IRS III: a platform and infrastructure for creating WSMO based semantic web services

Conference or Workshop Item

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/

Abstract. 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 and IRS-II integrated the UPML framework with web service technologies. In this paper we describe IRS-III. Within IRS-III we have now incorporated and extended the WSMO ontology. Our extensions to WSMO include the addition of input and output roles to goals and web services and a new type of mediator. As well as summarizing our additions to WSMO we outline the architecture of IRS-III and the associated interfaces.

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. For example, IDC [Muse, 2003] predicts that web services will drive software, services and hardware sales of $21 billion in the U.S. by 2007 and will reach $27 billion in 2010. Current web service technologies are, however, relatively inflexible and ongoing research is investigating how semantic web technology can alleviate this.

Existing web service technologies 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 description languages such as WSDL [WSDL, 2001] and UDDI [UDDI, 2003], 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. Analogously the same agent cannot locate the appropriate service in UDDI registry, given a specification of a target functionality. As a result, existing web service infrastructures by and large support a manual approach to web service management: only manual discovery can be supported and only 'static', manually configured web applications are possible.

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. Specifically, web service location, composition and mediation can become dynamic, with software agents able to reason about the functionalities provided by different web services, to locate the best ones for solving a particular problem and to automatically compose the relevant web services to build applications dynamically.

Recently a number of initiatives based on the WSMF framework [Fensel and Bussler, 2002] have started. WSMF is an extension of the UPML framework [Fensel and Motta, 2001] revised to integrate fully with web services and to support ecommerce. WSMF is centered on two complementary principles: a strong de-coupling of the various components that realize an ecommerce application; and a strong mediation service enabling web services 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. Three related initiatives associated with WSMF have recently begun. These are WSMO [WSMO, 2004] which will develop an ontology for WSMF, WSML [WSML, 2004] which will develop a formal language for representing the WSMO based descriptions and WSMX [WSMX, 2004] which will develop a reference implementation.

In this paper we describe 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.

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.

The paper is organized as follows: in the following section we give a brief overview of WSMO. We then describe IRS-III outlining the architecture, the IRS-III ontology, the Java API, and the Browser. We then illustrate IRS-III in use through a simple exchange rate example. The final section of the paper contains a summary.

2 Overview of WSMO

The *Web Service Modeling Ontology* (WSMO) [WSMO, 04] is a formal ontology for describing the various aspects related to Semantic Web Services following WSMF. As can be expected 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 web service. The WSMO definition of 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).

Ontologies provide the basic glue for semantic interoperability and are used by the three other WSMO components.

Finally, we have mediators, which specify mapping mechanisms. One of the main characterizing features of WSMO is that the first three components, goals, ontologies and web services are linked by mediators, in particular, WSMO provide four kinds of mediators:

- OO-mediators enable components to import ontologies,
- WW-mediators link web services to web services,
- WG-mediators connect web services with goals, and
- GG-mediators link different goals.

The incorporation of four classes of mediators in WSMO facilitates the clean separation of different interoperability mechanisms. For example, an oo-mediator may specify an ontology mapping between two ontologies whereas a ww-mediator may specify a process transformation between two web services.

3 Overview of IRS-III

IRS-III is a framework and implemented infrastructure which allows publication, composition, execution of different and heterogeneous web services, building upon the previous version, IRS-II [Motta et. al, 2003]. The specific extensions in IRS-III are:

- UPML based types of knowledge models domain model, task, problem solving method and application have been extended to include goal models, mediator models and web service models,
- The WSMO ontology has been implemented, and slightly extended, in OCML [Motta, 1998] the underlying representation used in IRS,
- A WSMO specific Java API has been created, and
- The IRS browser have been customised to reflect WSMO.

3.1 The IRS-III Architecture

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

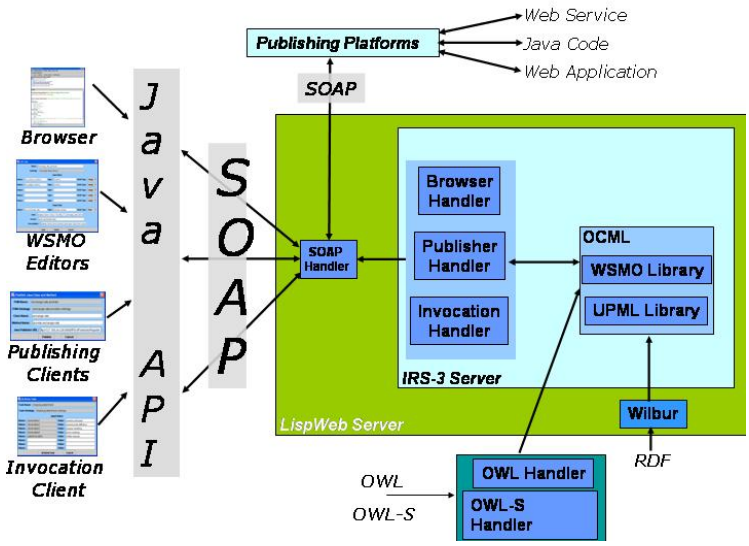

Fig. 1. The IRS-III Server Architecture

The IRS-III Server is based on an HTTP server written in lisp [Riva and Ramoni, 1996] which has been extended with a SOAP [SOAP, 2003] 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 within the WSMO library.

Publishing with IRS-III entails associating a specific 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.

3.2 The IRS-III Ontology

The IRS-III ontology is basically an implementation of WSMO standard D2v02 [WSMO, 2004] with some additional attributes. As can be expected the ontology underlying IRS-III is based on WSMO. Each main WSMO concept is represented as a class. 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.

3.3 Discovery

IRS discovery mechanism is based on gw-mediators and the assumptions. When a goal is invoked the IRS broker creates a set of possible contender web services using the gw-mediators. A specific web service out of the possible web services is then selected using an applicability function within the assumption slot of the web service's associated capability. If the applicability condition of more than one web service is satisfied then one is selected arbitrarily. We give an example of applicability functions in use in section 4 of the paper. 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.

3.4 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 [WSMO, 2004], which can be consumed by the IRS-III execution environment and reasoner. The API is also used within the browser and editor de-

scribed 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.

3.5 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 in 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 service, mediators and other classes. The user can also use the selector to include items which are inherited from ancestor ontologies.

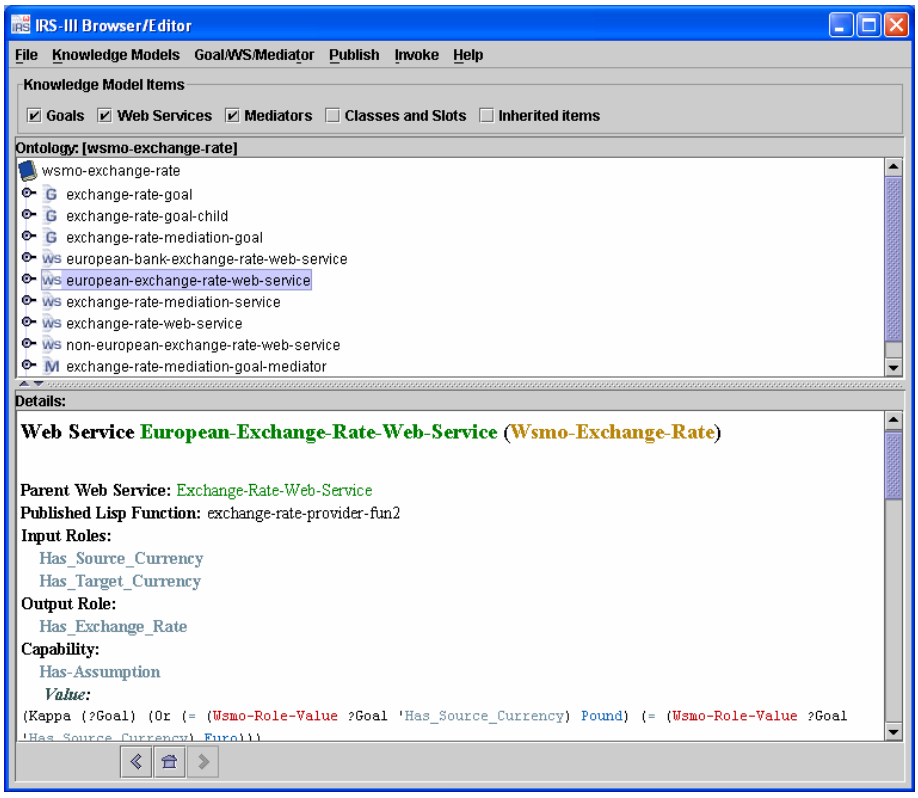

Fig. 2. A screen snapshot showing the main IRS-III browser.

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 ontolo-

gies, 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. So are 'standard' classes which describe domain concepts. 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. Figure 3 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 roles panel is shown in figure 3. The 'Add Input' and 'Delete Input' buttons can be used to add and delete input roles.

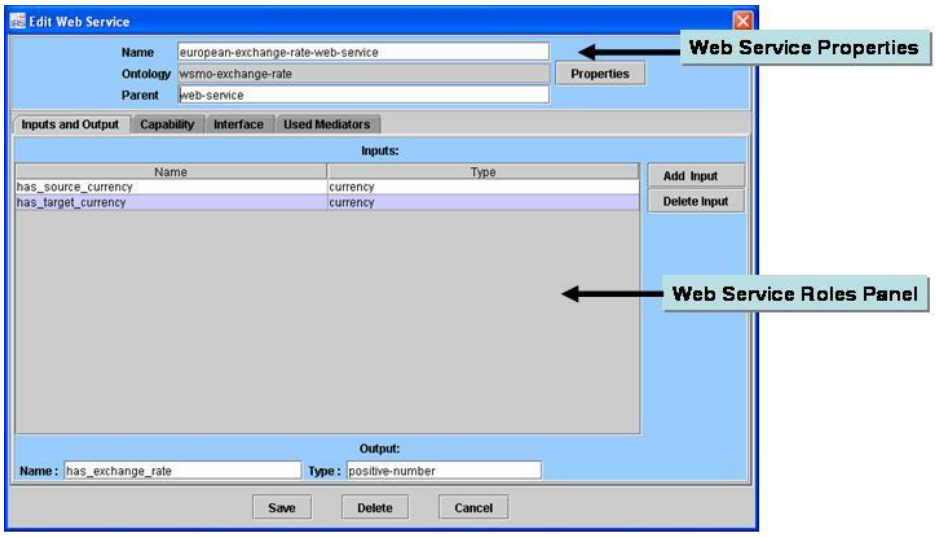

Fig. 3. A screen snapshot of the IRS-III dialog for defining web services.

IRS-III contains a simple goal discovery mechanism. Using the interface shown on the left of figure 4 a user can find goals whose input or output roles have a type which match a defined filter. In figure 4 the user is looking for all goals which have an input role which is a subclass of the currency class.

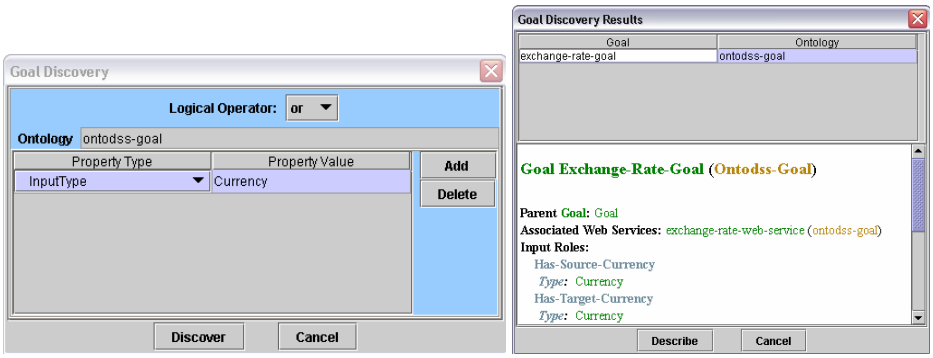

Fig. 4. Screen snapshots showing the input form (on the left) and output (on the right) of the IRS-III goal discovery interface.

4 A Simple Example

In the following example we have two services which give currency exchange rates. One for European currencies and one for non-European currencies. The WSMO definitions have the following form.

Goal

The goal has two input roles, `has_source_currency` and `has_target_currency` which are both of type `currency` and the output role `has_exchange_rate` which is of type `positive-number`.

```
exchange-rate-goal (goal)
  has-input-role :value has_source_currency
                 :value has_target_currency
  has-output-role :value has_exchange_rate
  has_source_currency :type currency
  has_target_currency :type currency
  has_exchange_rate :type positive-number
```

Fig. 5. The definition of the exchange rate goal.

Mediator

The exchange rate gw-mediator specifies that the source component is the exchange rate goal. The target component is not declared as this mediator is used in both web services (specified in the web service's associated capabilities). The value for mediation service is `exchange-rate-mediation-goal`. When the exchange rate goal is invoked the input values are transformed by a web service associated with `exchange-rate-mediation-goal` into a format suitable for a web service. In our example we use a web service which transforms currency symbols (e.g. '$' '£') into symbolic names (e.g. dollar, pound).

```
exchange-rate-mediator (gw-mediator)
  has-source-component :value exchange-rate-goal
  has-mediation-service :value exchange-rate-mediation-goal
```

Fig. 6. The definition of the exchange rate mediator.

Web Services
Two semantic descriptions have been created for European and non-European exchange rate web services. Figure 7 shows the definition of `european-exchange-rate-web-service`. The web service definition is relatively straightforward simply specifying the associated capability and interface.

```
european-exchange-rate-web-service (web-service)
  has-capability :value european-exchange-rate-capability
  has-interface :value european-exchange-rate-web-service-interface
```

Fig. 7. The definition of the European exchange rate web service.

Capability
The capability definitions for the European and non-European exchange rate web services both inherit from `exchange-rate-capability`. This capability specifies a mediator, `exchange-rate-mediator`, which links the web service to the exchange-rate goal. `European-exchange-rate-capability` contains an applicability function (in OCML) which specifies that the associated web service should be selected if the values for `has_source_currency` and `has_target_currency` are instances of the class `european-currency`.

```
exchange-rate-capability (capability)
  used-mediator :value exchange-rate-mediator

european-exchange-rate-capability (exchange-rate-capability)
  has-assumption
    :value
    (kappa (?goal)
      (and (european-currency
             (wsmo-role-value ?goal 'has_source_currency))
           (european-currency
             (wsmo-role-value ?goal 'has_target_currency))))
```

Fig. 8. The European exchange rate capability which inherits from a generic exchange rate capability.

Interface
The interface part of the web service is specified when we publish a deployed web service or code fragment against a web service description. Publication results in the IRS-III server creating an interface description including a choreography definition.

5 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 contains the following features:

- 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,
- Capability based invocation – web services can be invoked by asking for a WSMO goal to be solved.

To implement above we have extended WSMO, primarily by adding input and output roles to goal and web service descriptions. We have also defined a new type of mediator, a gw-mediator, which transforms the input values associated with an invoked goal into a format suitable for a target web service.

A final feature of IRS-III is that is open. Parts of the IRS-III brokering mechanism have been implemented as semantic web services. IRS-III users are free to publish their own services to implement these components if they desire.

IRS-III will be used as part of WSMO tutorials to be held in conjunction with the AIMSA and Semantic Web conferences. The API and browser can be downloaded from the IRS web site kmi.open.ac.uk/projects/irs/.

Acknowledgements

This work is supported by the DIP (Data, Information and Process Integration with Semantic Web Services) and AKT (Advanced Knowledge Technologies) projects. DIP (FP6 - 507483) is an Integrated Project funded under the European Union's IST programme. The AKT Interdisciplinary Research Collaboration (IRC), is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01.

References

[DIP] Data, Information, and Process Integration with Semantic Web Services - Objectives, http://dip.semanticweb.org/objectives.html

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

[Fensel and Motta, 2001] Structured Development of Problem Solving Methods, IEEE Transactions on Knowledge and Data Engineering, 13(6):9131-932.

[IDC, 2003] IDC, IDC direction 2003 Conference, Boston.

[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).

[Muse, 2003]. Web Services Market to Reach $21 Billion by 2007. Internet News, February 4, 2003 http://www.internetnews.com/xSP/article.php/1579171

[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).

[Riva and Ramoni, 1996] LispWeb: a Specialised HTTP Server for Distributed AI Applications. Computer Networks and ISDN Systems, 28, 7-11, 953-961.

[SOAP 2003] SOAP Version 1.2 Part 0: Primer, http://www.w3.org/TR/soap12-part0/

[UDDI 2003] UDDI Spec Technical Committee Specification v. 3.0, http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

[WSDL 2001] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[WSML, 2004] Languages for WSMO, http://www.wsmo.org/2004/d16/

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

[WSMX, 2004] Overview and Scope of WSMX, http://www.wsmo.org/2004/d13/.