# A framework and ontology for Semantic Grid Services: An integrated view of WSMF and WSRF

Clement Jonquet

# A framework and ontology for Semantic Grid Services:

## An integrated view of WSMF and WSRF

Clement Jonquet – KMi and LIRMM

May 2005

This little draft paper describes some first ideas to extend the theoretical concepts of WSMF (and WSMO) to Grid service approach. WSMF is a framework for Semantic Web Services and WSMO is the corresponding ontology. Advances in Web/Grid services have been detailed in the WSRF specification. This draft paper proposes firstly a framework as an extension of WSMF, and secondly details modifications/extensions of WSMO that are required to fit with WSRF principles. This integration WSMF-WSRF can be saw has a framework and ontology for future specification of Semantic Grid Services (SGS).

# 1. Introduction and context

## 1.1. Web, Semantics and Services

### *1.1.1. Web Service*

The emergence of Service Oriented Architecture (SOA) and Service Oriented Computing (SOC) show the importance of the concept of service in the development of distributed systems. This is historically due to the work of the Open Group with DCE and the OMG with CORBA. Nowadays, the main way of implementing SOA (framework) is Web services[1].

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols [WSAR 2004].

Web services are software components that perform some function (capability). They are based on the three XML-like standard languages: i) WSDL (Web Services Description Language) to describe Web service's interface i.e. capabilities that may be invoked; ii) SOAP (Simple Object Access Protocol) to describe methods for accessing these components, i.e. message exchanges; iii) UDDI (Universal Description, Discovery and Integration) to publish a service and to identify a service provider in a service registry. Due to recent work of consortia and companies on Web services' orchestration and choreography, another set of languages has to be added to Web service definition in order to describe Web services compositions[2].

Web services (WS) allow to access distributed functionalities on a network in a standardised way. With WS, different applications can communicate (through HTTP and firewalls) with each other without knowing something special about their implementation (operating system or programming language) but only dealing with a standardised interface and exchanged expressed by XML documents. Thus, standardisation is one of the two main purposes of WS and allows the second one: interoperability. However, WS discovery, invocation, composition, interoperation is limited to be human interpretable by their lack of semantics. Existing technologies for WS only provide descriptions at the syntactic level; as no explicit semantic information is normally defined, automated comprehension of the service description is limited to cases where the provider and requester assume pre-agreed ontologies, protocols and shared knowledge about operations. As example, consider WS discovery and publication (with registry such as UDDI) which is typically human oriented and based upon yellow or white-page queries (i.e. metadata descriptions of service types, or information about the service providers).

### *1.1.2. Semantic Web*

Today's Web is geared for use by people, Human Agents (HA) who can interpret the content of the information because they have the necessary background knowledge, which they share with the creators of the given pages. Programs, Artificial Agents (AA) could process this information only in a ad hoc way. It explains the emergence of the Semantic Web [], as an addition of machine-interpretable/readable information to Web content in order to

---

[1] See www.opengroup.org/dce for DCE, see www.corba.org for CORBA, and www.w3.org/2002/ws for Web services.

[2] BPEL4WS (Business Process Execution Language for Web Services) or WSFL (Web Services Flow Language) or WSCL (Web Services Conversation Language) or XLang etc.

provide AA to access to heterogeneous and distributed information. In the Semantic Wed, documents are marked up not only with presentation details, but also with a separate representation of the meaning of their content. The semantic layer of is given by ontologies. Ontologies have been developed within the Knowledge Modelling research community [] in order to facilitate knowledge sharing and reuse. They provide greater expressiveness when modelling domain knowledge and can be used to communicate this knowledge between any types of agents. In addition, the Semantic Web provides the necessary techniques for reasoning about ontologies' concepts, as well as resolving and mapping between ontologies.

The current components of the Semantic Web framework are RDF [], RDF Schema (RDF-S) [] and the Web Ontology Language – OWL (previously DAML+OIL). These standards build up a rich set of constructs for describing the semantics of online information sources.

### *1.1.3. Semantic Web Services*

Naturally, the Semantic Web advances touched the WS community and the notion of Semantic Web Services (SWS) appears as semantically described WS. SWS use ontologies to constitute the knowledge-level model of the information describing and supporting the usage of the services. These ontologies enable automated understanding of their capabilities. More generally this semantic layer helps WS discovery, invocation, composition, or interoperation. For example, WS discovery should be based on the semantic match between a description of the service request, and a description of published service. The enrichment of business process languages such as BPEL4WS or BPML etc. by semantic layer should enhance interoperation between services.

Related work: [] proposes to characterize SWS by three dimensions: *usage activities*, *architecture* and *service ontology*. Usage activities define the functional requirements, the architecture defines the components needed for accomplishing these activities and finally, the service ontology establishes the link with domain knowledge.

[] details specifically how WS discovery and composition are facilitated with SWS. It details the WSMF aspect separating service description and capabilities description and explains how defining capabilities, refinements and actual input and output data using appropriate ontologies, service description exposes enough information to enable automatic discovery and composition.

[Other papers.]

Two main approaches are today available for SWS development: OWL-S [] and WSMF []. OWL-S is an agent-oriented approach to SWS, coming from DARPA with DAML-S and providing fundamentally an ontology for describing WS capabilities. WSMF aims at providing an appropriate conceptual model for developing and describing services and their composition, based on the principles of maximal decoupling and scalable mediation, originally motivated by Knowledge Modelling community with PSM and ontologies.

In this paper the interest on SWS will be limited to the WSMF approach and specifically to WSMO, the corresponding ontologies and IRS the platform for SWS usage.

## 1.2. Grid, Semantics, and Services

### *1.2.1. Grid*

The essence of the Grid concept is nicely reflected by its original metaphor: the delegation to the electricity network to offer us the service of providing us enough electric power as we need it when we need it even if we do not know where and how that power is generated. In the same sense, the Grid aims to enable "resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organisation" []. Actually, it was originally designed to be an environment with a large number of networked computer systems where computing and storage resources could be shared as needed and on demand.

Grid provides the protocols, services and software development kits needed to enable flexible, controlled resource sharing on a large scale. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. At the heart of Grid is the concept of virtual organization (VO) []. A virtual organization is a dynamic collection of individuals, institutions and resources bundled together in order to share resources as they tackle common goals.

Grid technologies have evolved from ad hoc solutions, and de facto standards based on the Globus Toolkit (GT), to Open Grid Services Architecture (OGSA) [] which adopts WS standards and extend services to all kind of resource (not only computing and storage). Today, the WS-Resource Framework (WSRF) defines uniform mechanisms for defining, inspecting, and managing stateful resources in Web/Grid services.

### *1.2.2. Grid Service*

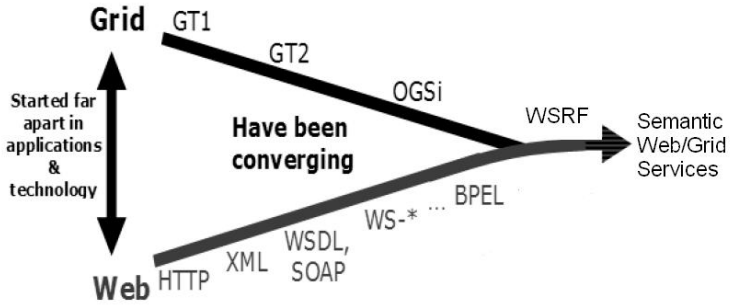
A Grid Service (GS) is a (potentially transient) Web service with specified interfaces and conventions: the interface address discovery, dynamic service creation, lifetime management, notification and manageability, while the conventions address naming and upgradeability.

GSs introduce explicit service creation and lifetime management. GSs introduce also the specification for managing state in service.

WSs have instances that are stateless and nontransient. In contrast, GSs can be either stateful or stateless, and can be either transient or non-transient.

### *1.2.3. Semantic Grid*

### *1.2.4. Semantic Grid Service*




## 1.3. An integrated view of WSMF and WSRF.

In this draft paper we try to show how to do this integration between SWS and GS approaches. As we will explain the main concept of the Grid service/WSRF approach is the externalization of stateful resource from Web services, which are considered stateless. WSRF defines the specification of WS-Resource as the association between stateful resources and stateless services. Therefore, we concretely propose a simple extension of WSMF by 2 key concepts:

i) Resources as another WSMF main element (like Ontologies, Web Services, Goals, and Mediators) and,

ii) Other types of mediation especially service–resource mediation.

The concrete extension/modification of WSMO are therefore: *Resource*, as a new top level element and *rw-mediator*, as a new kind of mediator embodying the WS-Resource.

Previous related work:

The WHAT and the WHY were briefly presented in 2 previous papers from the WSMO community:

> Towards a Semantic Grid Operating System, Moran M., Cabral L.S., Domingue J., Bussler C.; *Workshop on Network Centric Operating Systems*, March 2005, Brussels, Belgium.
>
> WSMO and Grid, Ioan Toma, Dumitru Roman and Kashif Iqbal, *WSMO Deliverable D25.1 v0.1*, November 2004.

The first paper, states that SWS technology can be extended to take Grid requirements into consideration. The authors firstly present the SWS infrastructure (ontology (WSMO), language (WSML) and environments (WSMX and IRSIII)) and then propose a conceptual framework for a Semantic Grid Service operating system (GRID OS).

The second one, states that Grid applications require both a conceptual model and a language to semantically described Grid services and Gird resources and that WSMO/WSML (with some restrictions/extensions) could provide these requirements. For example, current Grid languages, like GRAM (Grid Resource Allocation

Manager), are based on XML and XML-schema, thus inheriting all its drawbacks (semi structured data format, no formal semantics, no reasoning support, etc); here is where OGSA/WSRF could benefit from the conceptual model of WSMO and its associated language, WSML. This paper makes also a little overview of the Grid.

The rest of the paper is organised as follows: We firstly present the WSMF/WSMO/IRS approach of SWS. Then the WSRF approach of Web/Grid services. Then the integration of these two approaches.

# 2. The Web Service Modelling Framework (WSMF)

WSMF provides a model for describing the various aspects related to SWS. WSMF is the product of research on modelling of reusable knowledge components [].It is an extension of the UPML framework [] revised to integrate fully with WS and to support ecommerce.

WSMF is centred on two complementary principles: a strong de-coupling of the various components that realize an e-commerce application; and a strong mediation service enabling WS to communicate in a scalable manner.

Mediation is applied at several levels: mediation of data structures; mediation of business logics; mediation of message exchange protocols; and mediation of dynamic service invocation.

WSMF consists of four main elements: *ontologies* that provide the terminology and relationships used by other elements; *goal* repositories that define the problems that should be solved by Web services (users' objectives are expressed by goal); *Web services* descriptions that define various aspects of a Web service; and *mediators* which bypass interoperability problems.

WSMF insists on the separation between capabilities and services.

Three related initiatives associated with WSMF have recently begun. These are WSMO [] which will develop an ontology for WSMF, WSML [] which will develop a formal language for representing the WSMO based descriptions and WSMX [] which will develop a reference implementation. WSMF is also the product of experimental research with the IRS and the latest version of this SWS platform allows publishing and using of WSMO compatible SWS.

*2.1.1. The Web Service Modelling Ontology (WSMO)*

WSMO is an implementation project of WSMF. It is a formal ontology for describing the various aspects related to SWS following WSMF. The main components of WSMO are the four main elements of WSMF: Goals, Web Services, Ontologies and Mediators.

- Goals represent the types of objectives which users would like to achieve via a WS. Goals are basically an association between a capability of the services the user would like to have and the interface the service the user would like to have and interact with. Goals describe the state of the desired information space and the desired state of the world after the execution of a given WS.
- WS descriptions represent the functional behaviour of an existing deployed WS. A WS is related to one or more capabilities. The description also outlines how WS communicate (choreography) and how they are composed (orchestration).
- Ontologies provide the basic glue for semantic interoperability and are used by the three other WSMO components.
- Mediators, which specify mapping mechanisms In WSMO architecture, all interoperability aspects are concentrated in mediators. The goal, WS and ontology components are linked by four types of mediators as follows:
    - oo-mediators enable components (WS, capability, goal) to import ontologies when steps for aligning, merging, and transforming imported ontologies are needed, oo-mediators may also specify an ontology mapping between two ontologies,
    - ww-mediators link web services to WS, a WS can use ww-mediators to deal with process and protocol mediation,
    - wg-mediators connect WS with goals,
    - gg-mediators link different goals (a goal may be defined by reusing one or several already-existing goals).

Mediators have source components which defines the entity that is the source of the mediator. And target component defines the entity that is the target of the mediator. The incorporation of four classes of mediators in WSMO facilitates the clean separation of different interoperability mechanisms. For examples:

WSMO is an ontology, it aims is to describe SWS. The purpose of this paper is to extend this ontology to describe also stateful resources allowing WSMO to describe SGS.

*2.1.2. The Internet Reasoning Service (IRS-III)*

IRS III is a knowledge-based framework and implemented infrastructure for SWS, which evolved from research on reusable knowledge components []. The IRS project has the overall aim of supporting the automated or semi-automated construction of semantically enhanced systems over the internet. IRS-I supported the creation of knowledge intensive systems structured according to the UPML framework (UPML distinguishes with ontologies *domain models*, *task models*, *Problem Solving Methods* (PSMs), *bridges*, which was the first concept of the today's WSMF main elements). IRS-II integrated the UPML framework with WS technologies.

Then, IRS-III supports the creation of SWS according to the WSMO ontology. IRS-III has four main classes of features which distinguish it from other work on SWS.

- Firstly, it supports one-click publishing of 'standard' programming code. In other words, it automatically transforms programming code (currently we support Java and Lisp environments) into a WS, by automatically creating the appropriate wrapper. Hence, it is very easy to make existing standalone software available on the net, as WS.
- Secondly, by extending the WSMO goal and web service concepts users of IRS-III directly invoke WS via goals i.e. IRS-III supports capability-driven service execution.
- Thirdly, IRS-III is programmable. IRS-III users can substitute their own SWS for some of the main IRS-III components.
- Finally, IRS-III services are WS compatible – standard WS can be trivially published through the IRS-III and any IRS-III service automatically appears as a standard WS to other WS infrastructures.

Using IRS, a publisher agent should specify all the elements (WS, of course, but also ontology, goal and mediators) related to its service and which allow its service to be semantically connected to others element of the platform.

# 3. The Web Service Resource Framework (WSRF) approach

The motivation of WSRF comes from the fact that even if today's WSs successfully implement applications with state, it needs to be standardized to enhance interoperability, perhaps the major objective of WSs. WSRF defines these standards with respect to current Web services tecnhologies.

The purpose of WSRF is to define a generic and open framework for modelling and accessing stateful resources using Web/Grid services. Before WSRF, programmers may infer data identifier from the messages declared in the service's interface. WSRF standardizes the relationship between the stateful resources and the WS.

The origin of WSRF comes partly from OGSI/OGSA specification of Grid services. The 2 major advances provided by Grid services on Web services are the management of state, and the transient aspect of service. Both comes from the fact that OGSA distinguishes between service factory and service instance. A service instance has is own state and lifetime.

WSRF identifies 3 types of service:

1. A *stateless service* implements message exchanges with no access or use of information not contained in the input message. These are represented as pure functions. The advantage of easy composition of purely functional services comes at the cost that they can hardly (for example with streams) represent state.
2. A *conversational service* implements a series of operations such that the result of one operation depends on a prior operation and/or prepares for a subsequent operation. The behaviour of a given operation is based on processing preceding messages in the logical sequence. These are the most generic stateful services. Hard to be realized within a distributed and asynchronous context, heavy to be supported and maintained (HTTP sessions, and cookies, because HTTP doesn't have state)
3. A *service that acts upon stateful resources* provides access to, or manipulates a set of logical stateful resources (documents) based on messages it sends and receives.

Remark:

**Services are said to be stateless** if they delegate responsibility for the management of the state to another component. Statelessness is desirable because it can enhance reliability and scalability. A stateless Web service can be restarted following failure without concern for its history of prior interactions, and new copies (instances) of a stateless Web service can be created (and subsequently destroyed)..

**Stateful resources** are elements with state, including physical entities (e.g. database, file system, servers) and logical constructs (e.g. business agreements, contracts) that are persistent and evolve because of service interactions. In WSRF a stateful resource is defined to: i) have a specific set of state data expressible as an XML document; ii) have a well-defined lifecycle; and iii) be known to, and acted upon, by one or more Web services.

WSRF explicitly addresses the third type. These services are modelled by an association between 2 entities: a stateless Web services, that do not have state, and stateful resources that do have state. WSRF calls the resulting association a WS-Resources and introduces an "implied resource pattern" to formalize this relationship/association. A WS-Resource is accessible through its Web service interface and messages to a Web service include a component that identifies a stateful (identifier) resource to be used in the execution of the message exchange. A WS-Resource lifecycle starts with *creation* through the use of a WS-Resource factory, the assignment and use of the stateful resource identifier, and the *destruction* of a WS-Resource.

WSRF is a set of six specifications that define WS-Resource:

- WS-ResourceLifetime, which allows a user to specify the period during which WS-Resource definition is valid (mechanisms for WS-Resource destruction (immediately or scheduled)).
- WS-ResourceProperties which defines how a WS-Resource can be queried and changed using Web service technologies; it allows clients to build applications that read and update data. The WS-Resource properties document acts as a view on, or projection of, the actual state of the WS-Resource.
- WS-Notification, WS-RenewableReferences, WS-ServiceGroup, WS-BaseFaults.

We will be interested here specifically in the 2 first ones.

Both stateful resource and stateless service can be member of several WS-Resource. However, a WS-Resource is unique. (identified by a WS-Addressing EndpointReference after denoted *endpoint*). Requestor's messages must be sent to a Web service referred by an endpoint.

The ideas of factory and instanciation of Grid services from OGSI/OGSA which allows the management of state and the transient aspect of service is related in WSRF at the resource level.An instance of a stateful resource may be created via a Web service referred s a *stateful resource factory* or *WS-Resource factory* (because creating the WS-Resource implies creating an instance (or choosing an already existing one) of a stateful instance). That means that a stateful resource factory produces stateful resource instance which are implied in the WS-resource association.

Then, when a client request a Web service which should act upon resource, a WS-Resource is created and a endpoint is returned to the client. Even if the client does not directly act upon the stateful resource he is allocated to, he knows its identitier. This identifier is coupled to messages sent next to the Web service as it can act upon it.

The interaction steps between 2 clients and a Web service could be:

1. Client1→WS1: I would like to use a service that acts upon stateful resource.
2. Client2→WS1: I would like to use a service that acts upon stateful resource.
3. WS1→Client1: Ok, the endpoint of the WS-Resource dedicated to you is (WS1, SR1).
4. WS1→Client2: Ok, the endpoint of the WS-Resource dedicated to you is (WS1, SR2).
5. Client1→WS1: I would like to apply function f of WS1 with resource SR1.
6. Client2→WS1: I would like to apply function g of WS1 with resource SR2.

The fundamental lessons learned from the WSRF may be:

1. The required stateful and persistent nature of services. This is clearly coupled with the interactive (conversational) nature of services but which is not treated here.

2. The required flexible stateless behaviour of services, as it is hard to compose dynamically state dependent components.

[Section: Analogy between WSRF and the agent representation and communication model, STROBE, which consider service as programs and resource as environments of execution of these programs.]

# 4. An integrated view of WSMF and WSRF

Then, to extend WSMF with WSRF principles some questions raises:

1. Is WSMF Web Service element stateless?
2. How to represent stateful resource in WSMF?
3. How to realise the association between a stateful resource and a stateless service (WS-Resource)?
4. How to manage lifetime and properties of these associations?
5. How to realise these associations statically (when the WS is deployed) but also dynamically (at the message exchanges)?
6. How to realise the instantiation of WS-Resource; i.e. how to make the distinction between a stateful resource factory and a stateful resource instance, as it is important in Grid services and now in WSRF?

## 4.1. Extension of WSMF

First, to realise WSRF requirements, we need to consider the Web service entity of WSMF as a stateless entity. The 3 other elements of WSMF i.e. mediators, ontologies and goals do not change.

(to be detailed)

(cf. the notion of capability, pre/post conditions and effects etc.)

Then, we need to consider a new main element to represent stateful resources of WSRF. We will denote this WSMF element *WSMF-resource* in order to be not confusing between stateful resources, WS-Resources etc.

As mediation is at the centre of interaction between WSMF elements, by adding a new element in WSMF, we inevitably add new kinds of mediation. The new kinds of mediation are:

- Mediation between WSMF-resources,
- Mediation between WSMF-resource and Web service.

Note that the integration of this new element goes in the sense of WSMF by enhancing the decoupling of the various components (e.g. separating elements with state and elements without) of a service oriented application, and enhancing the mediation aspects between these components.

### *4.1.1. WSMF-Resource – WSMF-resource mediation:*

(to be thought)

(a resource can be defined by other resources (composition) or be an aggregation/composition of resources etc)

### *4.1.2. WSMF-Resource – Web service mediation:*

This type of mediation is particularly important because it realises the association between a Web service and a WSMF-resource. Therefore, the mediator between a WSMF-resource and a Web service embodies the WS-Resource. It has all the properties defined in WS-Resource specification especially the management of lifetime and properties.

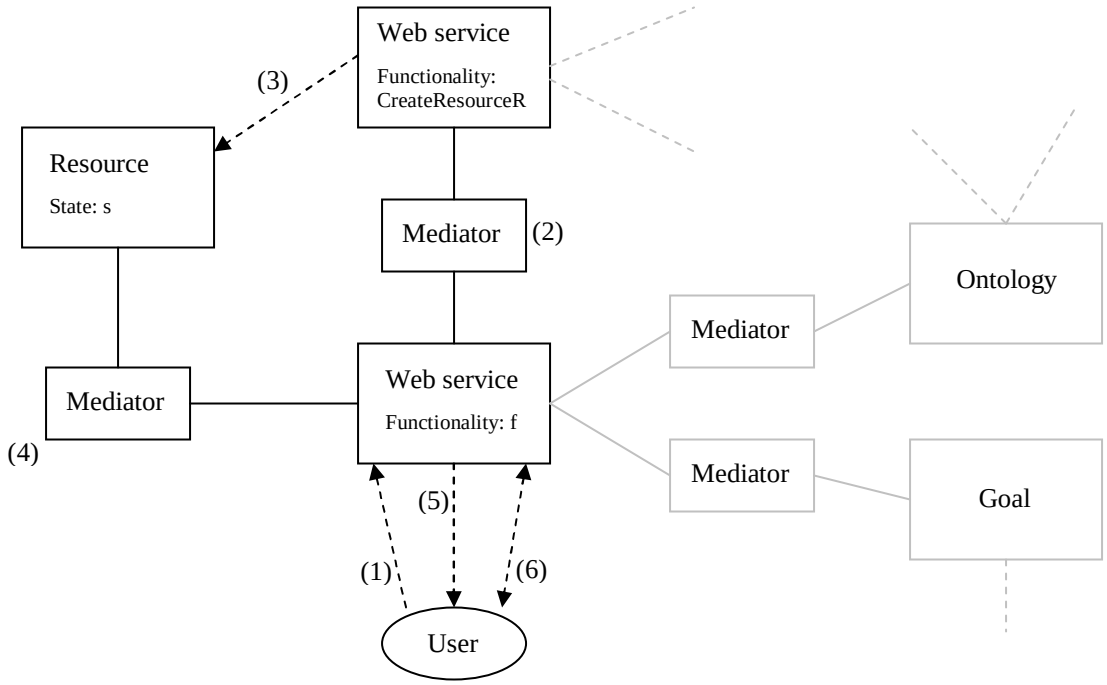
Each WSMF-Resource has the same properties that stateful resources described in WSRF (a specific set of state data, a well-defined lifecycle, it is acted upon, by one or more Web services).

In WSMF, the role of the WS-Resource factory of WSRF is done by an element of type Web service. We don't need to consider WS-Resource factory as a WSRF main element because WS-Resource factory are services that produce WS-Resource. The important element in the framework should be the stateful resource. Bringing a WS-Resource into existence consists of creating a new stateful resource (or choosing a new one), assigning this stateful resource an identifier, and creating the association between this stateful resource and its associated Web service.

Thus, a typical scenario could be (illustrated by the figure):

1. A client "connects" to a Web service according to the goals element (same step as before in WSMF).

2. The *requested Web service* interacts (via mediator) with another Web service, the *WS-Resource factory Web service*, to have the allocation of a WSMF-resource and thus the creation of a WS-Ressource (via the creation of a mediator).
3. The WS-Resource factory Web service produces a new stateful resource (or chooses an existing one) and allocates it, by creating a mediator, to the requested Web service. Finally, it sends the endpoint (which contains the identifier of the WSMF-resource created/chosen) of the WS-Resource created to the requested Web service.
4. The WS-Resource is embodied by the mediator between the requested Web Service and the WSMF-resource returned by the second Web service.
5. The endpoint of the WS-Resource created is sent back to the user by the requested Web service.
6. The user requests the Web service by specifying the WSMF-resource it wants to act upon. The "acts" upon the WSMF-resource done by the requested Web service are managed by the mediator which embodies the WS-Resource.




In this scenario, the WS-Resource is created dynamically, that means that the requested Web service asks the WS-Resource factory Web service for the creation of a WSMF-resource when it is itself requested by a client. It could of course be done statically, when the requested Web service was deployed.

(to be detailed: the role of the new mediator)

## 4.2. Extension/modification of WSMO

The previous section extension of WSMF could be expressed in the WSMF corresponding ontology i.e. WSMO.

Extension of WSMO: We need to add:

- A new top level element: **resource** to represent WSMF-resource (thus stateful resource !!),
- 2 new kinds of mediators: **rw-mediator** and **rr-mediator**.

Then, a WS-Resource is represented in WSMO by a rw-mediator.

Modification of WSMO: The Web Service elements of WSMO have to become stateless.

In WSMO, services are basically interfaces on capabilities. A capability defines the service by means of its functionality. This capability is expressed by:

- preconditions, which specify the information space of the service before its execution,

- assumptions, which describe the state of the world before the execution of the service,
- postconditions, which describe the information space of the service after the execution of the service,
- effects, which describe the state of the world after the execution of the service.

All of them applying on shared variables:

If `?v1,...,?vn` are the shared variables defined in a capability, and `pre(?v1,...,?vn)`, `ass(?v1,...,?vn)`, `post(?v1,...,?vn)` and `eff(?v1,...,?vn)`, are used to denote the formulae defined by the preconditions, assumptions, postconditions, and effects respectively, then the following holds:

**forAll** `?v1,...,?vn ( pre(?v1,...,?vn) and ass(?v1,...,?vn)` **implies** `post(?v1,...,?vn) and eff(?v1,...,?vn) ).`

This formula simply reflects the traditional situations of applying procedure in an environment: A procedure call consumes arguments, is evaluated in an environment, produces changes on the environment, and returns some values. No matter where the environment is "situated" (here: inside or outside the Web service). [Link to the STROBE approach]

In our extension of WSMO, the previous formula may always hold. However assumptions and effects, have not to be part of the capability anymore. They describe the old and new states of the resource (part of the world) affected by the application of the Web service. Thus they have to be defined in the resource elements. Thus, the sharedVariables element of a capability should now be accessible both by the resource (which provides assumptions and effects) and the Web service (which provides preconditions and postconditions). This is achieved by defining this element in the WS-Resource, that is to say in the rw-mediator.

[An interface describes how the functionality of the service can be achieved (i.e. how the capability of a service can be fulfilled) by providing a twofold view on the operational competence of the service: choreography decomposes a capability in terms of interaction with the service. Orchestration decomposes a capability in terms of functionality required from other services.

(to be thought) ask John the paper on orchestration and choreography… how this extension of WSMO changes choreography and orchestration]

Resource definition:

```
Class resource sub-Class wsmoTopLevelElement
        hasNonFunctionalProperties type nonFunctionalProperties
        importsOntology type ontology
        usesMediator type {oo-mediator, rw-mediator, rr-mediator}
        hasSetData type setData
        hasAssumption type axiom
        hasEffect type axiom
```

**Non-Functional Properties**

The non-functional properties recommended are: (to be done)

**Importing Ontology**

Used to import ontologies as long as no conflicts need to be resolved. Indeed, a resource needs to be described with a semantic and then is related to some ontology.

**Using Mediator**

A resource can import ontologies using ontology mediators (ooMediator) when steps for aligning, merging, and transforming imported ontologies are needed. A resource can be defined by relations between other existing resources (composition, aggregation, and specialisation). This is achieved by using resource mediators (rrMediator). A resource can be implied in one or more WS-Resource using rw-mediators.

**Set of Data**

A resource has a specific set of state data expressible as an XML document.

**Assumption**

Assumptions describe the state of the world before the execution of the service.

**Effect**

Effects describe the state of the world after the execution of the service.

rw-mediator and rw-mediator definition:

```
Class rw-mediator sub-Class mediator
        usesMediator type oo-mediator
        hasSource type {resource, service}
        hasTarget type {resource, service}
        hasSharedVariables type sharedVariables
        hasLifetime type lifetime
        hasProperties type properties

Class rr-mediator sub-Class mediator
        usesMediator type oo-mediator
        hasSource type {resource, service}
        hasTarget type {resource, service}
```

**Using Mediator**

rw-mediator uses a set of ooMediators in order to map between different vocabularies used in the description of resource and service and align different heterogeneous ontologies. Idem for the rr-mediator.

**Source**

The source components define entities that are the sources of the mediator.

**Target**

The target component defines the entity that is the target of the mediator.

**Shared Variables**

Shared Variables represent the variables that are shared between of preconditions and postconditions, given by the Web service, and assumptions and effects, given by the resource. They are all quantified variables in the formula that concatenates assumptions, preconditions, postconditions, and effects.

**Lifetime**

A rw-mediator has a date for its scheduled destruction. At any time this date can be changed by the requestor (via the Web service implied in the WS-Resource) to now (immediate destruction) or to another date (up-date destruction). Lifetime aspects need to be detailed according to WS-ResourceLifetime specification.

**Properties**

A rw-mediator has a set of properties according to WS-ResourceProperties specification. In WSRF, the declaration of the WS-Resource's properties represents a projection of or a view on the WS-Resource's state. This projection is defined in terms of a resource properties document.

[This resource properties document serves to define a basis for access to the resource properties through the Web service interface - It corresponds to the means to declare resource properties as part of a Web service description and it defines the message exchanges for querying and updating resource property values.– to be done cf WSRF specification].

# 5. References

Sections on SWS, WSMF, IRS, and WSRF are inspired from:

L. Cabral, J. Domingue, E. Motta, T. Payne and F. Hakimpour (2004). Approaches to Semantic Web Services: An Overview and Comparisons. In proceedings of the First European Semantic Web Symposium (ESWS2004); 10-12 May 2004, Heraklion, Crete, Greece.

R. Lara, H. Lausen, S. Arroyo, J. de Bruijn, D. Fensel: Semantic Web Services: description requirements and current technologies, International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003), Pittsburgh, PA, September 30, 2003

J. Domingue, L. Cabral, F. Hakimpour, D. Sell and E. Motta (2004). IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, CEUR Workshop Proceedings, ISSN 1613-0073

D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications, 1 (2002) 113-137

http://www-128.ibm.com/developerworks/library/ws-resource/ws-modelingresources.html