DIP

Data, Information and Process Integration with Semantic Web Services

FP6 – 507483

Deliverable

WP4: Service Usage

D4.18

QoS-enabled Service Discovery Component Prototype 1 Report

Le-Hung Vu (EPFL), Sebastian Gerlach (EPFL),
Fabio Porto (EPFL), Othman Tajmouati (EPFL), Manfred Hauswirth (EPFL)

August 11, 2006

Executive Summary

This report includes the installation instruction and a brief description of the *first prototype* of the QoS-enabled Semantic Web service discovery component, following the specification in the Deliverable D4.17.

The QoS-enabled Semantic Web service discovery is the process of automatically finding Web services that fulfill a certain user goal in terms of their quality of service (QoS) criteria. Typically, users express a goal by specifying their functional and QoS requirements the Web services should provide in order to achieve it. We extend goals and Web service descriptions to support the specification of QoS parameters and provide a discovery component which is capable of combining both QoS and functionality-based service discovery into one integrated module.

Regarding the development of exploitable tools, this report and its associated QoS discovery prototype has the following contributions:

- a first implemented discovery prototype capable of doing the most important functionality of a QoS discovery component: select and rank the services fulfilling user's QoS requirements by doing the semantic matchmaking and ranking between a list of services against the submitted user goal;
- the developed component could be used in two ways: as a stand-alone service discovery application which includes the capability of the functionality-based service discovery component; or as a discovery module integrated into a WSMX installation. The implementation of our QoS discovery prototype conforms to the WSMX/DIP API, making it be easy to use for the interested DIP partners and be inter-operable with the other DIP tools such as the WSMO Studio;
- we also provide a list of WSMO ontologies: upper ontologies for QoS discovery and ranking algorithms, example working full-fledged WSML Web service and goal descriptions, as well as the dedicated QoS and ranking ontologies for each developed example. These are useful for the demonstration of the modeling of QoS requirements and offerings in various realistic application scenarios.

Therefore, this report and its associated QoS discovery prototype are relevant for the following audience: the use case partners, the WSMO and WSML developing group, the developers and IT experts who are interested in technological solutions for Semantic Web service discovery based on QoS and/or non-functional properties criteria. In the DIP framework, the potential readers (respectively users) of this report (respectively the QoS discovery prototype) are:

- WP1 - to consider the extensions of the WSMO model to support QoS parameter modeling more explicitly;
- WP2 - requirements for the repository interface for retrieving Web service descriptions and ontologies as input parameters for the discovery process;
- WP8 - use case partner defining a B2B Telecom case study with QoS;
- WP10 - use case partner defining an e-banking case study with QoS;

- Other partners interesting in QoS-based Semantic Web service discovery and its applications;

Document Information

IST Project Number	FP6 – 507483	**Acronym**	DIP
Full Title	Data, Information, and Process Integration with Semantic Web Services		
Project URL	http://dip.semanticweb.org/		
Document URL			
EU Project Officer	Kai Tullius		

Deliverable	**Number**	4.18	**Title**	QoS-enabled Service Discovery Component Prototype 1 Report
Work Package	**Number**	4	**Title**	Service Usage

Date of Delivery	**Contractual**	31-Jun-2006	**Actual**	31-Jun-2006
Status	version 0.1		final ☐	
Nature	prototype ☐ report ☒ dissemination ☐ ontology ☐			
Dissemination Level	public ☒ consortium ☐			

Authors (Partner)	Le-Hung Vu (EPFL), Sebastian Gerlach (EPFL), Fabio Porto (EPFL), Othman Tajmouati (EPFL), Manfred Hauswirth (EPFL)			
Resp. Author	Le-Hung Vu, Sebastian Gerlach		**E-mail**	lehung.vu@epfl.ch sebastian.gerlach@epfl.ch
	Partner	EPFL	**Phone**	+41 (21) 693-7573

Abstract (for dissemination)	This report includes the installation instruction and description of the implemented features of the prototype 1 of the QoS-enabled discovery component.
Keywords	Semantic Web service, SWS, service discovery, QoS, Goal,API

Version Log			
Issue Date	**Rev No.**	**Author**	**Change**
21-06-2006	1	Le-Hung Vu	v 0.1 - First draft of the report
30-06-2006	2	Le-Hung Vu, Manfred Hauswith, Fabio Porto	v 0.2 - The second version of the report released to dip-all
09-08-2006	3	Le-Hung Vu	v 0.3 - Revised version of the report w.r.t to Sigurd Harand's comments
11-08-2006	4	Le-Hung Vu, Manfred Hauswith, Fabio Porto	v 0.4 - Revised version of the report w.r.t to two other reviewers' comments

Reviewers				
	Frank Kausfer and Matthias Klusch		**E-mail**	frank@kaufer.de, klusch@dfki.de
	Partner	DFKI, Germany	**Phone**	+49 (681) 302 5297
	Luís Costa		**E-mail**	Luis.F.Costa@sintef.no
	Partner	SINTEL, Norway	**Phone**	+47 (220) 673 11

Project Consortium Information

Partner	Acronym	Contact
National University of Galway	NUIG 	Dr. Sigurd Harand Digital Enterprise Research Institute (DERI) National University of Ireland, Galway Galway Ireland E-mail: sigurd.harand@deri.org Tel: +353 91 495112
Fundacion De La Innovacion.Bankinter	Bankinter 	Monica Martinez Montes Fundacion de la Innovation. BankInter, Paseo Castellana, 29 28046 Madrid, Spain Email: mmtnez@bankinter.es Tel: 916234238
British Telecommunications Plc.	BT 	Dr. John Davies BT Exact (Orion Floor 5 pp12) Adastral Park Martlesham Ipswich IP5 3RE, United Kingdom Email: john.nj.davies@bt.com Tel: +44 1473 609583
Swiss Federal Institute of Technology, Lausanne	EPFL 	Prof. Karl Aberer Distributed Information Systems Laboratory École Polytechnique Féderale de Lausanne Bât. PSE-A 1015 Lausanne, Switzerland E-mail : Karl.Aberer@epfl.ch Tel: +41 21 693 4679
Essex County Council	Essex 	Mary Rowlatt, Essex County Council, PO Box 11, County Hall, Duke Street, Chelmsford, Essex, CM1 1LX, United Kingdom. E-mail: maryr@essexcc.gov.uk Tel: +44 (0)1245 436524
Forschungszentrum Informatik	FZI 	Andreas Abecker Forschungszentrum Informatik Haid-und-Neu Strasse 10-14 76131 Karlsruhe Germany E-mail: abecker@fzi.de Tel: +49 721 96540
Institut für Informatik, Leopold-Franzens Universität Innsbruck	UIBK 	Prof. Dieter Fensel Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck, Austria Email: dieter.fensel@deri.org Tel: +43 512 5076485

ILOG SA	ILOG	Christian de Sainte Marie 9 Rue de Verdun, 94253, Gentilly, France E-mail: csma@ilog.fr Tel: +33 1 49082981
inubit AG	inubit	Torsten Schmale, inubit AG, Lützowstraße 105-106 D-10785 Berlin, Germany E-mail: ts@inubit.com Tel: +49 30726112 0
Intelligent Software Components, S.A.	iSOCO	Dr. V. Richard Benjamins, Director R&D Intelligent Software Components, S.A. Pedro de Valdivia 10 28006 Madrid, Spain E-mail: rbenjamins@isoco.com Tel. +34 913 349 797
MDR Partners	MDR	Rob Davies, MDR Partners, 8 St. Andrew Street, Hertford, Herts., United Kingdom, SG14 1JA, Email: rob.davies@mdrpartners.com Tel. +44 (0)208 8763121
Hanival Internet Services GmbH	HANIVAL	Alexander Wahler, Hanival Internet Services GmbH, Kirchengasse 13/1a A-1070 Wien Email: wahler@niwa.at Tel. +43(0)1 3195843-11
The Open University	OU	Dr. John Domingue Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK E-mail: j.b.domingue@open.ac.uk Tel.: +44 1908 655014
SAP AG	SAP	Dr. Elmar Dorner SAP Research, CEC Karlsruhe SAP AG Vincenz-Priessnitz-Str. 1 76131 Karlsruhe, Germany E-mail: elmar.dorner@sap.com Tel: +49 721 6902 31
Sirma AI Ltd.	Sirma	Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD, Office Express IT Centre, 3rd Floor 135 Tzarigradsko Chausse, Sofia 1784, Bulgaria E-mail: atanas.kiryakov@sirma.bg Tel.: +359 2 9768 303

Unicorn Solution Ltd.	Unicorn	Jeff Eisenberg Unicorn Solutions Ltd, Malcha Technology Park 1 Jerusalem 96951, Israel E-mail: Jeff.Eisenberg@unicorn.com Tel.: +972 2 6491111
Vrije Universiteit Brussel	VUB Vrije Universiteit Brussel	Pieter De Leenheer, Starlab- VUB Vrije Universiteit Brussel Pleinlaan 2, G-10 1050 Brussel, Belgium E-mail: Pieter.De.Leenheer@vub.ac.be Tel.: +32 (0) 2 629 3749

List of Keywords/Abbreviations

- CoDIMS-G - Configurable Data Integration Middleware System for the Grid.
- DBMS - Database Management System
- DHT - Distributed Hash Table
- EAI- Enterprise Application Integration.
- NFP - Non-Functional properties.
- P2P - Peer-to-Peer
- QoS - Quality of Service
- NFPs - Non-Functional Properties
- SWS - Semantic Web service
- UDDI - Universal Description, Discovery and Integration protocol
- WSD - Web service Description
- WSMO - Web Service Model Ontology
- wsmo4j - WSMO API for Java
- WSML - Web Service Model Language
- WSMX - Web Service Execution Environment
- QML - Quality of service Modeling Language
- WSLA - Web Service Level Agreement
- WSOL - Web Service Offering Language

Table of Contents

List of Figures

1 Quick Installation Instructions

For impatient readers, the simplest and fastest way to run the QoS discovery component is to download the whole compressed bundle available at: `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/qosdisc.zip`, unzip it into a local directory and run the shell script file *run.bat* in the extracted archive.

2 Detailed Installation Instructions for Window-based Systems

2.1 Downloading the Necessary Files

The main download page for the QoS discovery component is at: `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/`. From this starting point you can find the links to all other related documents.

To install the QoS discovery component, the simplest and fastest way is to download the whole compressed bundle available at: `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/qosdisc.zip`.

The following files are in the downloaded archive:

- `./qosdisc/qosdisc.wsmx`: the JAR file containing the binary executables of the QoS-enabled discovery component. This can be used separately or added into a WSMX installation as a component of the WSMX framework.
- `./qosdisc/qosdisc.properties`: the properties file for the configuration of the QoS discovery component.
- `./qosdisc/run.bat`: the shell script file for running the component in standalone mode.

In the subdirectory `./qosdisc/lib` of the archive, there are the following additional libraries:

- The library for the functionality discovery component *funcdisclite.jar*. Currently, this is the implementation of a light-weight semantic discovery.
- KAON2 Reasoning engine: *kaon2-2005-11-14.jar*
- Log4J library: *log4j-1.2.13.jar*
- WSML reasoner wrapper: *wsml2reasoner-20060522.jar*
- WSML parser library: *wsmlparser-20060210.jar*
- WSMO4j 0.5.2: *wsmo4j-0.5.2.jar* and
- WSMO API 0.5.2 library: *wsmo-api-0.5.2.jar*
- The WSMX integration API (*wsmx-integration-API-2006.jar*).

Note: The above files can also be downloaded separately from the main page `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/`. However, a user should pay attention to save the above files to his or her computer with their *original* names. Some browsers like Microsoft Internet Explorer have the tendency to automatically save a file under the new name with a default extension according to the file type, e.g., the file *qosdisc.wsmx* may be saved under the name *qosdisc.zip*, which makes thing more confusing.

2.2 Configuring the QoS-enabled Discovery Component

The default *qosdisc.properties* file has already been configured sufficiently for testing and running the QoS-enabled discovery component. The instructions bellow should help the user to better understand and re-configure it for other test cases according to his/her requirements:

- Property *ranking*: URI of the ontology to define the base concepts of the ranking algorithms.
- Property *comparison*: URI of the ontology to define the comparison between QoS instances (should be the QoS upper ontology)
- Property *output*: local path name of the directory to produce the output file containing the description of the ontological instances which describe the returned ranking values
- Property *service1*, *service2*, etc : URIs of the WSMO web service descriptions that are to be used as inputs of the discovery process. These descriptions should be annotated with the QoS description appropriately, as in the provided example descriptions in the default *qosdisc.properties* file.
- Property *goal*: URI of the WSMO goal containing the QoS requirements of the user, to be matched again the services pointed by the properties *service1*, *service2*, etc. The goal descriptions should also be semantically annotated with the QoS requirements, as in the provided example goals in the default *qosdisc.properties* file.
- Property *wsmxhost*: URI of the WSMX host entry point for testing.
- Property *functional*: set to *true* or *false* depending on whether we want the QoS discovery component to call the functionality-based discovery component or not.
- Property *subst1*, *subst2*, etc.: These values are used as follows: with each value of these properties of the form $A \rightarrow B$, *any* appearance of the string A in the namespace of *any* ontology mentioned in the *qosdisc.properties* file will be replaced by the string B when the discovery component loads the ontologies into its knowledgebase. The meaning of these replacements will be explained in Section 4.3.

The last part of the properties file is the configuration for various loggers of the discovery component. For the testing phase, one may need to set some loggers to the DEBUG/WARN level in order to turn/off the details information about the discovery process.

log4j.logger.ch.epfl.qosdisc.operators.ReasoningContext=DEBUG

The user can also reconfigure the loggers to print out the result/debug/info messages to a log file instead of the console.

NOTE: The URIs of the goal, service, related ontologies could be a remote identifier such as:

goal=`http://lsirpeople.epfl.ch/lhvu/ontologies/EUStockIndiciesGoal.wsml`

or could be a local path name:

goal=`file:///C:/TestOntologies/EUStockIndiciesGoal.wsml`

2.3 Installing the WSMX Framework (optional)

This installation is only necessary if the user wants to work with the WSMX framework. For convenience, we also provide the core components of the WSMX distribution that are necessary for the installation and deployment of our discovery component. These files are available at: `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/wsmx/wsmx_components_bin-0.3.zip` and `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/wsmx/wsmx_core_bin-0.3.zip`. The WSMX installation is as follows:

- Uncompress all files in the two above archives to *the same* folder, for example, `C:\WSMX`. Note that for simplicity, we should use the same directory for both the components and core files of WSMX.
- Follow the WSMX installation instructions in the file `C:\WSMX\INSTALL` to know how to run WSMX with a policy file suitable to your environment.

2.4 Deployment of the QoS-enabled Discovery Component in a Previous WSMX Installation

Let us assume that the WSMX framework has been installed into the folder `C:\WSMX`.

The deployment of the QoS-enable Discovery Component into the existing WSMX is as follows:

- Copy the file *qosdisc.wsmx* and file *qosdisc.properties* into the WSMX directory `C:\WSMX`.
- Configure the file *qosdisc.properties* to suit your needs as instructed in Section 2.2.
- Start WSMX, the new component will be added to the WSMX framework and initialized automatically.

3 Installation Instructions for UNIX-based Systems

The installation instruction for UNIX-like systems is mostly the same as in Section 2. The main difference is that one should change the permission of the file `./qosdisc/run.bat` to "executable" appropriately before running it.

4 Running the QoS-enabled Discovery Component

4.1 Running the QoS-enabled Discovery Component in Stand-alone Mode

After the installation and deployment of the QoS-enabled discovery component as described in Sections 2 and 3, a user can run and test the component in the stand-alone mode by:

- configuring the *qosdisc.properties* to suit his/her needs, e.g., specify the goal and the list of Web service descriptions you are going to work with;
- opening a DOS (or UNIX) console and run the file *run.bat*;
- the result of the QoS discovery process will be displayed in the console (by default) or written into a log file according to the user's configuration of the logger in the *qosdisc.properties* file.

4.2 Running the QoS-enabled Discovery Component via the WSMX Web-interface

To run the QoS discovery component within the WSMX framework, a user needs to do the following steps:

- configure the *qosdisc.properties* to suit our needs, e.g., specify the goal and the list of Web service descriptions he/she is going to work with;
- open a console and start WSMX as specified in the WSMX installation instruction. The WSMX framework has an embedded Web server which will be started automatically as well.
- Open the Web browser and go to the default server view of WSMX at `http://localhost:8080/serverbydomain`. The QoS discovery component is plugged-in as a WSMX component with name *components:name=QoSDiscovery*.
- Click on the link to the QoS discovery component, which should be something like `http://localhost:8080/mbean?objectname=components\%3Aname\%3DQoSDiscovery`.
- Click the *Invoke* button to run the discovery process.
- The result of the QoS discovery process will be displayed in the console where the WSMX framework was started (by default) or written into a log file according to the user's configuration of the loggers in the *qosdisc.properties* file.

4.3 Testing the QoS Discovery Component with Local Copies of Web Service and Goal Descriptions

To perform a full-fledged test of the QoS discovery component, we can write our own Web service and Goal descriptions, put them somewhere on the Web and ask the QoS discovery component to do the discovery job. Another simplified procedure is:

- copy an example service description and its QoS specification to the local directory, for example, copy *WSVariationDependentActionUC1.wsml* and *SWSQoS_UC1.wsml* from their original place `http://lsirpeople.epfl.ch/lhvu/ontologies/WP10/` and put them in a local directory *C:\TestOntologies*;
- copy an example WSMO goal description and its QoS specification to a local directory, for example, copy *GoalVariationDependentActionUC1.wsml* and *GoalQoS_UC1.wsml* from `http://lsirpeople.epfl.ch/lhvu/ontologies/WP10/` to the local directory *C:\TestOntologies*;
- change the QoS specification of the Web service *WSVariationDependentActionUC1.wsml* by modifying the local copy of the file *SWSQoS_UC1.wsml*;
- change the QoS requirements in the user's goal *GoalVariationDependentActionUC1.wsml* by modifying the local copy of the file *GoalQoS_UC1.wsml*;
- configure the *qosdisc.properties* file to load the local and modified version of the Web service description and goals with the following property setting:
  service1=`file:///C:/TestOntologies/WSVariationDependentActionUC1.wsml`
  goal=`file:///C:/TestOntologies/GoalVariationDependentActionUC1.wsml`
  subst1=`http://lsirpeople.epfl.ch/lhvu/ontologies/WP10->`
  `file:///C:/TestOntologies`;
- comment out the other properties (e.g., *service2, service3*, etc) referring to the ontologies having `http://lsirpeople.epfl.ch/lhvu/ontologies/WP10` as part of their URIs. Alternatively, we can download these ontologies and store them locally in the same directory *C:\TestOntologies*;
- run the QoS discovery component as described in Section 4.1 and Section 4.2. The QoS discovery component will then perform the discovery process taking the inputs from the goal and the Web service described in the local files `GoalVariationDependentActionUC1.wsml` and `WSVariationDependentActionUC1.wsml`.

Note that if a user calls the discovery component via the WSMX Web interface as in Section 4.2, after the modification of the WSMO description files, he/she will have to restart the WSMX framework to reload the modified ontologies since WSMX caches everything in its resource manager and there is no way to refresh these caches from external.

4.4 Developer's Guide to Interface with the QoS-enabled Discovery Component

For developers who want to interface with the QoS-enabled discovery component themselves, the following example code is provided:

- *TestStandalone.java*: Illustrates how to interface with the discovery component in a stand-alone fashion. This may be of interest for the users who want to integrate the QoS-enabled discovery component with their application, for example, WSMO Studio.
- *TestWSMX.java*: Shows how to call the discovery component via the WSMX entry point.

The above files are available at `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/example/TestWSMX.java` and `http://lsirpeople.epfl.ch/lhvu/download/qosdisc/example/TestStandalone.java`, respectively.

5 Implemented Features of This Prototype Release

In this release, we have implemented the core functionalities of the QoS discovery component:

1. The QoS discovery component can select QoS-annotated Web services descriptions that match the WSMO goals given the QoS requirements and classify the result according to a ranking algorithm

2. The main class *ch.epfl.qosdisc.wsmx.QoSDiscovery* of our discovery component implements the standard *org.wsmo.execution.common.component.Discovery* interface specified by the latest release of the WSMX API

3. The QoS matching and ranking algorithms use information from the developed ontologies, utilizing the KAON2-based WSML reasoner as its underlying inference engine

4. The WSMX resource manager is re-used as an internal repository to store QoS-annotated web services descriptions and goals

5. We already integrated the first implementation of the functionality-based service discovery component into our framework.

This first prototype release also includes:

1. A list of developed ontologies tested successfully with the WSML-tools and reasoner. Among them are the QoS and ranking ontologies: upper ontologies and the derived ones for the example application of WP10

2. A list of WSMO goals and Web service descriptions given the input from the use case partners (so far only WP10 has provided it)

3. The implementation of the core operators: Bloom filter, Matchmaking, Ranking, and Reputation management operators.

In the first prototype, only the Matchmaking and Ranking operators are integrated in the system. The next step would be to integrate all (already implemented) operators to the database query processing system (CoDIMSD) and provide a user interface to demonstrate the whole discovery process.

6 Known Issues

The following issues are left open at the time of writing this document:

- If the QoS-enabled discovery component is used with WSMX as in Section 4.2, then changing WSMO definitions will require to restart WSMX as WSMX caches everything and currently does not provide a method for notifying it of the change. However, this is a minor problem which will be fixed in future WSMX versions.
- If a user would like to use the current functionality discovery component which is integrated in our QoS component, he or she should adhere to some restrictions which cannot be influenced by the QoS discovery component described in this document. The input WSMO service description and goal to be used should comprise only post-conditions and effects (and not preconditions or assumptions) in their capabilities. This is due to the fact that the current lightweight functionality discovery engine only considers the outcome of a service execution, and not the pre-state and post-state. Attentive readers can refer to the documentation page of the functionality discovery component at `http://wiki.wsmx.org/index.php?title=Discovery_Tutorial` for more detailed instructions.
- The current functionality discovery component we are using (dated August 09th 2006) still has some stability issues: the invocation of that component occasionally produces no effect. We are collaborating with the developers of the functional discovery component to help them identify the problem.