

# Design of Intelligent Agent-based Web Services Gateway

**Noh-Sam Park, Eun-Jin Ko, Gil-Haeng Lee**

Network Technology Laboratory  
Electronics & Telecommunications Research Institute  
161 Gajeong-Dong, Yuseong-Gu, Daejeon, 305-350 KOREA  
TEL : +82-42-860-3826 FAX : +82-42-860-6342  
{siru23, ejko, ghlee}@etri.re.kr

## **Abstract**

Web services enable the integration of application independent of platforms, programming languages. Moreover the intelligent web services approach makes it possible to express information in a machine-readable form.

In this paper, we first introduce web services and the concept of intelligent web services. We then propose an agent-based web services gateway. We designed the intelligent web services platform using existing standards. In particular, we present an agent approach to dynamic web services. The proposed gateway invokes access points of UDDI registry automatically and returns execution results to consumers.

The suggested approach is used to build experimental solutions involving intelligent web service systems. Finally, we'll give an example to illustrate a typical scenario.

**Keywords:** Web Services, Agent, Gateway

**Contact Person:** Noh-Sam park (siru23@etri.re.kr)

# Introduction

---

## •Web-Service

- A software application identified by a URI
- Its public interfaces and bindings are defined and described using XML
- Interoperable with agreement on several standards
  - UDDI
    - register and make web services available to consumers
  - WSDL
    - describe interface and functionality
  - SOAP
    - a base communication protocol for consumers to exchange XML messages
- Facilitate web-based system integration
- Make it possible for diverse applications to find each other and exchange data

An Web service is a software application identified by a Uniform Resource Identifier(URI)[1], whose interfaces and binding are capable of being defined, described and discovered by XML artifacts. Its definition can be found by other software systems. It supports direct interactions with other software applications using XML based messages via internet-based protocols.

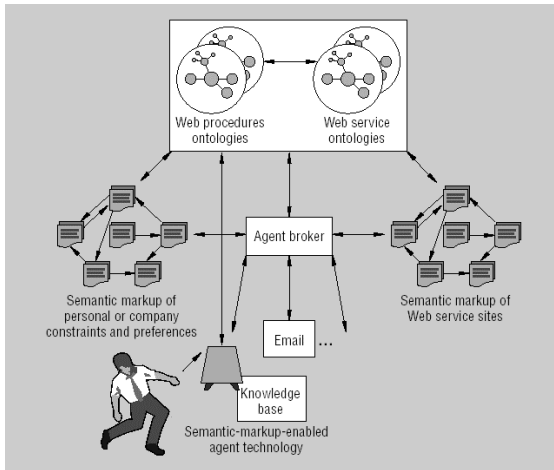
Standards of Web Services are still developing. UDDI(Universal Description, Discovery, and Integration) [2] is a standard for web services to register and make web services available to consumers. WSDL(Web Services Description Language) [3] is a standard for web services to describe their interface and functionality. SOAP(Simple Object Access Protocol) [4] defines a base communication protocol for consumers to exchange XML messages with each other.

Web services are not especially new [5]. Application service providers (ASPs) have been providing end-user based web services for many years, but web services make it possible for diverse applications to find each other and exchange data seamlessly via the Internet. For instance, programs written in Java and running on Solaris can find and call codes written in C# that run on Windows XP, or programs written in Perl that run on Linux, without any regard to the details of how that service is implemented.

Today's web was designed primarily for human interpretation and use. Nevertheless, we are seeing increased automation of web services interoperation. To let computer programs or agents implement the reliable, large-scale interoperation of web services, Semantic Web Services, whose properties, capabilities, interfaces, and effects are encoded in an unambiguous, machine-understandable format, are recommended.

We present an approach to an intelligent agent-based web service gateway that provides dynamic and effective results via SOAP-compliant agents. Web services middleware accepts consumers' requests, which are accessed to UDDI server, and it executes web services which are the results of a UDDI search. In addition to the execution results provided by middleware, consumers can find the web services that they really want.

# Semantic Web Services



- Automating web services
  - Automatic web service discovery
  - Automatic web service execution
  - Automatic web service composition and interoperation
- Semantic web service markup
  - Domain-independent web service ontologies

APNOMS 2003

(3)

ETRI

## •Automating web services

Automatic web service discovery involves automatically locating web services that provide a particular service. With a semantic markup of services, we can specify the information necessary for web service discovery as computer-interpretable semantic markup at the service web sites, and provide a service registry.

Automatic web service execution involves an agent automatically executing an identified web service. The semantic markup of web services provides a declarative, computer-interpretable API for executing services. The markup tells the agent what input is necessary, what information will be returned, and how to execute the service automatically.

Automatic web service composition and interoperation involves the automatic selection, composition, and interoperation of appropriate web services in order to perform some task when given a high-level description of the task's objective.

## •Semantic web service markup

The three automation tasks are driving the development of semantic web services markup in the DAML family of markup languages. Reference [6] is marking up

- Web services, such as Yahoo's driving direction information service or United Airlines' flight booking service;
- user and group constraints and preferences, such as a user's schedule that he/she prefers driving over flying if the driving time to his/her destination is less than three hours; and
- agent procedures, which are (partial) compositions of existing web services, designed to perform a particular task and marked up for sharing and reuse by groups of other users.

In [6], ontologies are used to encode the classes and subclasses of concepts and relations pertaining to services and user constraints. Domain-independent web service ontologies are augmented by domain-specific ontologies that inherit concepts from the domain-independent ontologies and that additionally encode concepts that are specific to the individual web service or user.

# Intelligent Web Services

---

- SWOBIS
  - Take descriptions of services(metadata) to enable service discovery
  - A self-updating list of software tools for the semantic web
  - SWST ontology creation
  - Interpret the mapping results of SWST & visualize information from ontologies in one report
- ITTALKS
  - A web portal that lists information technology talks
  - Internally DAML use for knowledge representation, reasoning, and communication
  - ITTALKS does not provide robust direct external access to its underlying DAML content
- DAML-S
  - A promising add-on to the DAML+OIL language for describing services
  - a merger between DAML-S and current industry quasi-standards
- UPML
  - Sit on top of the DAML+OIL layer
  - Defines the architecture for describing reasoning services on the Web

SWOBIS is an acronym of the Semantic Web Ontology-Based Information Service, which is analogous to the recommended UDDI registry, which takes descriptions of services expressed as metadata in order to enable service discovery[8].

SWOBIS offers a self-updating list of software tools for the semantic web generated as a web-based report, by utilizing the descriptions of software tools provided as metadata on the web. In addition to showing the next step, SWOBIS keeps the research community updated on the performance of current semantic web technologies, thus supporting the vision of the semantic web.

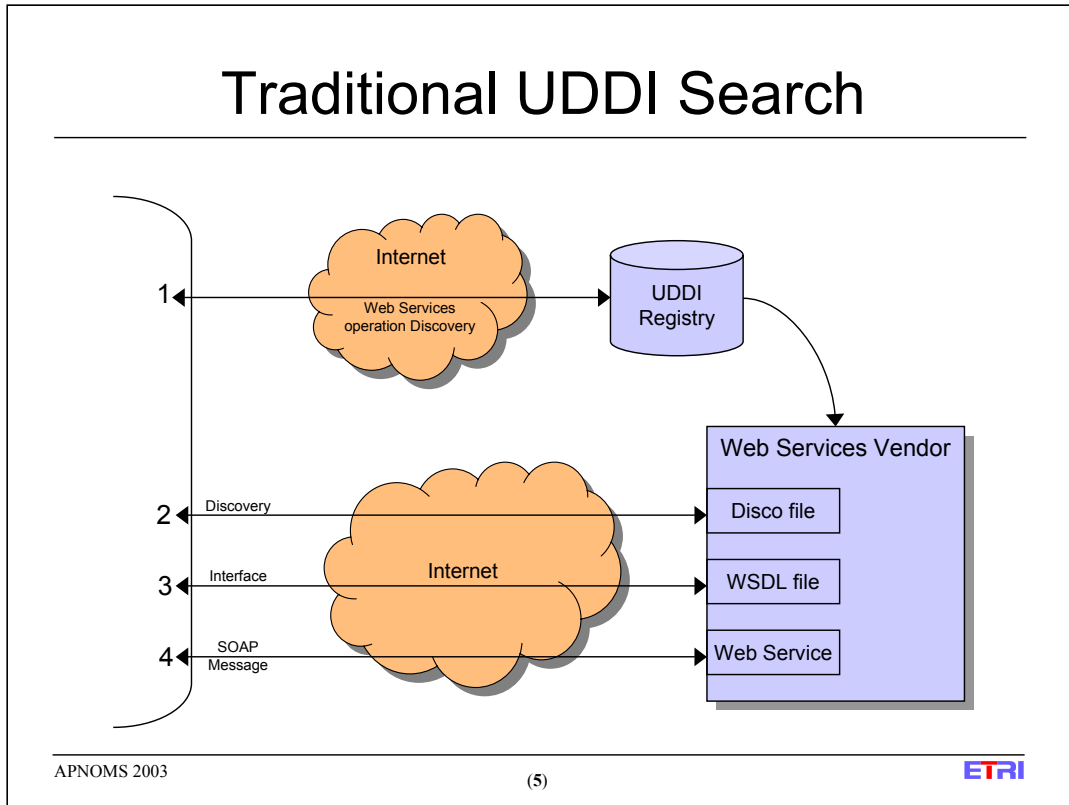
In the first step, the Semantic Web Software Tools(SWST) ontology was created, which is needed to describe software tools in a machine-understandable way. SWOBIS is capable of interpreting the mapping results of SWST and thus can visualize information from various ontologies in one report. The report always contains up to date information. The user saves time that would have been otherwise spent retrieving information by looking at the list instead of visiting all web pages.

ITTALKS is a web portal that lists information technology talks, such as distinguished lectures at universities. It internally uses DAML for knowledge representation, reasoning, and communication[7]. A user can fill out a standard Web form that will result in a DAML file containing data that ITTALKS can process, which the users can then host on their own Web server. Talk information can similarly be entered via traditional HTML forms, or one can submit the URL of a DAML file using the standard ITTALKS content. Talks information is kept in a database, much like in any other (non-Semantic-Web) Web service. ITTALKS does not provide robust direct external access to its underlying DAML content. This means that it is impossible to robustly republish, search, or annotate its content - negating the key idea of the Semantic Web. Thus, while ITTALKS is a novel and immediately useful “traditional” Web service, its internal use of DAML seems to provide few benefits to external users.

DAML-S is a promising add-on to the DAML+OIL language for describing services[9]. In fact, the need for a merger between DAML-S and current industry quasi-standards has even been recognized by the developers of DAML-S. However, to the best of our knowledge, it has not yet been specified in any way, nor have there been any implemented web services using DAML-S at the time of writing.

UPML, the Unified Problem-solving Method Development Language, sits on top of the DAML+OIL layer just like DAML-S, and defines the architecture for describing reasoning services on the Web[10]. Using UPML, IBROW aims to develop intelligent brokers that are able to configure reusable components into knowledge systems via the web [11].

# Traditional UDDI Search



UDDI is a collection of specifications for distributed Web-based information registries of Web services. These specifications are broken down into a number of categories. UDDI is also a set of implementations of the specifications that allow businesses to register information about the Web services that they offer so that other businesses can find them. These implementations are publicly accessible. Also, UDDI registries are themselves available as Web services.

Traditional UDDI can provide answers to queries such as:

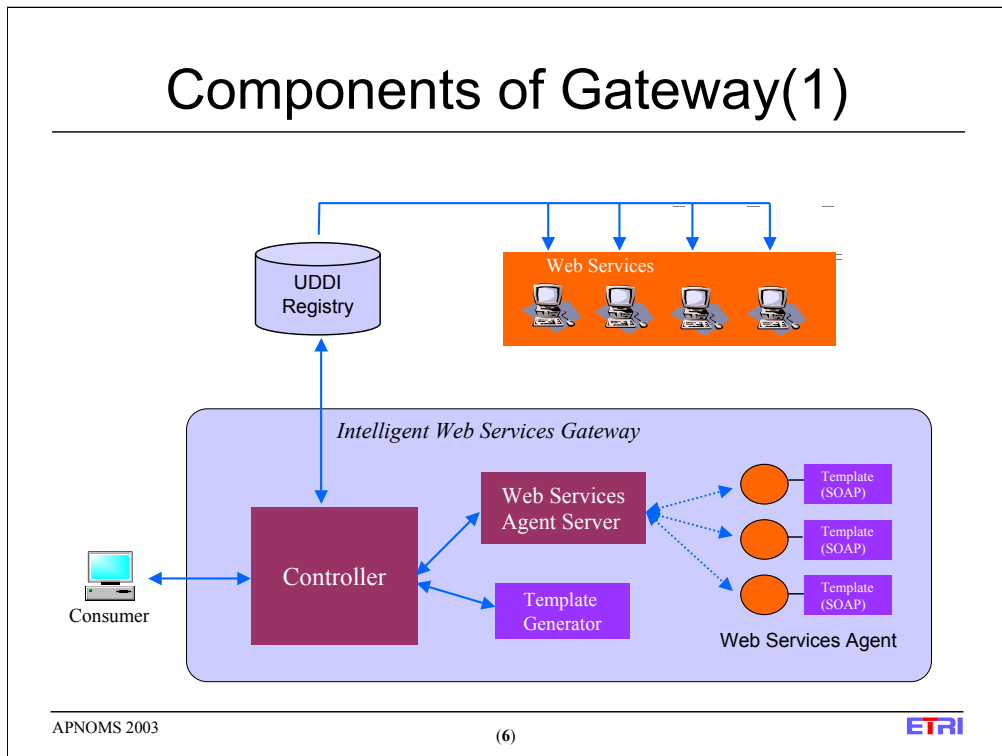
- What web services does a specific business provide?
- What are all the known endpoints for a specific web service?
- What is the current binding information (supported protocols, and so on.) for a specific web service endpoint?

Other possible queries, such as price comparison of Web services or geographic proximity, are not part of the UDDI specification.

In the traditional method, consumers search web services with UDDI, and manually access the web services that appear in the result. The pitfalls are that UDDI search results only provide specifications for registered web services, and cannot express if they are what consumers really want. Also it is impossible to know the states of registered web services.

We designed the intelligent web services gateway to make requests to the UDDI server, and retrieve search results. After that, the gateway automatically generates appropriate random values by analyzing web services' registry. This generation is processed with a template generator. Web services gateway accesses web services' URI with those randomly generated values.

# Components of Gateway(1)



We incorporate web service technologies, namely UDDI, SOAP, and WSDL in intelligent web services gateway. We design a proper platform so that consumers can interact with the UDDI server via SOAP. Consumers can obtain results of web services via the user interface supported by the gateway. Consequently, our system is a framework with the autonomy of agents and the convenience of web services.

The intelligent web services gateway consists of a controller, web services agent server, template generator and web services agent. Consumers can use a convenient and straightforward method to enjoy web services in the same way that they use a UDDI server.

**Controller** gets consumers' input, searches UBR with it, and passes UDDI search results to the web services agent server. Finally, the controller also displays the results to the consumers. In order to communicate with the UDDI server and to obtain web services, the controller is designed to be able to connect to UDDI with SOAP messages, and to apply the functionality that is provided by UDDI. The controller is made with the ability to run SOAP, so that consumers can interact with the UDDI server

**Web services agent server** takes the results of a UDDI search from the controller, and creates as many web services agents as the number of results. A web service has an access point, and the agent server will set up execution rules for the service agent according to user's demands, preferences, access points.

## Components of Gateway(2)

---

- **Intelligent Agent-based Web Services Gateway**
  - A framework with the autonomy of agents and the convenience of web services
  - **Components**
    - **Controller** : connect to UDDI with SOAP messages, apply the functionality that is provided by UDDI
    - **Web services agent server** : create web services agents, set up execution rules for the service agent
    - **Template generator** : generate templates to invoke web services
    - **Web services agent** : interact with a web service, find operations of the web service, invoke the appropriate web service

**Template Generator** generates templates in order to invoke web services actually. The template has a SOAP format, contains WSDL messages from web services, and proper values. Values are randomly generated according to a parameter's number and type. Furthermore, templates have consumers' preference information to controls agent's process with.

**Web Services agent** interacts with a web service, and finds operations of the web service. Then it analyzes the parameter's number and the type of the WSDL via a template generator. The agent invokes the appropriate web service of service provider with a template generated by template generator. Depending on the service provider's circumstances, the web service is in-service or not, although it is registered to UDDI. The agent directly accesses the web service via SOAP, and sends the result to consumers. Consequently consumers determine whether the web service is alive. Moreover, as we provide execution results invoked with randomly generated values, consumers know how exact the provided web service is.

Whenever the services agent server's request reaches a service agent, it can interact with a service provider automatically according to the contents of the request and the rules that the system made before. The result will be sent to the services agent sever and the server passes the result to the controller. Finally the controller displays the result to a consumer. When it shows the result, the user interface is designed to be able to get actual parameters. The consumer determines search results, and uses the web service by setting the parameter values to actual values.

# Search using Gateway

- Example : find a phone number using web services
  - UDDI search results

Service name	Access point
<b>Phonebook</b>	<b><a href="http://wma5.icominfo.com/WIDWebService/Service1.asmx">http://wma5.icominfo.com/WIDWebService/Service1.asmx</a></b>
PhoneConverter	<a href="http://www.redune.cc/PhoneConverter/Convert.asmx">http://www.redune.cc/PhoneConverter/Convert.asmx</a>
Phonehome	<a href="http://www.anothergp.com/test">http://www.anothergp.com/test</a>
Phone Number	1 512 750 8453
Phone Enquiry	352-392-2680

- Consumer
  - Input search keyword 'Phone'
- Controller
  - Retrieves a list of UDDI searches and passes it to web services agent server
- Web service agent
  - Finds & Invokes operation 'GetListings'

We illustrate a scenario that involves the technologies we discussed. All the actions have agents, UDDI, WSDL and SOAP technologies behind.

We assumed that a consumer wants to find a phone number on the internet using web services in this scenario. Typically, in order to find an appropriate web service, he/she could visit a UDDI web site, or type a keyword such as phone, phonebook. As a result of UDDI search, he/she obtains a list of web services. Table shows a list of UDDI search results with the keyword phone. It may contain what he/she needs, but to check the result, he/she must manually access the web services one by one.

We can get a result using intelligent web services gateway in another way. The consumer starts the search by typing 'phone', and awaits the detailed execution results until gateway provides them. The controller retrieves a list of UDDI searches like Table, and passes it to the web services agent server. In the case of the web service, whose name is phonebook, the agent server will create a web service agent with the access point such as '<http://wma5.icominfo.com/WIDWebService/Service1.asmx>'.

As the service agent interacts with the access point, it finds the operation *GetListings* and a description, which is '*Retrieves listings from a mainframe phonebook*'. The operation and the description are also provided to the consumer. The agent analyzes the operation in order to find parameters' number and type. The *GetListings* operation has two parameters, **sName** and **nMaxListings**, and each type is **string** and **int** respectively.

# SOAP Messages

POST /WIDWebService/Service1.asmx	HTTP/1.1 200 OK
HTTP/1.1	Content-Type: text/xml; charset=utf-8
Host: wma5.icominfo.com	.....
Content-Type: text/xml; charset=utf-8	<?xml version="1.0" encoding="utf-8"?>
.....	<soap:Envelope
SOAPAction: "http://www.icominfo.com/webservices/example/ms/phonebook/GetListings"	.....
<?xml version="1.0" encoding="utf-8"?>	xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Envelope	<soap:Body>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"	<GetListingsResponse
xmlns:xsd="http://www.w3.org/2001/XMLSchema"	xmlns="http://www.icominfo.com/webservices/example/ms/phonebook">
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">	<GetListingsResult>
<soap:Body>	<Listing>
<GetListings	<Name>ABDALIAN, LINDSEY</Name>
xmlns="http://www.icominfo.com/webservices/example/ms/phonebook">	<Phone>979-695-8822</Phone>
<sName>abc</sName>	<Grade>U3</Grade>
<nMaxListings>5</nMaxListings>	<Major>ENGL</Major></Listing>
</GetListings>	<Listing>
</soap:Body>	.....
</soap:Envelope>	</GetListingsResult></GetListingsResponse>
	</soap:Body>
	</soap:Envelope>
SOAP request message	SOAP response message

APNOMS 2003

(9)



The agent creates the template, whose format may be SOAP or HTTP. The template generator generates the value of **abc**, whose name is **sName**, and a value of **5**, whose name is **nMaxListings**.

The agent will automatically start the web service and reply from the service provider. If the web service is in-service state, the SOAP response message will be sent. In the response message, the number of the **Listing** is **5**, the same as the number of **nMaxListings** in the request message. The name adjacent to **abc** is displayed in the response message.

The same process is applied to other web services, and SOAP response messages are gathered in the controller via the web service agent server. However, the remaining two results of Table (*Phone Number*, *Phone Enquiry*) are not processed in the same way. Their access points are not proper URIs but a mere telephone number, so the web services agent server validates access points not to take any more steps, and the failed results are returned to the consumer.

We assumed the other two services are not what the user wants to find, so they will not provide proper results to the consumer. Finally, the web service the consumer wants is *phonebook*. Instead of accessing all the web services, the consumer can find web services by reviewing the results provided by gateway.

## Conclusion & Future Work

---

- UDDI as Web Services Broker
- Pitfalls of web services search using UDDI
- Agent-based Web Services Framework
  - Agent approach to dynamic web services
  - Dynamic web services results
- Future work
  - User preferences
  - Web services caching

The role of an web service broker is essential in web services architecture to make it possible for potential consumers to easily find an web service. UDDI registries fulfill the role of an web service broker. In this paper, we explored a search methods of using UDDI and discovered its pitfalls. We recommended an intelligent agent-based web services gateway, so that consumers can obtain dynamic and effective results from web services. The proposed gateway utilizes the existing standards such as UDDI, WSDL, SOAP instead of suggesting another mechanism. The gateway makes it possible for consumers to communicate with web services in the same way that they use search engine.

Firstly, we introduce the concept of web service and intelligent web services. Next, we propose an intelligent agent approach for the web services. Then we analyzed the components of the intelligent web services gateway. We facilitate the user's convenience, in the same way as a UDDI search. An example using the suggested gateway is presented, and we showed request and response SOAP messages.

As consumers configure their preferences, the gateway must give a variety of option in order to express the results. And to decrease network traffic load, the gateway should adopt the web caching. According to the implications of the research, future work has been conducted to the preferences and the caching of UDDI invoke results.

## References

- [1] RFC-2396, <http://www.faqs.org/rfcs/rfc2396.html>
- [2] UDDI.ORG Version 3.0 specification, [http://uddi.org/pubs/uddi\\_v3\\_features.htm](http://uddi.org/pubs/uddi_v3_features.htm)
- [3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Web Services Description Language(WSDL) Version 1.1. W3C Note 15, March 2001, <http://www.w3.org/TR/wsdl>
- [4] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, Simple Object Access Protocol(SOAP) 1.1., <http://www.w3.org/TR/SOAP>, 2000
- [5] David Orchard, "Web Services Pitfalls", <http://www.xml.com/pub/a/2002/02/06/webservices.html>, XML.com, Feb, 2002.
- [6] Sheila A. McIlraith, Tran Cao Son, Honglei Zeng, "Semantic Web Services", IEEE Intelligent Systems. Special Issue on the Semantic Web. 16(2):46-53, March/April, 2001
- [7] Sollazzo, T., Handschuh, S., Staab, S., Frank, M., "Semantic Web Service Architecture – Evolving Web Service Standards toward the Semantic Web," Proc. of the 15th International FLAIRS Conference. Pensacola, Florida, May 2002
- [8] Cost, R. S., Finin, T., Joshi, A., Peng, Y., Nicholas, C., Chen, H., Kagal, L., Perich, F., Zou, Y., Tolia, S., "Ittalks : A case study in the semantic web and daml," International Semantic Web Working Symposium, 2001
- [9] Ankolenkar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H., "DAML-S: A Semantic Markup Language For Web Services," In Proceedings of SWWS'01, August 2001.
- [10] Fensel, D., Benjamins, R., Motta, E., Wielinga, B. "UPML : A Framework for knowledge system reuse," In Proceedings of the Internatioinal Joint Conference on AI(IJCAI-99), 1999
- [11] Benjamins, V. R., Plaza, E., Motta, E., Fensel, D., Studer, R., Wielinga, B., Schreiber, G., Zdrahal, Z., Decker, S., "IBROW3 : An intelligent brokering service for knowledge component reuse on the world-wide web," In The 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop(KAW98), 1998