A Web Registry for Publishing and Discovering Mathematical Services

Rebhi Baraka*, Olga Caprotti, and Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, Linz, Austria
{rbaraka,ocaprott,schreine}@risc.uni-linz.ac.at

Abstract

This paper describes an extension of the ebXML registry for publishing and discovering mathematical service descriptions. The MathBroker registry is able to handle descriptions given in the Mathematical Service Description Language, a language designed ad-hoc for capturing the semantics of web services dealing with mathematical problems. The registry is distributed with a Java API implementing a MathBroker speci£c JAXR provider.

1. Introduction

A web service is a problem solution that is available on the web and can be accessed by a user or another service or program via standard web protocols [13]. A mathematical web service is a service that offers the solution to a mathematical problem (based on e.g. a computer algebra system or on an automated theorem prover). Among the requirements of web services is the need of them to be advertised by providers and discovered by clients; therefore they need to be described in a machine-understandable format. In the case of mathematical web services, these descriptions must be based on formal mathematics.

Among the approaches for achieving this goal, MONET [11] and our own MathBroker project [9] have been pursued simultaneously with mutual in¤uence and have speci£cally concentrated on handling mathematical knowledge. As with conventional web services, a web registry provides a set of functionalities to facilitate the sharing and exchange of (mathematical) service descriptions. For this purpose, we developed the Mathematical Services Description Language (MSDL) [2] and extended the ebXML registry reference implementation [6] to handle the publication and discovery of objects that are presented in MSDL descriptions [4, 3].

* This work was sponsored by the FWF Project P17643-NO4 "Mathbroker II: Brokering of Distributed Mathematical Services".

The remainder of this paper is organized as follows: Section 2 introduces our model for describing mathematical services; this represents the basis for the design of MSDL and for an extension of the information model of a widely used registry framework. Section 3 describes this extension in greater detail. In Section 4, we present a sample client for using the extended framework for publishing MSDL descriptions and discovering such descriptions.

2. A Model for Mathematical Service Descriptions

Figure 1 illustrates the MathBroker information model for the description of mathematical web services. It shows the kinds of entities that can be associated to a service and the relationships among them.

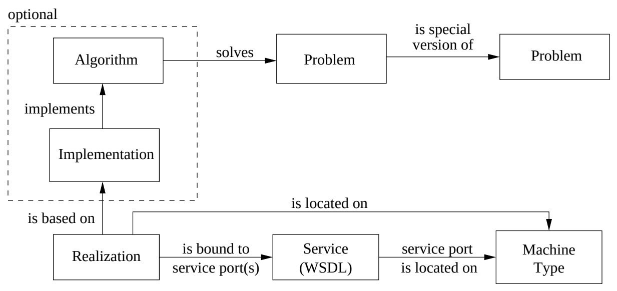

Figure 1. The MathBroker Information Model

The information model is implemented as a highly structured language called Mathematical Services Description Language (MSDL) [2]. MSDL was developed in the frame of MathBroker project [9] with in¤uences from the MONET project [11].

The rationale behind the decomposition of descriptions into multiple interlinked entities is to avoid redundancy between speci£cations (by sharing description components) and to provide a quick shortcut for detecting the identity of speci£cation entities by reference equality. MSDL thus provides for the development of a reusable library of descriptions.

Now we introduce the entities of the model.

Problem: There exist various kinds of problems. For instance, a computing problem can be speci£ed by input parameters, an input condition, output parameters, and an output condition. It can be a special version of another problem. A problem description for the inde£nite integration may contain the following mathematical knowledge (represented in XML using OpenMath):

Input: $f : \mathbb{R} \rightarrow \mathbb{R}$
Output: $i : \mathbb{R} \rightarrow \mathbb{R}$
Post-condition: $i' = f$

Algorithm: an algorithm is described by (a link to the description of) the problem it solves, as well as by time and memory complexity, termination conditions, and bibliographical information. Bibliography may be given by Dublin Core Metadata.

Implementation: an implementation is described by referring to the algorithm on which it is based (or optionally the problem it solves), the software, programming language, and numerical libraries that are used, and optionally time and memory ef£ciency w.r.t. some reference architecture.

Realization: a realization of a service is the description that brings together the abstract speci£cation of the service functionality with the actual details of the interface. Hence, it contains both a reference to either the underlying software implementation, or to the algorithm or the problem, and to a WSDL description of the service interface. It may also specify additional information on the hardware on which the service is running.

3. The MathBroker Registry

In web service technology, a registry is a software application for publishing and discovering information about web services. A registry maintains web service metadata as objects in a repository and provides access to them via a speci£c protocol. Currently there are two dominating registry standards: the Universal Description, Discovery, and Integration (UDDI) registry [12]; and the ebXML registry and repository standard [7].

We based our development on the ebXML registry reference implementation [6] because its information model [5] is much more generic and extensible than UDDI. Furthermore, this model closely follows Sun's Java API for XML registries (JAXR) [8] which provides generic access a variety of XML registries.

3.1. The ebXML Registry

The ebXML registry architecture consists of a registry service and a registry client. The registry service manages the objects associated with the registry and the queries for them. A registry client is an application that accesses the registry. It utilizes the registry service to submit objects, to classify them, to associate them to each other, to browse them, and to query for them.

The registry information model [5] represents a blueprint for the registry. It provides the information on the classes of metadata that are stored in the registry as well as the relationships among metadata classes. It de£nes what types of objects are stored in the registry and how they are organized. The information model is extensible by new kinds of objects.

3.2. Extending the ebXML Registry to MSDL Descriptions

In order to to capture the MathBroker mathematical service description model given in Figure 1, it was necessary to extend the information model of ebXML. Figure 2 shows the inheritance view of the MathBroker objects and their relationships inside the ebXML information model.

A generic `MathBrokerObject` class was introduced as an extension of the ebXML class `ExtrinsicObject`. From this class, all other MathBroker classes are inherited such that ebXML treats the MathBroker objects as instances of `ExtrinsicObject`.

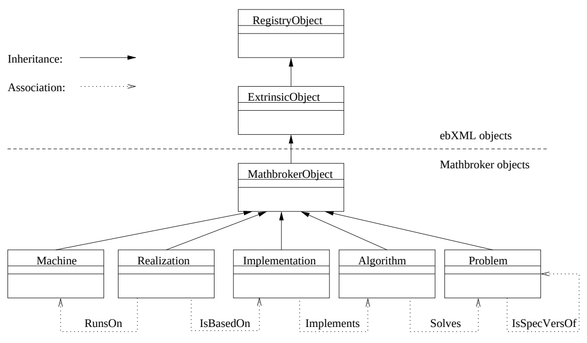

Figure 2. The MathBroker Inheritance Hierarchy

The rationale for using `ExtrinsicObject` as the basis of our classes is that it allows to hold metadata about which the registry has no prior knowledge. In particular, it may hold an XML document, e.g., the MSDL description of the entity. The MathBroker subclasses provide additional methods to extract and to modify this information.

3.3. Association of Entities

An essential characteristic of the MathBroker service description model is the ability to express relationships among the various entities that comprise a service. The registry facilitates this feature by the concept of an `Association`. A registry object may be associated with zero or more other registry objects. The service description model de£nes the following mathematical associations (see Figure 2):

- `IsSpecialVersionOf`: Problem P "is special version of" Problem P′.
- `Solves`: Algorithm A "solves" Problem P.
- `Implements`: Implementation I "implements" Algorithm A.
- `IsBasedOn`: Realization R "is based on" Implementation I.
- `RunsOn`: Implementation I "runs on" Machine M.

Associations are themselves registry objects and correspondingly stored in the registry with links to a source and a target registry object.

3.4. Classi£cation of Entities

A classi£cation scheme (or taxonomy) is a hierarchical tree of concepts that structures a particular knowledge area. The ability to classify an object (i.e., to link it to a concept in a classi£cation scheme) is an important feature of a registry, because it facilitates the process of discovering the object. An object in the registry may be classi£ed multiple times in one or in multiple schemes.

ebXML allows to submit new classi£cation schemes into the registry such that registry objects may be classi£ed in these schemes. We used this feature to import as an example the GAMS (Guide to Available Mathematical Software) classi£cation scheme [1]. MSDL entities may be classi£ed in this and in other mathematical schemes.

3.5. Using the Entities

Since the MathBroker mathematical entities are (extensions of) ebXML entities, they can be accessed using the standard ebXML mechanisms. For instance, Figure 3 demonstrates how the ebXML browser displays sample entities (of type Service, Implementation, Problem, Algorithm, and Machine) with their names drawn in rectangular boxes and the relations among them illustrated as arrows.

3.6. Implementation Aspects

We implemented the entities of the service description model such that this implementation captures all the aspects and features of MSDL. Moreover the MSDL entity classes inherit all the functionality of the ebXML class "ExtrinsicObject" they extend. We also extended the management functionality of the ebXML registry to allow the registration, classi£cation, association, and discovery of MSDL descriptions. The result of this implementation is a Java API for MSDL registries [10].

4. Publishing and Querying Service Descriptions

The use of the registry API is best demonstrated by a client that performs two tasks: publishing, i.e., registering service descriptions, and querying, i.e., discovering them.

4.1. Publishing to the Registry

The client publishes a description to the MathBroker registry by performing the following steps: (1) open a connection to the registry, (2) invoke the `publishMathBrokerObject` method, provided by the `MathBrokerLifeCycleManager`. This method takes a £le containing an MSDL description of a service and extracts the information of each entity it contains. For each entity description, it creates a registry object embedding that description and also creates/updates all required associations and classi£cations already in the registry.

4.2. Querying the Registry

The client queries the MathBroker registry by performing the following steps: (1) open a connection to the registry and (2) invoke methods provided by the `MathBrokerQueryManager` to make queries for mathematical objects according to ID, name, or classi£cation.

5. Conclusion

We presented £rst results on the development of a registry where the descriptions of mathematical services are published and can be discovered by potential clients. Our results demonstrate that standards and technologies that were originally developed for facilitating electronic business can be successfully used in a completely different (and considerably more sophisticated) application area, namely computer mathematics. Thus we pro£t from the work in the web community, preserve compatibility with its standards, and build on its software.

This framework serves as the basis for our ultimate goal of developing a "semantic broker" where services register

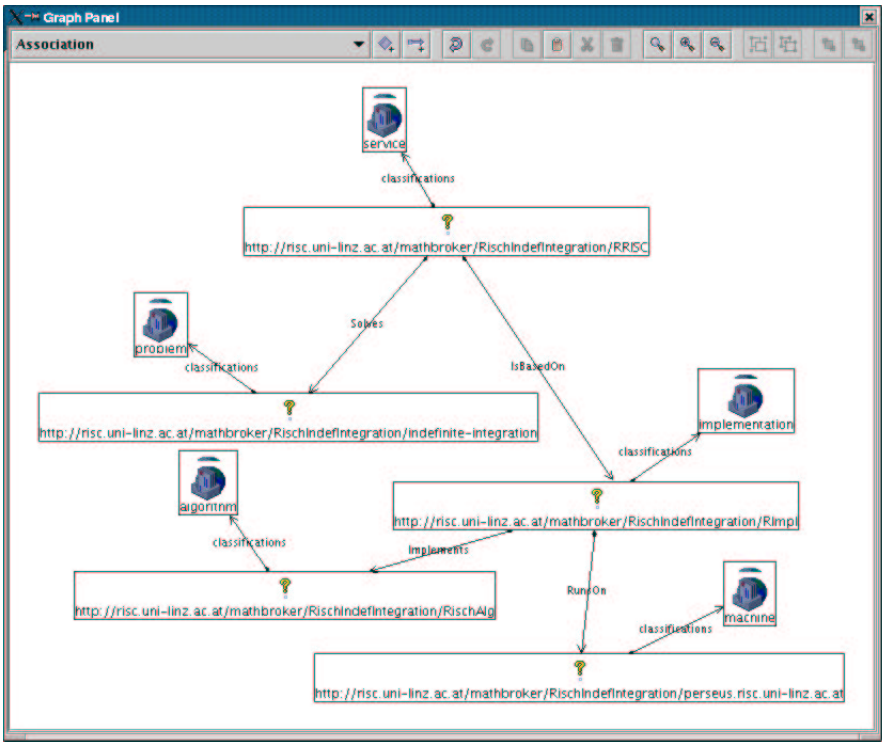
Graph Panel
Association
service
classifications
http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RRISC
problem
Solves
classifications
IsBasedOn
implementation
http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/indefinite-integration
classifications
algorithm
http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RImpl
classifications
Implements
RunsOn
machine
http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RischAlg
classifications
http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/perseus.risc.uni-linz.ac.at

Figure 3. Registry browser with MathBroker entities, their classi£cations and associations

their problem solving capabilities, clients submit task descriptions, and a broker then determines the suitable services and returns them to the client for invocation. Our next steps will be the design of a more expressive query model based on the syntactic and semantic content of the registered descriptions and of a corresponding query language that allows clients to discover suitable services.

References

[1] R. F. Boisvert, S.E. Howe, and D. K. Kahaner. GAMS: A Framework for the Management of Scienti£c Software. ACM Transactions on Mathematical Software, 11(4): 313–355, December 1985.

[2] O. Caprotti. Extending MONET to the MathBroker Information Model. Project Report, RISC-Linz, Johannes Kepler University, Linz, Austria, June 2003.

[3] M. Dewar, D. Carlisle, O. Caprotti. Description Schemes For Mathematical Web Services. Proceedings of EuroWeb 2002 Conference: The Web and the GRID: from e-science to e-business. St Anne's College Oxford, UK, December 2002. British Computer Society Electronic Workshops in Computing (eWiC).

[4] O. Caprotti and W. Schreiner. Towards a Mathematical Services Description Language. ICMS2002 , International Congress of Mathematical Software, Beijing, China, August 17-19, 2002.

[5] ebXML Registry Information Model v2.0. OASIS, December 2001. `http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrim.pdf`

[6] ebXML Registry Reference Implementation Project (ebxmlrr). OASIS, April 2004. `http://ebxmlrr.sourceforge.net/`

[7] ebXML Registry Services Speci£cation v2.0, OASIS, April 2002. `http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf`

[8] Java API for XML Registries (JAXR). Sun Microsystems, April 2004. `http://java.sun.com/xml/jaxr/`

[9] MathBroker: A Framework for Brokering Distributed Mathematical Services. Research Institute for Symbolic Computation, December 2001. `http://poseidon.risc.uni-linz.ac.at:8080/mathbroker/index.xml`

[10] MathBroker Registry API. Research Institute for Symbolic Computation, April 2004. `http://poseidon.risc.uni-linz.ac.at:8080/results/registry/MBregistryAPI`

[11] MONET: Mathematics on the Net. The MONET Consortium, April 2004. `http://monet.nag.co.uk/`

[12] UDDI Version 2.04 API Speci£cation. OASIS, July 2002. `http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf`

[13] Web Services Activity. World Wide Web Consortium, March 2004. `http://www.w3.org/2002/ws.`