gives pointers to hand-made mediators. WSDL-S through its WSDL extension elements provides some means to mediate between types defined in the service description (WSDL) and the external ontology. They can use XSLT declarative language. OWL-S requires consistent ontology to describe services and user request (expressed as service profile), leaving mediation possibly to external tools.

3.3. Non-functional properties

All the presented languages differ in the approach to non-functional properties. Since the possible attribute set for non-functional properties is broad ([Fan, Kambhampati 2005]) and subject to extensions, it is best if a language provides means to include additional non-functional properties. It is directly available in OWL-S and indirectly (through ontology extension) in WSMO.

3.4. Input/output parameters, preconditions/postconditions

An important factor for describing service capability is a specification of its input/output parameters and the way service execution changes a world state. The first part is provided in all the languages described. They differ in degree of semantics that is assigned to the input/output parameters. Syntax based languages such as WSDL can be extended to provide semantically described parameters (WSDL-S), while semantic enabled languages (WSMO, OWL-S) differ slightly in the way parameters are defined. In OWL-S they are explicitly part of the service description, in WSMO they sit in assumption /preconditions/postconditions/effects as variables. OWL-S allows for greater flexibility in the logical expressions that show the state change as result of service execution – it allows for arbitrary language to specify the expressions. It is however not clear how different languages should interoperate in the description. WSMO on the other hand sticks to the WSML family of languages on that matter.

3.5. Service interfaces and choreography

Service description languages differ in the number of interfaces that can be assigned to one service. Simple languages (WSDL) follow some object oriented concept allowing for multiple interfaces. This provides greater flexibility for transforming object-oriented components into Web services. The drawback of this approach is that it is not easy to give semantic description to multiple interface service. Each interface appears with its own semantics, that is why OWL-S allows only for one service model for a service, and WSMO allows for one capability but multiple ways to interoperate with the service (multiple choreographies).

3.6. Composite processes

Handling complex processes is a difficult issue (as indicated in the beginning) and one of the most challenging parts for service description languages is to describe service choreography or orchestration. In OWL-S composite processes can define functionality of aggregated processes (services) explicitly showing which components constitute a composite service. In WSMO it is not possible to derive from the preconditions and postconditions of a composite process, what was the functionality of its components.

3.7. Grounding

The actual protocols and formats to invoke the service constitute service grounding in different description languages. Some of them are mainly build around this topic (WSDL and similar), some base on WSDL standard (OWL-S,WSRF) in their grounding parts, and some do not clearly tackle grounding issues (WSMO – it will be probably mapped to WSDL in grounding issues for standard compatibility).

Table 1. Language comparison

		OWL-S	WSMO	WSDL	WSRF
Discovery	**match-making**	one step comparing 2 service profiles	3 steps of discovery: – goal formalization – discovering potential services (e.g. by keywords) – choreography and capabilities comparison	keyword based categories	mechanism of notifications distributed discovery of resources service of independent groups WS-Resource properties natural language description
	registry	single	single	single (UDDI)	
	capabilities	input, output, precondition, effect logical expressions (language not specified)	assumptions, preconditions, postconditions, effects	syntactical choreography	
	non-functional properties	supported, extensible	supported		
Composition and substitution		– choreography in Service process model – composite services based on workflow-like control structures	choreography and orchestration based on states (described by ontology) and guarded transitions	only for atomic operations; no means to deduce dependencies between separate service specifications	just grouping service group, no workflows, interface based on WSDL
Execution		service grounding to WSDL	grounding to WSDL-S (extensions)	bindings to SOAP HTTP	design patterns for WSDL
Interoperability		OWL, WS-Addressing	mediators	this is standard	WS-Addressing, becomes standard in grid environment
Monitoring		extensible, properties to hold quality parameters	extensible	subject to extensions	lifecycle management, base faults, notifications

3.8. Expressivity of logic languages

Last but not least there is a problem of expressivity of logical language used to define semantics of service description. Allowing for wide spectrum of languages – like OWL-S does – detaches service description language from the description of the world that is changes, i.e. the language for describing service may be different from the language describing the world it changes. Though, due to this flexibility OWL-S approach suffers from interoperation problems. It is not clear what reasonning mechanism would be capable of processing such separate descriptions and what should be its logics model. On the other hand adopting one-language-WSMO approach places restriction on providers (have to use single language) and requires tools (reasoners, editors, validators) to be built for that language.

Table 1 shows the most important features of the compared languages. They are grouped according to the influenced activities of discovery, composition, substitution, execution, interoperability, and monitoring. The table focuses on the most significant service description efforts, leaving languages like WSDL-S and SWSF (Semantic Web service Framework out. These are too young to judge its impact and application in the areas of discovery, composition, execution, interoperability and monitoring, and they did not mature enough to be supported by and tools dealing with the mentioned processes.

4. Conclusions

This article provides comparison between state-of-the-art service description languages available and under development in research community. It evaluates them in terms of suitability for service discovery, substitution, composition, execution, interoperation, and monitoring. The main findings are that there are four points of view on Web service description.

WSRF and generally Grid Web services focus on resources management, especially with respect to their state. The approaches flatten notion of Web service to interface between user and the resource.

On the opposite side there are frameworks like WSMO, OWL-S or SWSF that put stress on semantic description of service capabilities. They allow expressing service functionality. Their weakness is grounding to communication and execution standards. The reason why it is not trivial is that actually communication and description standard for Web services – WSDL was developed without rich semantics in mind.

The third point of view on services focuses on service interaction patterns – choreography. As far as composite services are considered it is necessary to deal with message exchange patterns that are more complex than simple input-output. Most of presented specification languages enable modeling of complex interaction patterns. Unfortunately they are not compatible when grounding is considered from

abstract conceptual models (e.g. WSMO, OWL-S) to technical communication standards (WSDL). There are basically two approaches to deal with the problem. In the first one, conceptual models are mapped onto WSDL language (e.g. OWL-S). The other approach adds some extra elements to WSDL descriptions and puts into them necessarily semantic meaning (e.g. WSDL-S).

The last point of view is a simplistic, technology oriented languages like WSDL and SSDL. They are limited to interface description and selection of communication protocol and format. Regardless of this simplicity (or thanks to that) they received wide acceptance and became standards for non-semantics oriented applications.

References

Akkiraju R. i in., *Web Service Semantics – WSDL-S. Technical report*, A joint UGA-IBM Technical Note, kwiecień 2005.

Baida Z. i in., *A Shared Service Terminology for Online Service Provisioning*, ICEC '04: Proceedings of the 6th international conference on Electronic commerce, ACM Press 2004.

Bruijn J., *The Web Service Modeling Language WSML*, Final draft, DERI 20 marca 2005.

Christensen E. i in., *Web Services Description Language (WSDL)* 1.1. http://www.w3.org/TR/wsdl, marzec 2001.

Czajkowski K. i in., *From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution*, Whitepaper Version 1.1, International Business Machines Corporation and The University of Chicago, maj 2004.

Dhesiaseelan A., *What's New in WSDL 2.0*, Technical report, 2004.

Fan J., Kambhampati S., *A Snapshot of Public Web Services*, „SIGMOD Rec.", 2005 34(1), s. 24-32.

Globus Alliance – WSRF Website. http://www.globus.org/wsrf.

Graham S. i in., *Web Services Resource* 1.2. Working Draft 03, OASIS, marzec 2005.

Lara R., i in., *A conceptual Comparison between WSMO and OWL-S. WSMO*, Final Draft D4.1 v0.1, DERI, kwiecień 2005.

Martin D., i in., *Owl-s: Semantic Markup for Web Services*, Whitepaper Version 1.0, DARPA, listopad 2004.

O'Sullivan J., Edmond D., Hofstede A.H.M., *Service Description: A Survery of the General Nature of Services. Technical Report FIT-TR-2003-02*, Queensland University of Technology, kwiecień 2002a.

O'Sullivan J., Edmond D., Hofstede A.H.M., *What's in a Service*? „Distributed and Parallel Databases", 2002b nr 12, s. 117-133.

Preist C., *A Conceptual Architecture for Semantic Web Services*, Proceedings of the Third International Semantic Web Conference, listopad 2004.

Roman D. i in., *Web service modeling ontology (WSMO)*, kwiecień 2005.

Semantic Web Services Framework Website, http://www.daml.org/services/swsf.

Web Services Architecture, W3C Technical Note, 2004.

PRZEGLĄD JĘZYKÓW OPISU USŁUG SIECIOWYCH

Streszczenie

Artykuł zawiera analizę porównawczą języków opisu usług sieciowych. Po ogólnym zarysowaniu problemu i podkreśleniu istotności opisywania własności pozafunkcjonalnych usług, skupiamy się na porównaniu czterech języków opisu usług sieciowych: OWLS, WSMO, WSDL oraz WSRF.

Prof. dr hab. Witold Abramowicz jest kierownikiem Katedry Informatyki Ekonomicznej Akademii Ekonomicznej w Poznaniu, e-mail: witold@abramowicz.pl.

Mgr Tomasz Kaczmarek jest asystentem w Katedrze Informatyki Ekonomicznej Akademii Ekonomicznej w Poznaniu e-mail: T.Kaczmarek@kie.ae.poznan.pl.

Mgr Marek Kowalkiewicz jest asystentem w Katedrze Informatyki Ekonomicznej Akademii Ekonomicznej w Poznaniu, e-mail: m.kowalkiewicz@kie.ae.poznan.pl.

Mgr Karol Wieloch jest asystentem w Katedrze Informatyki Ekonomicznej Akademii Ekonomicznej w Poznaniu, e-mail: k.wieloch@kie.ae.poznan.pl.