# Dagstuhl Seminar on Service-Oriented Computing

# Session Summary “Cross Cutting Concerns”

**Heiko Ludwig, Charles Petrie**

## Participants of the Core Group

Monika Kazcmarek, University of Poznan

Michael Klein, Universität Karlsruhe

Heiko Ludwig, IBM TJ Watson Research Center

Michael Maximilian, IBM Almaden Research Center

Charles Petrie, Stanford University

Pierluigi Plebani, Politecnico di Milano

Stefan Tai, IBM TJ Watson Research Center

Xia Wang, Fernuniversität Hagen

## Introduction to the session and the proceeding of the work group

A session on cross cutting concerns of Web services necessarily addresses a multitude of domains, unlike the other “core” working groups focusing on development and life-cycle, foundations, composition, and management. A cross-cutting issue is a topic that relates to a multiple or all of the core subjects but cannot be address in the context of a single core workgroup.

The working group identified the following set of cross cutting concerns and worked in further detail on the bold faced ones, as outlined in the subsequent sections:

- **Definitions** of the notion of a service as a Web service.
- The issue of **semantics of Web services**. Issues of explicit semantics are relevant throughout the life-cycle of a Web service, i.e. during development, discovery, composition, and invocation of a service.
- Besides its role as means of enterprise application integration, Web services were always thought of a way to integrate functionality of different business, which would offer services commercially. The model of Web service suitable as a business service, its design and management, its integration as well as providing the legal, accounting, regulatory context are not investigated at this point.
- Web services might not only be performed as electronic services but also may include, or wrapper, service elements that are performed by employees or have effect in the real world beyond the scope of the IT systems. Web services with external effects require in particular a deep understanding how to address contingencies or execution failure and issues of asynchronous

execution of services and delays in synchronizing real world state with its representation in the service system.

- **Quality of Service (QoS)** relates to “non-functional” characteristics of a service. It is often associated with performance and availability-related parameters such as performance and downtime but also relates to other properties of a service, e.g., to run on a virus-checked platform or the ability to participate in types of coordination. Definition, enablement and monitoring of QoS is a cross cutting concern that spans development and life-cycle management but also and in particular systems management.
- A service must be able to be adapted to run in different contexts. This may apply to the specialization of the interface of a service as well as adapting a composition to a specific need. How much can a service or composition be repurposed? This touches on multiple aspects of a service, in particular the development, composition and life cycle management.
- Security and establishment and preservation of identity and privacy are classic cross cutting concerns.

## Definition of a Service as a Web Service

We worked off an agreed wide definition of a Web service in the context of this working group:

A service is:

- A possibly remote procedure with an
- invocation that is described in a standard (preferably XML-based) machine-readable syntax,
- reachable via standard Internet protocols,
- including at a minimum a description of the allowed input and output messages, as well as
- possible semantic annotation of the service function and data meaning.

## Semantics in the Context of Web Services

Work group: Core group + Bernhard Steffen and Tiziana Margaria

### Scope and Issues of Semantics in Web services

Web services often cross boundaries of administrative and organizational domains. Common understanding of terms, be it the meta-data describing a service or the parameters marshaled in an invocation, cannot be assumed under these circumstances. Implicit common semantics, which can be assumed in a centrally coordinated environment, must be made explicit to enable the correct interpretation of terms across boundaries. These domains of implicit semantics often coincide with organizations or a specific software package such as an ERP system.

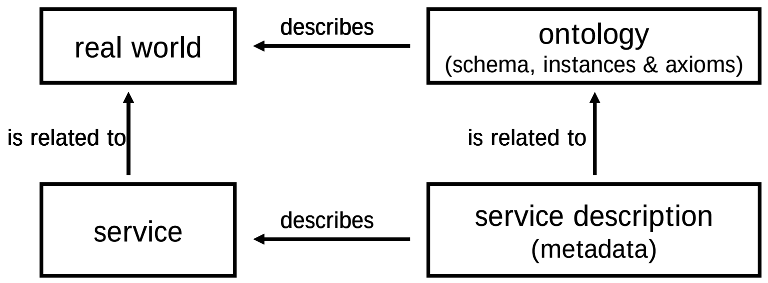
Semantics of Web services is to be considered throughout the life-cycle of a Web service and by different constituents.

- Upon **implementation of a Web service**, the implementer must be aware of the intended semantics of the parameters of messages related to the service.
- Correspondingly, **implementers of a Web service client** must be aware of the semantics of message elements.
- **Advertising** a service to a directory service or other match-making mechanism requires the establishment of shared semantics of meta-data items with potential users and search and match-making intermediaries. Often, directory services are the venue of establishing shared semantics of meta-data terms

- Likewise, **searching and match-making** is based on the awareness of the semantics of meta-data and must be considered in the process of binding a Web service client to a specific Web service.

Formalization of semantics is relevant in particular for automated steps in the Web services life-cycle, in particular the match-making and binding process.

The most common approaches to formalize semantics are the **ontology** and the vocabulary. The following figure illustrates the role of ontologies in the context of Web services.



In general, ontology is defined as a specification of a shared conceptualization of a specific domain. Typically, this is seen as comprising a schema for describing and deriving concepts, a set of instances and axioms. The schema may comprise rules, inheritance and other mechanisms how to derive concepts from a set of existing ones. If no axioms and rules are present, an ontology is a **vocabulary**, the simpler form.

## Current state of the art

A number of approaches to define and use ontologies through the phases of a Web services life-cycle have been proposed and used.

The main approaches to ontology are the following:

**OWL-S** is an OWL (Ontology Web language)-based ontology framework that provides Web services providers with an ontology of Web services and the means to describe:

- The service profile; what the service does. This is to be used for match-making.
- The service model; how it works. This could, e.g., include a description of a process being performed to enact a service operation and provides deeper understanding to the service client, e.g., for monitoring.
- The service grounding; how to access the service. This relates directly to WSDL definitions of service interfaces. Basic OWL-S grounding elements can be included as extension elements in a WSDL specification.

Further information: http://www.daml.org/services/owl-s/1.0/

The Web Services Modeling Framework **(WSMF)**, comprises **WSMO**, the Web Services Modeling Ontology, the Web Services Modeling Language (**WSML**) and describes its execution semantics using the Web Services Execution Environment (**WSMX**). One unique objective of the approach is to enable decoupling between Web services providers and clients to the greatest possible extent using mediators. Based on a defined ontology, clients phrase goal, e.g., as post-conditions which are then matched against descriptions of Web services by suitable mediators.

Further information: http://www.wsmo.org/

**FLOWS**, the First order Logic Ontology for Web Services, is an axiomization of OWL-S based on first order logic introducing a set of process-related constructs that can be used to describe service behavior. It is part of Semantic Web Services Framework (**SWSF**), along with other specifications such as the Semantic Web Services Language and the Semantic Web Service Ontology. Overall, SWSF can be seen as a logic-based approach to establish the semantics of a Web service.

Further information: http://www.daml.org/services/swsf/1.0/

**WSDL-S**, uses WSDL's extension mechanism to associate descriptions of element semantics to WSDL constructs. This includes in particular, the definition of precondition and effects of operations as well as the extension of typing of description elements beyond their XML schema and schema instance. This additional typing information would refer, e.g., to an OWL construct. WSDL-S by itself does not define an ontology or condition language but reuses existing work.

### Issues addressed

As shown above, there are a number of approaches to describe the semantics of Web services, typically grounding ontologies to elements of a WSDL description of a service. Based on the above approaches, a number of ontologies have been developed, typically for particular domains or functions. Independently of Web services, there are projects to develop generic ontologies such as the Psych project. Also, there are a number of tools for creating and managing ontologies.

### Open Issues

A number of issues has not been address sufficiently, though:

- There is no accepted programming model yet that allows implementers to refer to and use ontologies while implementing a Web service and managing its use through the life-cycle.
- There is not commonly accepted process yet to establish a common ontology or to merge ontologies. Neither top-down design approaches nor bottom-up folksonomies (common semantics emerges by the use of the same term for a concept in the tagging process of a participatory Web site) have found their right usage scope yet.
- It is still unclear whether ontologies can be created independently of a particular function that they are associated with. Can it be done at all?
- How do we use ontologies in the context of REST-type Web services that don't have a strong WSDL associated with them and primarily rely on the submitted document content structure to determine the behavior of the service.

# Quality of Service

Work group: core group + systems management group + Mike Papazoglou

### Issues of QoS and Monitoring

The quality of a Web service (QoS) is an important decision-making criterion for a client to use a Web service, an element of pricing for Web services, hence, an aspect to monitor for a service client and, in particular, for a service provider to implement the QoS. QoS is understood to refer to those aspects of a service that can vary without impacting the core function of the service, e.g., security provisions and performance. QoS aspects typically relate to the management of a Web service's life-cycle as well as the management of the system that implement the service at the desired quality. QoS is usually addressed in the context of service levels, or service level agreements (SLAs). A service level is a set of QoS parameters and a set of constraints over these parameters.

Typical issues to be addressed in this context include the following.

- How to implement a given service level? In the context of performance, this typically means how to manage the resources assigned to one or a set of Web services. This is typically the hardest issue.
- How to establish a service level? Can the service provider unilaterally define and maintain a service level, e.g., by implementing a particular security feature, or does a service level depend on client behavior and provider resources, hence, requiring agreement between provider and client. Another reason for using agreements is the establishment of penalties and rewards related to service level achievements.

- How to describe and execute the negotiation process for establishing QoS?
- What to query for a Web services at a given QoS? When should an individual client ask for a single service binding or operation at a given service level and when should an organization ask for an aggregate service level for a set of clients.
- How to manage different service levels (potentially from different providers) from a service consumer's point of view for multiple client applications?
- How to derive aggregate QoS properties in a service composition? How to use composition to yield aggregate QoS properties.
- How to deal with changing QoS requirements through the life-cycle of a binding?

### Issues addressed

Some issues have been addressed by the systems management and the Web services community:

- QoS properties can be associated with Web services by using the WS-Policy framework, in the case of a unilateral management of a service level by a provider, or using WS-Agreement for SLAs.
- Performance management is a traditional discipline of systems management. Depending on the implementation platform of a Web service, e.g., EJBs in an application server architecture, workload estimation and management techniques can be employed for the purposes of Web services.

### Open Issues

A number of other issues have been addressed only marginally:

- How to enable third party monitoring for Web services QoS?
- How to manage Web service QoS and capacity from a service consumer's point of view?
- How to aggregate QoS properties?

A large number of open issues at the intersection of Web services and system management remain to be addressed.

## Other Issues

The remaining set of issues that was mentioned in the initial list above was only discussed by the group on the side, not in a systematic way: Business Web Services, Web Services with external effects, Security, and Service Adaptation. Hence, no detailed results beyond the basic issues were addressed.

## Acknowledgements

This session summary contains contributions from all participants. In particular, it includes material from presentations of Michael Maximilien and Michael Klein on Web services semantics.