Demo of IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services

John Domingue, Liliana Cabral, Farshad Hakimpour,
Denilson Sell, and Enrico Motta

Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes, MK7 6AA, UK
{j.b.domingue, l.s.cabral, f.hakimpour,
d.sell, e.motta}@open.ac.uk
http://kmi.open.ac.uk/

1. Introduction

Web services promise to turn the web of static documents into a vast library of interoperable running computer programs and as such have attracted considerable interest, both from industry and academia. Existing web service technologies, however, are based on a manual approach to their creation, maintenance and management. At the centre of the conceptual architecture is a registry which stores descriptions of published web services. Clients query the registry to obtain relevant details and then interact directly with the deployed service. The descriptions, represented in XML based languages such as WSDL and UDDI, mostly focus on the specification of the input and output data types and the access details. These specifications are obviously not powerful enough to support automatic discovery, mediation and composition of web services. A software agent cannot find out what a web service actually does, by reasoning about a WSDL specification.

The above issues are being addressed by ongoing work in the area of semantic web services [OWL-S, 2002; Fensel and Bussler, 2002]. The overall approach is that by augmenting web services with rich formal descriptions of their competence many aspects of their management will become automatic. A recent initiative based on the WSMF framework [Fensel and Bussler, 2002] is the *Web Service Modeling Ontology* (WSMO) [WSMOa, 04]. WSMO is a formal ontology for describing semantic web services based on the four components of WSMF: Goals, Web Services, Mediators, and Ontologies.

Goals represent the types of objectives which users would like to achieve via a web service. The WSMO definition of a goal describes the state of the desired information space and the desired state of the world after the execution of a given web service. A goal can import existing concepts and relations defined elsewhere either by extending or simply reusing them as appropriate.

Web service descriptions represent the functional behaviour of an existing deployed web service. The description also outlines how web services communicate (choreography) and how they are composed (orchestration).

One of the key features of WSMO is mediators which are first class representations of the mappings required to connect WSMO components. WSMO distinguishes between different classes of mediators by the mapping source and target. The final component of WSMO are ontologies which provide the basic glue for semantic interoperability.

Our demonstration would be based on IRS-III (Internet Reasoning Service) a framework and implemented infrastructure which supports the creation of semantic web services according to the WSMO ontology. IRS-III has four main classes of features which distinguish it from other work on semantic web services. 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 web service, by automatically creating the appropriate wrapper. Hence, it is very easy to make existing standalone software available on the net, as web services. Secondly, by extending the WSMO goal and web service concepts users of IRS-III directly invoke web services via goals i.e. IRS-III supports *capability-driven* service execution. Thirdly, IRS-III is programmable. IRS-III users can substitute their own semantic web services for some of the main IRS-III components, for example, how web services are selected from a goal request or how complex services are executed. Finally, IRS-III services are web service compatible – standard web services can be trivially published through the IRS-III and any IRS-III service automatically appears as a standard web service to other web service infrastructures.

2. IRS-III

IRS-III builds upon the previous version, IRS-II [Motta et. al, 2003] incorporating and extending the WSMO ontology within the IRS-III server, browser and API.

2.1 The IRS-III Architecture

The IRS-III architecture is composed of the main following components: the IRS-III Server, Publisher and Client, which communicate through a SOAP-based protocol, as shown in figure 1.

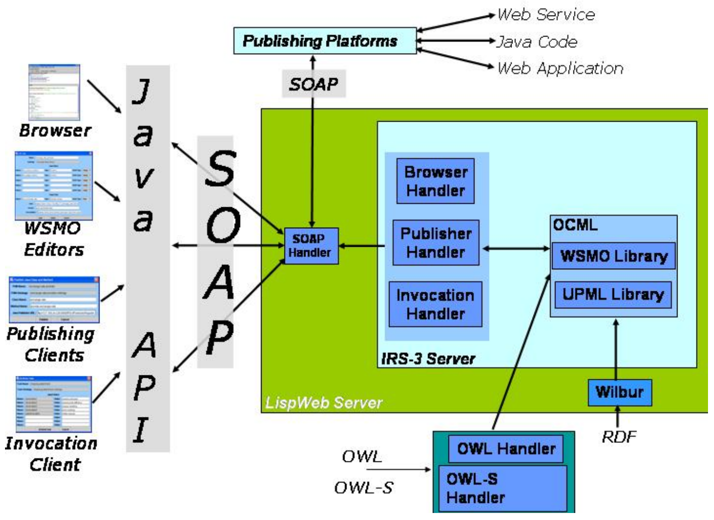


Fig. 1. The IRS-III Server Architecture

The IRS-III Server is based on an HTTP server written in lisp which has been extended with a SOAP handler. Separate modules handle soap based requests from the browser, the publishing platforms and the invocation client. Messages result in a combination of queries to or changes within the entities stored in the WSMO library.

Publishing with IRS-III entails associating a deployed web service with a WSMO web service description. Within WSMO a web service is associated with an interface which contains an orchestration and choreography. Orchestration specifies the control and dataflow of a web service which invokes other web services (a composite web service). Choreography specifies how to communicate with a web service. When a web service is published in IRS-III all of the information necessary to call the service, the host, port and path are stored within the choreography associated with the web service. Additionally, updates are made to the appropriate publishing platform. IRS-III contains publishing platforms to support the publishing of standalone Java and Lisp code and of web services. Web applications accessible as HTTP GET requests are handled internally by the IRS-III server.

IRS-III was designed for ease of use, in fact a key feature of IRS-III is that web service invocation is capability driven. The IRS-III Client supports this by providing a goal-centric invocation mechanism. An IRS-III user simply asks for a goal to be solved and the IRS-III broker locates an appropriate web service semantic description and then invokes the underlying deployed web service.

2.2. The IRS-III Ontology

The IRS-III ontology is fundamentally an implementation of WSMO standard D2v02 [WSMOa, 2004] with some additional attributes. A web service invocation results in instances of the appropriate classes being created.

The differences are mainly derived from the fact that in IRS-III we aim to support capability driven web service invocation. To achieve this we require that goals and web services have input and output roles. In addition to the semantic type we also store the soap binding for input and output roles.

Consequently a goal in IRS-III has the following extra slots `has-input-role`, `has-output-role`, `has-input-role-soap-binding` and `has-output-role-soap-binding`.

Goals are linked to web services via mediators. More specifically, the mediators found in the `used-mediator` slot of a web service's capability. If a mediator associated with a capability has a goal as a source, then the associated web service is considered to be linked to the goal. Note that according to WSMO standard v 0.2 a goal can only fill the target slot of a wg-mediator, the source should be a web service or ww-mediator. In IRS-III we have therefore created a new type of mediator, a *gw-mediator* which transforms the input values associated with a specific goal invocation into a format acceptable for a target web service. Complementary wg-mediators are used to map the result of the web service invocation back to the goal output type.

Web services which are linked to goals 'inherit' the goal's input and output roles. This means that input role definitions within a web service are used to either add extra input roles or to change an input role type.

When a goal is invoked the IRS broker creates a set of possible contender web services using the gw-mediators. A specific web service is then selected using an applicability function within the assumption slot of the web service's associated capability. As mentioned earlier gw and wg mediators are used to transform between the goal and web service input and output types during invocation.

In WSMO standard v 0.2 the mediation service slot of a mediator may point to a goal that declaratively describes the mapping. Goals in a mediation service context play a slightly different role in IRS-III. Rather than describing a mapping goals are considered to have associated web services and are therefore simply invoked.

2.3. The IRS-III Java API

The IRS-III Java API enables developers to use IRS-III to create semantic web service based applications and to integrate IRS-III with other platforms. The API models concepts as in [WSMOa, 2004], which can be consumed by the IRS-III execution environment and reasoner. The API is also used within the browser and editor described in the next section. The API contains four packages representing WSMO ontologies, goals, web services and mediators. These data models are sent to the IRS-III server via SOAP messages when semantic web services are edited, published or invoked. Within the server the data is instantiated within WSMO based ontologies.

2.4. The IRS-III Browser

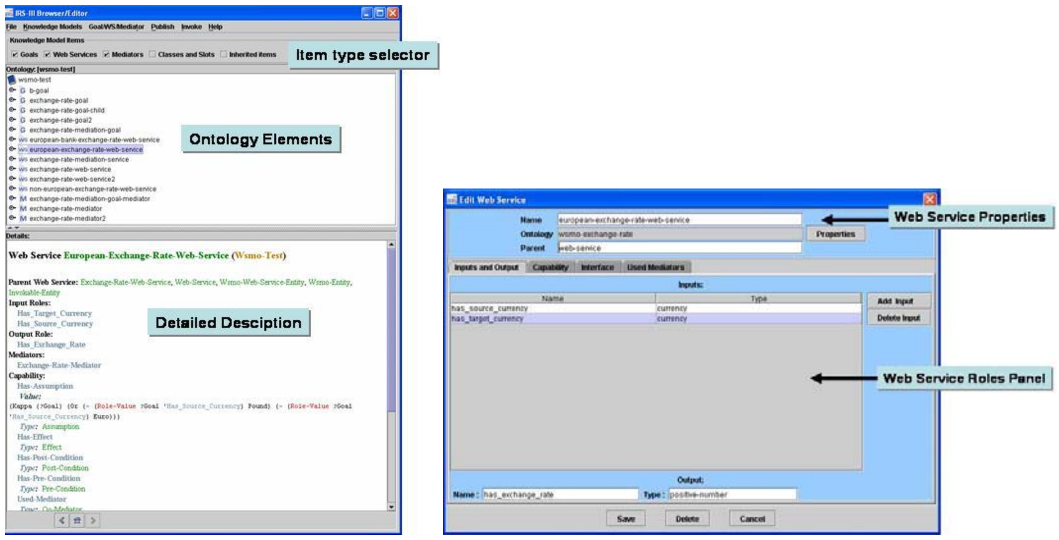
The IRS-III browser provides an easy to use interface to support the creation of WSMO classes, to publish web services against the WSMO descriptions and then to invoke the services. The basic browser is shown on the left of figure 2. The main elements of the browser are a toolbar which controls which types of elements are visible, a window which shows the elements of the selected ontology and a window which displays detailed descriptions of selected items. Icons are used to differentiate between goals, web services and mediators. The item type selector allows users to toggle the display of goals, web services, mediators and other classes. The user can also use the selector to include items which are inherited from ancestor ontologies.

Selecting an item in the elements window results in a detailed description appearing in the lower window. Here colour coding is used to differentiate between ontologies, classes, relations, instances and functions. Double clicking on an item results in its detailed description appearing.

One design feature behind the IRS-III browser was to ease some of the complexities of WSMO by clustering. Within the browser the sub-classes of the main WSMO classes goal, web service and mediator are visible. The 'second tier' WSMO classes interface, capability, orchestration and choreography are not displayed. Instead these classes are shown as part of the detailed description of their associated main classes. In figure 2 we can see the description of the capability class associated with `european-exchange-rate-web-service`.

The browser also allows new items to be defined or edited. Again for reasons of ease of use we cluster components together using panels. The right of figure 2 shows the dialog window for defining web services. The top part of the panel shows the generic web service properties. Selecting the properties button here brings up a dialog for defining the non functional properties. The other properties associated with a web service are contained in four panels: Inputs and Output, Capability, Interface and Used Mediators. The input and output panel is shown in figure 2. The 'Add Input' and

‘Delete Input’ buttons can be used to add and delete input roles. The browser also contains an interface to a simple goal based discovery mechanism.



✓ Goals ✓ Web Services ✓ Mediators ☐ Classes and Slots ☐ Inherited items

Ontology: [wsmo-test]

- wsmo-test
- G b-goal
- G exchange-rate-goal
- G exchange-rate-goal-child
- G exchange-rate-goal2
- G exchange-rate-mediation-goal
- WS european-bank-exchange-rate-web-service
- WS european-exchange-rate-web-service
- WS exchange-rate-mediation-service
- WS exchange-rate-web-service
- WS exchange-rate-web-service2
- WS non-european-exchange-rate-web-service
- M exchange-rate-mediation-goal-mediator
- M exchange-rate-mediator
- M exchange-rate-mediator2

Web Service European-Exchange-Rate-Web-Service (Wsmo-Test)

Parent Web Service: Exchange-Rate-Web-Service, Web-Service, Wsmo-Web-Service-Entity, Wsmo-Entity, Invokable-Entity
Input Roles:
Has_Target_Currency
Has_Source_Currency
Output Role:
Has_Exchange_Rate
Mediators:
Exchange-Rate-Mediator
Capability:
Has-Assumption
Value:
```
(Kappa (?Goal) (Or (= (Role-Value ?Goal 'Has_Source_Currency) Pound) (= (Role-Value ?Goal
'Has_Source_Currency) Euro)))
```
Type: Assumption
Has-Effect
Type: Effect
Has-Post-Condition
Type: Post-Condition
Has-Pre-Condition
Type: Pre-Condition
Used-Mediator
Type: Oo-Mediator

Name: european-exchange-rate-web-service
Ontology: wsmo-exchange-rate
Parent: web-service

Inputs:

Name	Type
has_source_currency	currency
has_target_currency	currency

Output:
Name: has_exchange_rate Type: positive-number

Fig. 2. A screen snapshot showing the IRS-III browser (left) and IRS-III dialog for defining web services (right).

3. Summary

Our aim in designing IRS-III was to support the easy creation and use of semantic web services based on WSMO descriptions. To this effect IRS-III facilitates *one click publishing* – by filling in a simple form a piece of standalone Java or Lisp code, a web application accessible through a browser or a web service can be attached to a semantic description, and *capability based invocation* – web services can be invoked by asking for a WSMO goal to be solved.

IRS-III was successfully used at the WSMO tutorials to be held in conjunction with the AIMSA conference at the beginning of September. Within the demo we intend to show an application based on the WSMO virtual travel agency [WSMO, 2004b] and a simple example based on currency exchange. The API and browser can be downloaded from the IRS web site kmi.open.ac.uk/projects/irs/.

Acknowledgements

This work is supported by DIP (Data, Information and Process Integration with Semantic Web Services) (EU FP6 - 507483) and AKT (Advanced Knowledge Technologies) (UK EPSRC GR/N15764/01) projects.

References

[Fensel and Bussler, 2002] The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1(2): 113-137.

[Motta et. al, 2003] IRS-II: A Framework and Infrastructure for Semantic Web Services. 2nd International Semantic Web Conference, Sundial Resort, Sanibel Island, Florida., USA.

[Motta, 1998] An Overview of the OCML Modelling Language, The 8th Workshop on Knowledge Engineering Methods and Languages (KEML '98).

[OWL-S, 2002] Web Service Description for the Semantic Web. In the Proceedings of The First Int’l. Semantic Web Conf. (ISWC), Sardinia (Italy) (2002).

[WSMO, 2004a] Web Service Modeling Ontology – Standard, http://www.wsmo.org/2004/d2/

[WSMO, 2004b] WSMO Use Case Modeling and Testing, http://www.wsmo.org/2004/d3/d3.2/v0.1/