A Registry Service as a Foundation for Brokering Mathematical Services

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

February 2004

Abstract

In this report, we present our results of developing a framework for publishing and discovering mathematical service descriptions in a registry. The registry is based on and extends the ebXML registry to handle mathematical service descriptions based on the MSDL specification developed in the framework of the MathBroker project. This work serves our ultimate goal to produce a "semantic broker" where services register their problem solving capabilities and clients submit task descriptions; the broker then determines the suitable services and returns them to the client for invocation.

Contents

*This work was sponsored by the FWF Project P15183 "A Framework for Brokering Distributed Mathematical Services".

1 Introduction

Interest from the mathematical community in using the Internet and the Web to facilitate the use of mathematics has paved the way for the emergence of mathematical web services. A mathematical web service can be defined in line with the definition of a web service as a description of a solution to a mathematical problem that is available on the Web and can be accessed by a user or another service or program. This description may contain information related to algorithm(s) used to solve the problem, type of problem, related problems, machines executing the problem, etc.

Mathematical web services need to be advertised by developers and discovered by users. There have been several approaches to achieving this goal. MONET [9] investigates how service discovery can be performed for mathematical Web services. It presents the following process of discovering and then invoking a service:

- Registration: The services register their capabilities, access policies, etc., with the broker's service manager.
- Inquiry: The client sends a description of the kind of service it is looking for to the broker. This description may be generic (e.g., something like "find me a service that performs definite integration"), or specific (e.g., "find me a service to solve the problem").
- Analysis: The planning manager inside the broker analyzes the problem and extracts the criteria on which to select a service, which it then matches against the registry maintained by the service manager. If it finds one or more possible matches there, details are returned to the client along with an indication of how closely they fit its requirements.
- Selection: The client selects a suitable service and requests access to it via the broker's service manager. What this entails depends on the access policies of the service and the particular service infrastructure being used. In the case of grid services, for example, where every abstract service is actually a factory, a new service instance would be generated and a handle returned to the client.
- Connection: If access is granted, then the client initiates a connection to the service.

A major goal of the Semantic Web [15] is to provide service discovery. Service descriptions is presented in formal languages or ontologies whose meaning is well defined and unambiguous. Given a set of service descriptions and a job specification written using a suitable collection of ontologies, a software agent takes care of selecting the appropriate service for the job and facilitates the interaction between the client and the service.

In the MathBroker project [8] our approach was to accomplish the goal of publishing and discovering such services using a registry. A (mathematical) registry provides a set of functionalities to facilitate the sharing and exchange of (mathematical) service descriptions. Figure 1 shows how a mathematical registry would be used for the sharing of mathematical web service information between interested parties.

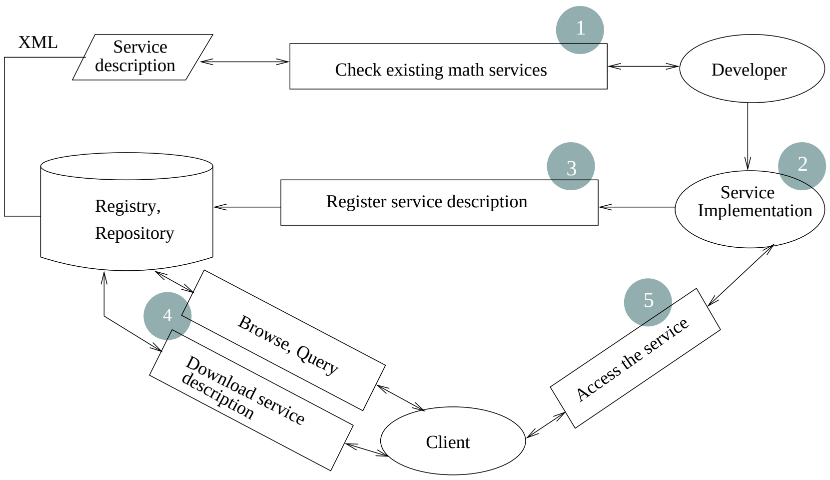

Figure 1: Registry use case scenario (based on [6])

1. A Mathematical web service developer checks existing services to see if his/her problem is already implemented.
2. If the service is not implemented, or if the developer is not satisfied by existing implementations, she implements the problem as a web service and deploys it in the web.
3. The developer registers the service together with some descriptive information.
4. Clients browse and query the registry for their desired services and download relevant descriptions.
5. Clients then access the service.

Taking into account the special nature of mathematical objects and the complications of transforming them into web services (see MathBroker sample services [14]), their registry descriptions must be based on a method that combines their formalism with the necessity for their web accessibility. We developed the Mathematical Services Description Language MSDL [1] as the method for representing service descriptions. We extended an existing registry implementation to handle the publication and discovery of objects that are based on MSDL descriptions.

The remainder of this report is organized as follows: Section 2 describes the architecture of our mathematical information model which serves as the basis for the design of MSDL and in turn as the basis for functionality which extends the basic registry information model. Section 3 deals with the ebXML registry and its information model which is the basis of our registry development. Section 4 presents the registry extension we have developed, the information model in context of the registry and the overall architecture of the registry. In Section 5, we present some examples of using the developed framework for publishing MSDL descriptions and discovering such descriptions.

2 Information Model

The MathBroker Information Model for descriptions of mathematical web services is based on a modular view of the various facets of a mathematical service. Thus, a service comprises descriptions of the communication layer (`service` and `realization`), of the implementation layer (`implementation` and `algorithm`) and of its abstract functionality (`problem`). The rationale behind the decomposition of descriptions into multiple interlinked components is to avoid redundancy between specifications (by sharing description components) and to provide a quick shortcut for detecting the identity of specification components by reference equality.

2.1 Architecture

Figure 2 shows the kind of metadata that can be associated to a service and the relationships between the various fragments of information. Some of the arrows symbolizing the references to problems, algorithms, or implementations are optional.

Every description is extensible, but the following information is recommended:

Problem: E.g. computing problems can be specified by input parameters, an input condition, output parameters, an output condition. Optionally, a problem can be declared as a special version of another problem (stronger input and/or weaker output condition);

Algorithm: An algorithm is described by (a link to the description of) the problem it solves plus time and memory complexity, termination conditions, etc.

Amplementation: An implementation is described by the algorithm on which it is based (or optionally the problem it solves) plus the software on which it is based plus time and memory efficiency w.r.t. some reference architecture;

Realization: A realization of a service is described by the underlying software implementation (or optionally the algorithm or problem), by the **type of machine** on which it is running and by a **WSDL** description of the service interface.

Machine type description specifies the underlying hardware specifications such as processor type, speed, memory size, etc. A **WSDL** description includes the syntactical interface of the service which might be availble at multiple ports.

The issues listed in Figure 2 are separate descriptions and represent important characteristics of the information model:

- Problems are organized in a hierarchy where generalizations and specializations are taken into account.
- Algorithms are in relation to the problems they solve. Several algorithm may solve the same problem.

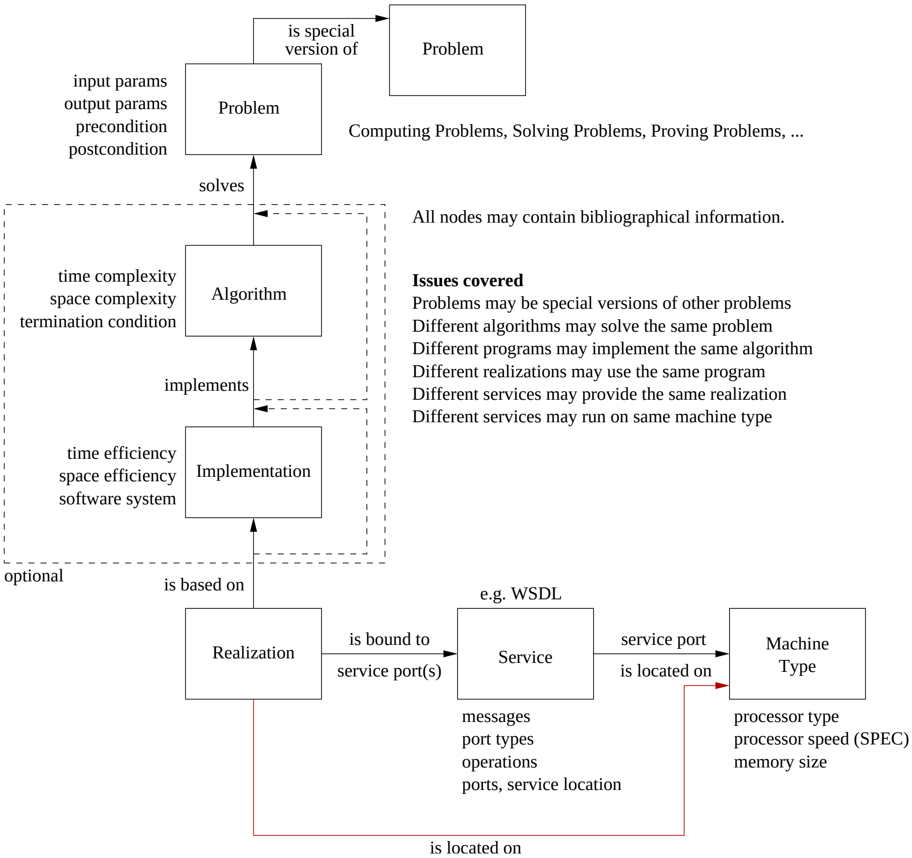

Figure 2: MathBroker Information Model

- Implementations are in relation to the algorithm they compute. Several programs may implement the same algorithm.

- Realizations are in relation to the implementation they are based on and to the service interface they are bound to. Additionally they might carry information about hardware details of the machine running the service. Several realizations may be based on the same implementation and several service interfaces may be bound to one realization.

2.2 Implementation

The information model is implemented as an XML based structured language called Mathematical Services Description Language MSDL [1, 2, 10]. Its grammar is defined by an XML schema [12]. Binding the schema using the Java Architecture for XML Binding (JAXB) [5] generated the MSDL library API [11]. The generated classes in the API represent the MathBroker schema. These classes define methods that are used later by the MathBroker registry provider to obtain and specify data for each type of element. A sample service description in MSDL is shown in Appendix A. It gives a description for each of the components presented above. A problem description for example is given from line 28 through line 73.

The next section explains the registry framework to which the above components are added.

3 ebXML Registry and Information Model

A registry is a web-based shared resource that enables the creation, deployment, and discovery of Web services. Web services shared information is maintained as objects in a repository and managed by the Registry Services defined as interfaces.

Currently there are two predominant specifications for a registry.

- The Universal Description, Discovery, and Integration (UDDI) registry [17].

- The ebXML Registry and Repository standard [4].

After some experimenting with the UDDI registry server provided as part of Sun's Java Web Services Developer Pack (JWSDP) [7], we decided to base our development on the OASIS ebXML registry server reference implementation [3] whose information model is much more generic and extensible than UDDI. Furthermore, ebXML information model closely follows Sun's Java API for XML registries (JAXR) [6] which provides a uniform access to different kinds of XML registries.

In the remaining of this section, we describe the ebXML registry functionality and information model. We introduce those components and entities that are most relevant to the MathBroker development. But first we briefly introduce JAXR API since the ebXML registry client-side implementation is based on it.

3.1 JAVA API for XML Registries (JAXR)

JAXR is designed with the intention of easy-to-use abstraction API to access a variety of XML registries. JAXR information model describes content and metadata within XML registries. Registry clients that are based on JAXR are meant to be portable across different target registries.

The high-level architecture of JAXR consists of the following parts:

- A JAXR client: a client program that uses the JAXR API to access a registry via a JAXR provider.

- A JAXR provider: an implementation of the JAXR API that provides access to a specific registry provider or to a class of registry providers that are based on a common specification.

JAXR specification [6] includes detailed bindings between the JAXR information model and the ebXML Registry.

3.2 ebXML Registry Architecture

The ebXML registry architecture, Figure 3, consists of an ebXML `Registry Service` and ebXML `Registry Client`. The ebXML `Registry Service` provides the ability for managing a repository. An ebXML `Registry Client` is an application used to access the registry.

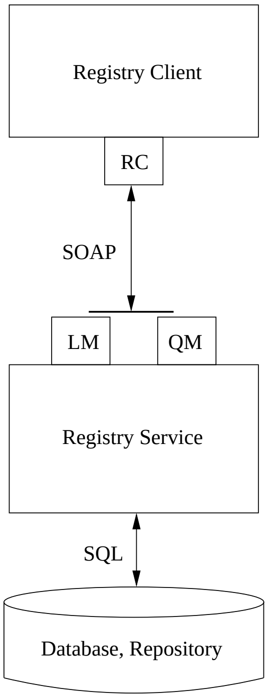

Figure 3: ebXML Registry Architecture

3.2.1 Registry Service

The ebXML `Registry Service` fundamentally manages objects and queries associated with the ebXML registry. The two primary interfaces for the `Registry Service` consist of:

- `A Lifecycle Management` (LM) interface that implements a collection of functionalities for managing objects within the registry.

- `A Query Management` (QM) interface that controls the discovery and retrieval of information from the registry.

3.2.2 Registry Client

A registry client program utilizes the services of the registry by invoking methods on one of the above interfaces. The client may use this interface to submit objects, to classify and associate objects, to remove objects, to browse objects, query for objects and their associated repository items.

The `Registry Client` (RC) interfaces may be local to the registry or local to the user. In the first case the registry provides a web based "thin client" application for accessing the registry that is available to the user using a common web browser. In this scenario the `Registry Client` interfaces reside across the Internet and are local to the registry from the user's view. In the second case the user uses a "fat client" registry browser application to access the registry. In this scenario the `Registry Client` interfaces reside within the registry browser tool and are local to the registry from the user's view. The `Registry Client` interfaces communicate with the registry over the Internet in this scenario.

3.3 ebXML Registry Information Model

The information model of the ebXML registry (see Figure 4) provides information on the type of metadata that is stored in the registry as well as the relationships among metadata classes. It defines what types of objects are stored in the registry and how these objects are organized in the registry.

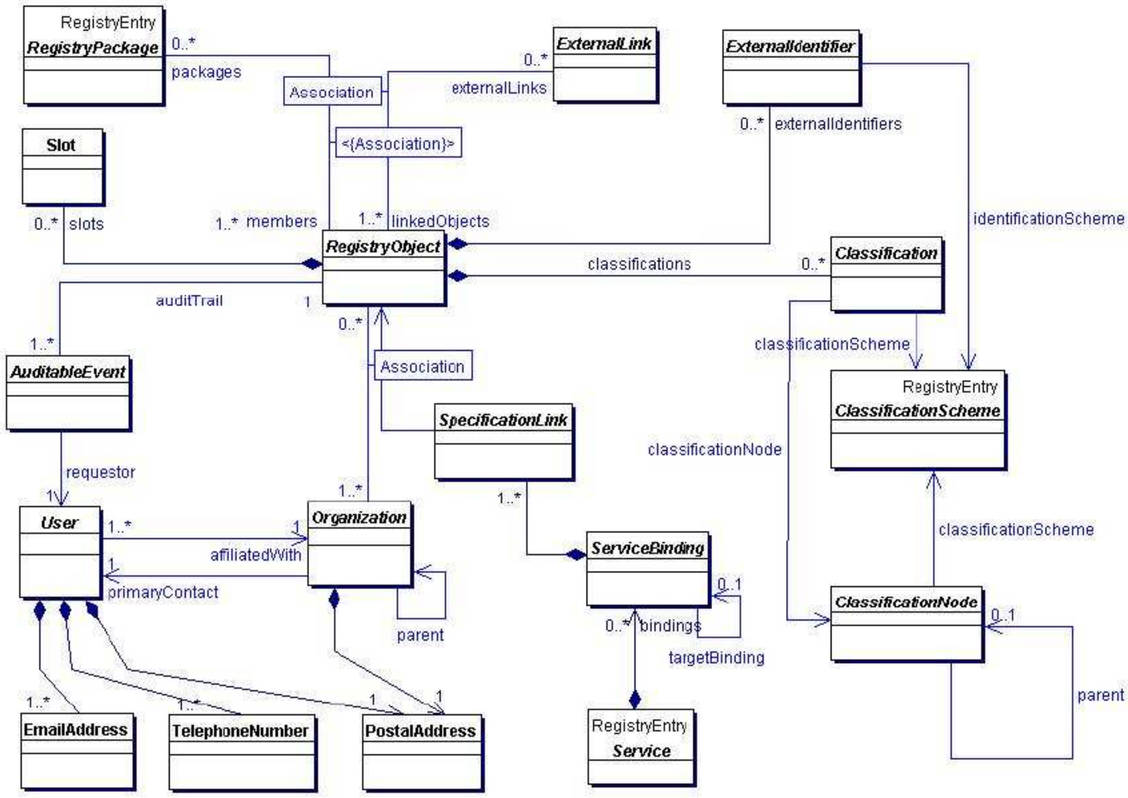

Figure 4: ebXML Information Model [3]

In our case, we used the ebXML information model to determine which components of it to include in MathBroker (registry) provider. The following entities are the most relevant ones and are introduced as they appear in [3].

RegistryObject: The `RegistryObject` class is an abstract base class used by most classes in the model. It provides minimal metadata for registry objects. It also provides methods for accessing related objects that provide additional dynamic metadata for the registry object.

Association: `Association` instances are `RegistryObject` instances that are used to define many-to-many associations between objects in the information model.

ClassificationScheme: `ClassificationScheme` instances are `RegistryEntry` instances that describe a structured way to classify or categorize `RegistryObject` instances. The structure of the classification scheme may be defined internal or external to the registry, resulting in a distinction between internal and external classification schemes.

Classification: `Classification` instances are `RegistryObject` instances that are used to classify other RegistryObject instances. A `Classification` instance identifies a `ClassificationScheme` instance and taxonomy value defined within the classification scheme. Classifications can be internal or external depending on whether the referenced classification scheme is internal or external.

ExtrinsicObject: `ExtrinsicObject` provides metadata that describe submitted content whose type is not intrinsically known to the registry and therefore MUST be described by means of additional attributes (e.g., MIME type). Examples of content described by `ExtrinsicObject` include Collaboration Protocol Profiles [ebCPP], Business Process descriptions, and XML schemas.

4 Extending ebXML Registry to MathBroker

In this section, we describe the extension we made to the ebXML registry information model to accommodate the MathBroker information model. This allows MSDL descriptions to be stored and retrieved using the basic ebXML mechanisms.

The information model of a registry defines the type of objects that reside in the registry and how they relate to each other. The MathBroker information model is added as an extension to the ebXML information model [3] to accommodate mathematical objects defined in MSDL (see Figure 2).

4.1 MathBroker Information Model and its Registry Implementation

The MathBroker information model in the context of the ebXML registry is shown in Figure 5. It shows basic entities and relationships among them to each other as associations; the term association is used in ebXML to relate two or more registry objects.

Next we explain the components of the MathBroker ebXML-extended information model.

The motive behind defining an `ExtrinsicObject` (explained above) in ebXML registry information model is to provide metadata for a repository item (e.g. WSDL document or an XML schema document) about which the registry has no prior knowledge. A mathematical object description in the form of MSDL fits under that definition. Therefore the decision was to define mathematical objects as extensions to the `ExtrinsicObject`.

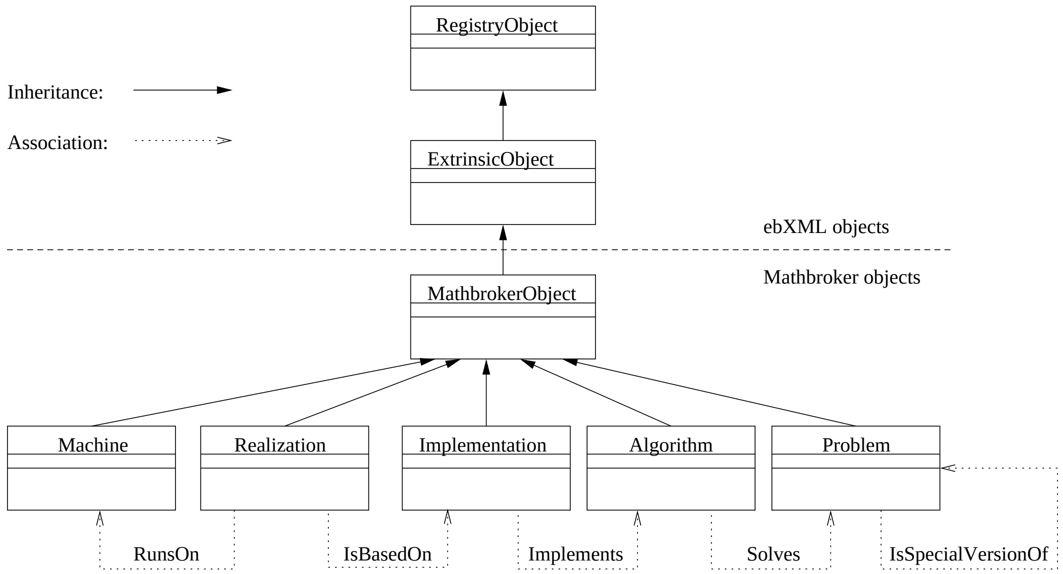

Figure 5: MathBroker Information Model in the registry context

MathBroker Object: This class is an extension of the ebXML class textttExtrinsicObject. It provides additional functionality to process a mathematical object description in the form of MSDL. For example extracting the different fields of the description of an object in order for instance to specify the kind of object or to show them to the user. This functionality as explained later is handled by the `MathBrokerLifecycle manager` which is part of the MathBroker registry provider in behalf of that object.

MathBroker Problem: This class provides information about a repository item (MSDL description) which contains the detailed specification of a problem such as input/output parameters and conditions, associations to other objects, etc.

MathBroker Algorithm: This class provides information about a repository item (MSDL description) which contains the detailed specification of an algorithm, efficiency factors, associations to other objects, e.g., link to the problem it solves, etc.

MathBroker Implementation: This class provides information about a repository item (MSDL description) which contains the detailed specification of an implementation such as the algorithm which implements, its time and memory efficiency, association(s), etc.

MathBroker Service: This class provides information about a repository item (MSDL description) which contains the detailed specification of a service such as its underlying implementation, the WSDL description of its interface, etc.

MathBroker Machine: This class provides information about a repository item (MSDL description) which contains the detailed specification of a machine such as its hardware characteristics.

The basic functionality of these components as registry objects is implemented as part of the MathBroker Registry API [13] which is shown in Appendix C. The rest of their functionality, such as get and set methods as defined by the MSDL schema are implemented in the MSDL Library API [11].

Publishing and discovering mathematical objects having the above types and also handling them through their Lifecycle is the responsibility of the MathBroker registry provider "the broker". See Section 4.4.4.

4.2 MathBroker Associations

An essential characteristic of the MathBroker information model is the ability of relating different mathematical objects that comprise a service to each other (see Section 2). The ebXML registry facilitates this feature by the concept of "association". A registry object maybe associated with zero or more registry object instances. The MathBroker information model (Figure 5) defines the following MathBroker associations:

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.

A more detailed view of MathBroker associations is shown in Figure 6.

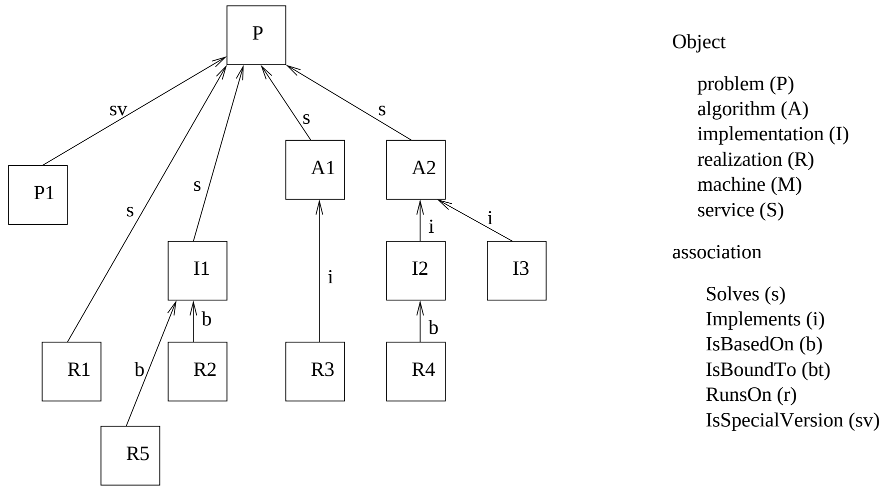

Figure 6: MathBroker associations

For every association there exists a source object and a target object. For example, in the association: Algorithm A "Solves" Problem P, the source object is Algorithm A and the target object is Problem P.

These associations are considered as registry objects and are defined to the registry like its predefined association types and therefore they are published under the ebXML registry's `AssociationType` classification scheme.

4.3 Classification of mathematical objects

The ability to classify objects is one of the main features of a registry. This is because classifications facilitate the process of discovering objects within the registry. An object in the registry may be classified along multiple classifications. A classification scheme in the registry represents taxonomies that provide values to classify objects. The ebXML registry provides the `SubmitObject` protocol to publish a classification scheme to the registry. Once a classification scheme is submitted/published to the registry, objects published can be classified against that scheme.

We were able to submit the GAMS (Guide to Available Mathematical Software) classification scheme to the ebXML registry. Figure 7 shows the scheme viewed in the registry browser.

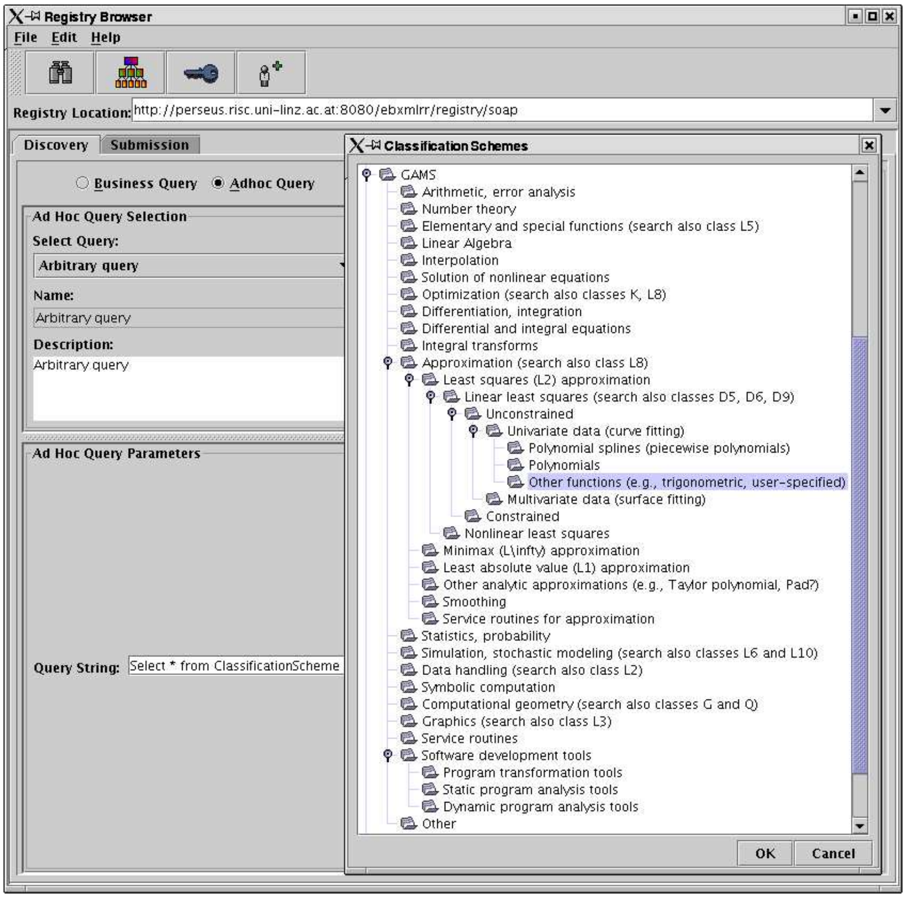

Figure 7: Registry browser screenshot of GAMS classification scheme

originally GAMS scheme was in a generic XML format as shown below:

```
<gamslist>
  <title>GAMS classification</title>
  <gamslist>
    <gamscode>A</gamscode>
    <title>Arithmetic, error analysis</title>
    <gamslist>
      <gamscode>A1</gamscode>
```

```
      <title>Integer</title>
    </gamslist>
    <gamslist>
      <gamscode>A2</gamscode>
      <title>Rational</title>
    </gamslist>
    <gamslist>
      <gamscode>A3</gamscode>
      <title>Real</title>
      <gamslist>
        <gamscode>A3a</gamscode>
        <title>Standard precision</title>
      </gamslist>
      <gamslist>
        <gamscode>A3c</gamscode>
        <title>Extended precision</title>
      </gamslist>
      .
      .
      .
</gamslist>
```

We wrote a stylesheet to transform it to the ebxml `SubmitObjectRequest` XML format and submitted it to the registry by means of the `SubmitObject` protocol mentioned above. The `SubmitObjectRequest` includes a `RegistryObjectList` which contains the `ClassificationScheme` object to be submitted a long with a number of `ClassificationNode`s preserving the parent-child relation determined by the id. For example the parent id is "urn:uuid:62aebe64-866a-4706-982f-eec59a32e7ad" and its name is "GAMS", while the id of its imeddiate child is "urn:uuid:62aebe64-866a-4706-982f-eec59a32e7ad/A" and the name is "Arithmetic, error analysis"

```
  <?xml version="1.0" encoding="UTF-8"?>
<rs:SubmitObjectsRequest
        xmlns = "urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.0"
        xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation = "urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.0
        http://www.oasis-open.org/committees/regrep/documents/2.0/schema/rim.xsd
        urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.0
        http://www.oasis-open.org/committees/regrep/documents/2.0/schema/rs.xsd"
        xmlns:rim = "urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.0"
        xmlns:rs = "urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.0">
 <LeafRegistryObjectList>
  <ClassificationScheme id="urn:uuid:62aebe64-866a-4706-982f-eec59a32e7ad"
    isInternal="true" nodeType="UniqueCode"
    xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.0">
    <Name>
        <LocalizedString charset="UTF-8" value="GAMS"/>
    </Name>
    <Description>
```

```
        <LocalizedString charset="UTF-8"
        value="This is the classification scheme for GAMS"/>
    </Description>
   <ClassificationNode id="urn:uuid:62aebe64-866a-4706-982f-eec59a32e7ad/A" code="A">
      <Name>
         <LocalizedString charset="UTF-8" value="Arithmetic, error analysis"/>
      </Name>
          .
          .
          .
   </ClassificationNode>
  </ClassificationScheme>
 </LeafRegistryObjectList>
</rs:SubmitObjectsRequest>
```

4.4 MathBroker Registry Architecture

The overall structure of the MathBroker registry extending the ebXML registry and incorporating the MSDL library is shown in Figure 8.

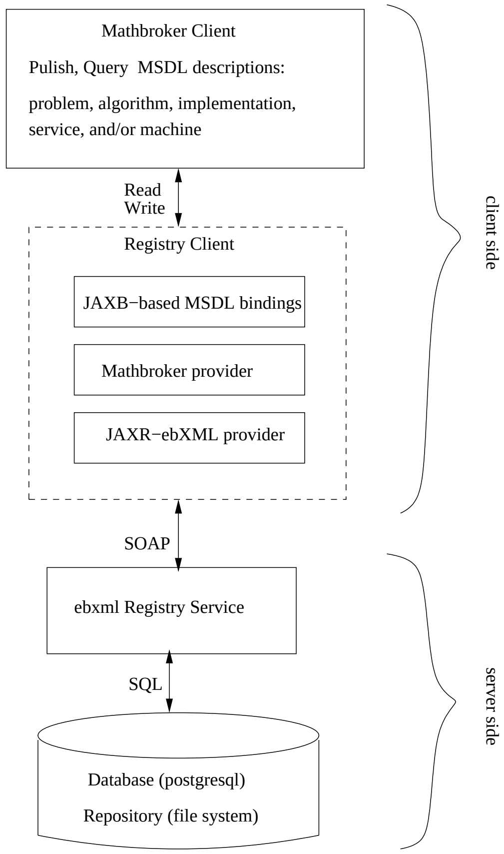

Figure 8: MathBroker Registry Architecture

4.4.1 Database and Repository System

At the bottom of the Figure is the rational database system where metadata of registry objects is stored. Metadata are defined attributes of a MathBroker registry object. Repository items in the form of MSDL descriptions associated with these objects are stored in the repository which is basically a file system managed by a server-side component called Repository Manager. Each repository item is assigned a UID by the Life Cycle Manager which servers as its name in the file system.

4.4.2 Registry Service

The server-side component managing the database and the repository is known as Registry Service in ebXML terminology (registry provider in JAXR terminology) and is deployed in our case to the Tomcat servlet container [16].

4.4.3 JAXR-ebxml Provider

The JAXR-ebxml Provider provides an implementation of the JAXR specification to access the ebXML registry provider (registry service in ebXML terminology). It accepts JAXR requests from the client and transforms them into equivalent requests based on the specification of the ebXML registry. From the registry service (provider) point of view, its client is the JAXR-ebXML provider.

4.4.4 MathBroker Provider: the "Broker"

The MathBroker Provider extends the JAXR-ebXML provider and implements features required to enable the client to manage MSDL based objects. Specifically it performs the publishing and discovering of these objects. It has the following management components:

MathBroker Life Cycle Manager

In order to publish Mathematical objects to the registry and to manage them through their life cycle in the registry taking into account their accompanying MSDL structures, we extended the `LifecycleManager` of ebXML registry. `MathBrokerLifeCycleManager` (See Appendix C) performs all management functionalities in the registry on behalf of MathBroker objects. When a service description (See Appendix A) is submitted to the registry, `MathBrokerLifeCycleManager` mainly performs the following:

- determines the type of MathBroker entity/entities contained in the description,
- for each entity, it creates a registry object and retrieves the coressponding metadata from the description to uniquely name the object in the registry (mathematical objects in the registry are given unique names aside from the unique ID given to them by the registry.) The unique name is formed by concatenating the enitiy's name to the target name space,
- for each entity, it extracts the coressponding description and stores it as a separate repository item "attached" to the created registry object,
- creates the required associations and classifications as specified in the description,
- saves each created object and its repository item,

- if an object already exists in the registry it updates it.

The `MathBrokerLifeCycleManager` uses the "get methods" of the MSDL Library API [11] to perform any of the above steps involving service descriptin. For example, to read the name of any entity in the description, it uses the `getName()` method.

MathBroker Query Manager

The `MathBrokerQueryManager` adds to the ebXML query manager the ability to query or search the registry for mathematical objects, fetches their respective MSDL descriptions from the repository and extracts the required fields from these descriptions. It can display these fields and save the whole description as an MSDL file. Inheriting the functionality of the ebXML Query manager, it can query for associations, an object's associated objects, classifications, etc. It performs queries according to object's ID, classification, or name.

The implementation of these two managers is the core of the MathBroker Registry API [13]. Appendix C shows the methods of these managers.

4.4.5 JAXB-based MSDL Bindings

The JAXB-based MSDL bindings (MSDL Library API [11]) provides a collection of content classes (with get and set methods) generated from a schema representing the MathBroker information model by means of JAXB (Java API for XML Bindings). The `MathBrokerLifeCycleManager` uses these classes and methods to specify the kind of content contained within an MSDL description and gets/sets for instance its name, namespace, service url, input/output parameters, etc.

4.4.6 MathBroker Client

It is a user program (command line program in our case and a registry browser in ebXML) that accesses the registry via the registry client. A client sample for publishing and querying the registry is shown in the next section.

5 Publishing and Querying in MathBroker Registry

We wrote a sample client that demonstrates the use of the MathBroker registry. The client performs two tasks: publishing, i.e., submitting service descriptions, and querying, i.e., discovering them. For publishing, the client takes an MSDL file and registers all entities described in it (also creating the related associations and classifications). For querying, the client takes a question from the user and prints the resulting MSDL descriptions.

5.1 A Sample Service Description

An MSDL description can contain one or more service descriptions entities. A complete service description would include all the entities introduced in Section 2 and pointers for entity associations and classifications. A complete service description is shown in Appendix A.

5.2 Publishing to the Registry

The publish interface the client is shown in Appendix B.1. It makes a connection to the registry using the URL of the registry, uses this connection to obtain the registry service, and utilizes the service by accessing the `MathBrokerLifecycle manager` and `MathBrokerQuery manager` to publish to the registry.

The client takes an MSDL description stored in a file (e.g. `risch.xml`) and extracts the description of each entity. For each description it creates a registry object embedding that description and also creates all required associations and classifications. All this functionality is hidden in the method `publishMathBrokerObject` of the MSDL registry API.

5.3 Querying the Registry

The query program shown in Appendix B.2 allows to make queries for mathematical objects according to ID, name, or classification by invoking the following methods of the registry API.

```
executeQueryById(argument)
executeQueryByName(argument)
executeQueryByClassification(argument)
```

Ultimately the `MathBrokerLifecyle manager` is invoked to retrieve a description from the repository, then extracts and displays the respective fields from this description using the get methods of the MSDL Library API [11].

The following example shows a query for an algorithm:

```
MathBrokerAlgorithm algorithm =
(MathBrokerAlgorithm)mlcm.createMathBrokerAlgorithm(
  null, null, classificationConcept, dh);
algorithm.showContent();
```

The `showContent()` method displays individual fields from the description of an entity. It also shows the whole description in MSDL indented properly on the screen of the user. The user can also use the ebXML registry browser to graphically view mathematical objects, their associations, and their classifications.

Using ebXML Registry Browser to View MathBroker Objects

Figure 9 shows some MathBroker objects namely Service, Implementation, Problem, Algorithm, and Machine with their names appear in the rectangular boxes. The Figure also shows associations among these objects. For example, Service *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RRISC* **Solves** Problem *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/indefinite-integration* and **IsBasedOn** Implementation *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RImpl* which **Implements** Algorithm *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/RischAlg* and **RunsOn** Machine *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/perseus.risc.uni-linz.ac.at*.

The name of an object is formed, when the object is read from its MSDL description, from its namespace and its element name as it appears in the description. For example, the namespace for Problem *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/indefinite-integration* is *http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration/* and its element name is *indefinite-integration*.

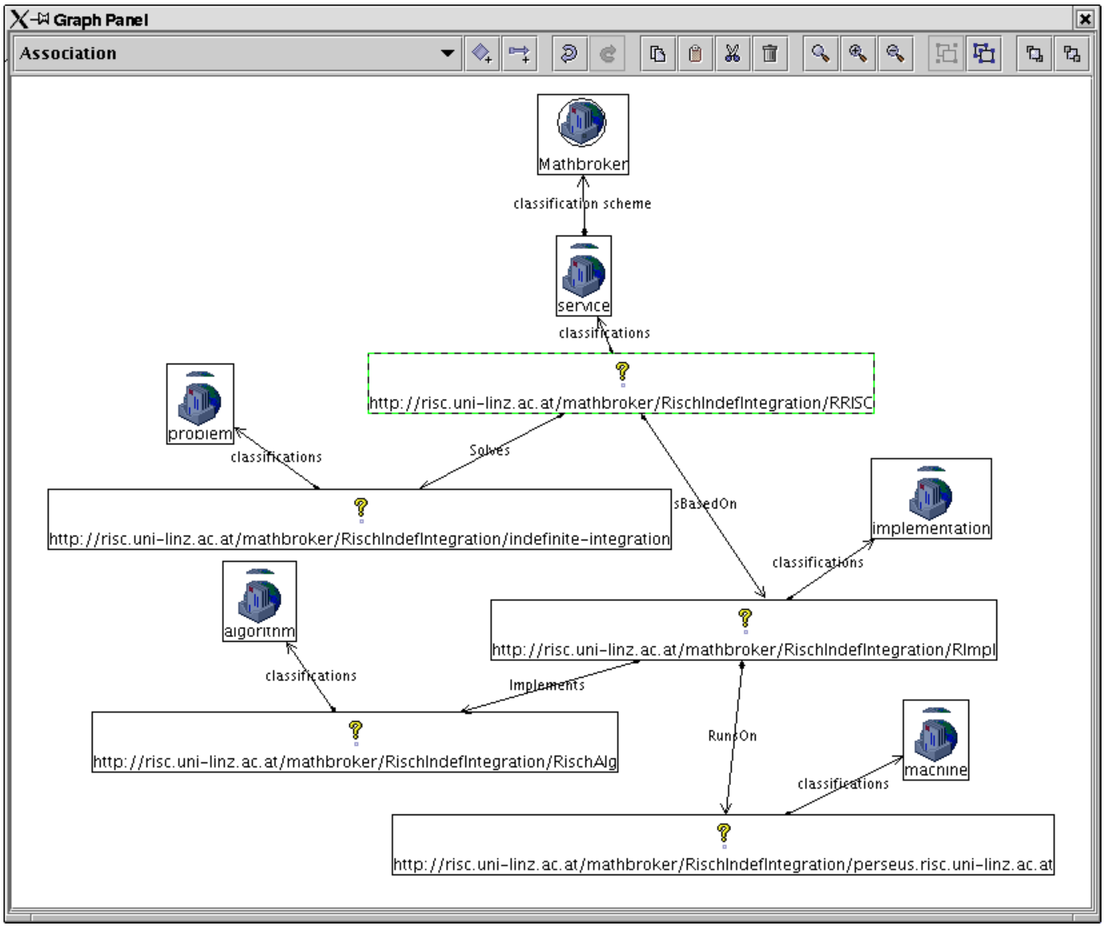

Figure 9: Registry browser screenshot of MathBroker objects with their classifications and associations

6 Conclusion

We presented our results on the development of a registry framework where descriptions of mathematical Web services are represented in a standard mathematical description language [1, 2], published in the registry, and discovered by registry clients. Our results demonstrate the fact that standards and technologies developed for a particular application area (such as ebusiness) can be used in a more sophisticated application area such as computer mathematics.

This framework serves as the foundation for our ultimate goal of developing a "semantic broker" where services register their problem solving capabilities, clients submit task descriptions, and the broker then determines the suitable services and returns them to the client for invocation. Our next step is to enhance the discovery mechanism of the registry by designing an MSDL query model and to develop a query language based on this model.

A A Sample MSDL Service Description (risch.xml)

```
<monet:definitions
 targetnamespace="http://risc.uni-linz.ac.at/mathbroker/RischIndefIntegration"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:mathb="http://risc.uni-linz.ac.at/mathbroker/ns"
 xmlns:monet="http://monet.nag.co.uk/monet/OpenMathDC"
 xmlns:om="http://www.openmath.org/OpenMath"
 xmlns:symbint="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicIntegration"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemalocation="http://monet.nag.co.uk/monet/OpenMathDC/
                 home/olga/cvs/perseus/monet-based-xsd/xsd/monetOM_DC.xsd">
  <!--  $Id: risch.xml,v 1.1 2004/04/23 10:49:52 rbaraka Exp $ -->
  <!-- actually these are just directives - at this stage no tool processes these -->

  <mathb:machine_hardware address="193.170.37.69" name="perseus.risc.uni-linz.ac.at">
    <mathb:CPU name="Intel Celeron"></mathb:CPU>
    <mathb:CPU_speed mhz="733"></mathb:CPU_speed>
    <mathb:RAMsize mb="256"></mathb:RAMsize>
    <mathb:disksize gb="40"></mathb:disksize>

    <mathb:OS href="http://www.suse.de"></mathb:OS>
  </mathb:machine_hardware>

  <monet:import location="./SymbolicIntegration.wsdl"
    namespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicIntegration">
  </monet:import>

  <monet:problem name="indefinite-integration">

    <monet:header></monet:header>
    <monet:body>
      <monet:input name="f">
        <monet:signature>

          <om:OMOBJ>
            <om:OMA>
              <om:OMS cd="sts" name="mapsto"></om:OMS>
              <om:OMS cd="setname1" name="R"></om:OMS>
              <om:OMS cd="setname1" name="R"></om:OMS>
              <om:OMS cd="setname1" name="R"></om:OMS>
```

```
          </om:OMA>
        </om:OMOBJ>
      </monet:signature>

    </monet:input>
    <monet:output name="i">
      <monet:signature>
        <om:OMOBJ>
          <om:OMA>
            <om:OMS cd="sts" name="mapsto"></om:OMS>
            <om:OMS cd="setname1" name="R"></om:OMS>
            <om:OMS cd="setname1" name="R"></om:OMS>
            <om:OMS cd="setname1" name="R"></om:OMS>

          </om:OMA>
        </om:OMOBJ>
      </monet:signature>
    </monet:output>
    <monet:post-condition>
      <om:OMOBJ>
        <om:OMA>
          <om:OMS cd="relation1" name="eq"></om:OMS>
          <om:OMV name="i"></om:OMV>

          <om:OMA>
            <om:OMS cd="calculus1" name="indefint"></om:OMS>
            <om:OMV name="f"></om:OMV>
          </om:OMA>
        </om:OMA>
      </om:OMOBJ>
    </monet:post-condition>
  </monet:body>
</monet:problem>

<monet:algorithm name="RischAlg">
  <monet:documentation>This is the metadata for the algorithm
    Risch. The namespace is the target namespace of this document.
    </monet:documentation>

  <monet:bibliography href="http://www.emis.de/cgi-bin/zmen/ZMATH/en/
                                  quick.html?type=xml&an=0184.06702">
    <!-- more dublin core -->
    <monet:documentation> Dublin Core Data </monet:documentation>
    <dc:creator>Risch,R.H.</dc:creator>

    <dc:title>The Problem of Integration in Finite Terms</dc:title>
    <dc:source> Trans. A.M.S. 139 pp.167 - 189</dc:source>
    <dc:publisher>AMS</dc:publisher>
    <dc:date>1969</dc:date>
  </monet:bibliography>
</monet:algorithm>

  <monet:implementation name="RImpl">
  <mathb:efficiency_factor wrt="S200Spec">
    <mathb:speed>1.1</mathb:speed>
    <mathb:throughput>0.7</mathb:throughput>
  </mathb:efficiency_factor>
    <monet:software href="http://www.wolfram.com"></monet:software>
```

```
      <monet:software href="http://riaca.win.tue.nl/software/ROML">
      </monet:software>
      <monet:hardware href="http://risc.uni-linz.ac.at/mathbroker/
                        RischIndefIntegration/perseus.risc.uni-linz.ac.at">
      </monet:hardware>
      <monet:algorithm href="http://risc.uni-linz.ac.at/
                        mathbroker/RischIndefIntegration/RischAlg">
      </monet:algorithm>
    </monet:implementation>

  <monet:service name="RRISC">
      <monet:documentation>This is an implementation of the algorithm
        Risch. We use the mathb namespace to state expected performance
        of the  concrete implementation wrt to its theoretical
        complexity measure.</monet:documentation>
    <monet:classification>

      <monet:problem href="http://risc.uni-linz.ac.at/mathbroker/
                       RischIndefIntegration/indefinite-integration">
      </monet:problem>
    </monet:classification>
    <monet:implementation href="http://risc.uni-linz.ac.at/mathbroker/
                            RischIndefIntegration/RImpl">
    </monet:implementation>
    <monet:service-interface-description
      href="http://perseus.risc.uni-linz.ac.at:8080/axis/
             services/SymbolicIntegration?wsdl">
    </monet:service-interface-description>
    <monet:service-binding>
      <monet:map action="exec" operation="symbint:Integrator:indefInt"
       problem-reference="indefinite-integration"></monet:map>
      <monet:message-construction io-ref="f"
      message-name="symbint:IndefIntRequest" message-part="in0">
      </monet:message-construction>
    </monet:service-binding>
    <monet:service-metadata></monet:service-metadata>

    <monet:broker-interface>
      <monet:service-URI></monet:service-URI>
    </monet:broker-interface>
  </monet:service>
</monet:definitions>
```

B Publish and Query Examples

B.1 Publish Example

```
import java.io.*;
import java.util.*;
import java.io.InputStream.*;
import java.net.PasswordAuthentication;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
```

```java
import java.util.Set;
import java.util.List;
import javax.activation.*;
import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.xml.parsers.*;
import javax.xml.registry.*;
import javax.xml.registry.infomodel.*;
import javax.xml.registry.BulkResponse;
import javax.xml.registry.ConnectionFactory;
import javax.xml.registry.infomodel.*;
import com.sun.xml.registry.ebxml.ConnectionFactoryImpl;
import at.ac.uni_linz.risc.mathbroker.registry.infomodel.*;
import at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import com.sun.msv.grammar.*;
import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import monet.openmath.lang.*;
import monet.openmath.lang.impl.*;
import org.openmath.lang.*;
import nl.tue.win.riaca.openmath.io.OMXMLReader;
import nl.tue.win.riaca.openmath.lang.OMObject;
import org.xml.sax.XMLReader;
import at.ac.uni_linz.risc.mathbroker.lang.*;
import at.ac.uni_linz.risc.mathbroker.lang.impl.*;
import at.ac.uni_linz.risc.mathbroker.lang.ObjectFactory;
import at.ac.uni_linz.risc.mathbroker.registry.infomodel.*;
import com.sun.xml.registry.ebxml.*;
import org.oasis.ebxml.registry.bindings.rim.ExtrinsicObjectType;

/**
 * The MathBrokerPublish class consists of a main method, a
 * makeConnection method, and a publish method.
 * It takes an xml file with mathbroker element defenitions,
 * extracts each element to a separate xml file, and
 * creates a Mathbroker object for it and loads the
 *  corresponding repository item file to the Mathbroker
 *  registry.
 */
public class MathBrokerPublish {
        MathBrokerRegistryService mrs = null;
        MathBrokerConnection connection=null;
        MathBrokerFocusedQueryManager fqm=null;
        MathBrokerLifeCycleManager mlcm=null;
        Marshaller marshaller =null;
        String uri=null;
        String username = "rbaraka"; //"Rebhi S Baraka";
```

```
        String password = "testuser1"; //"mathbroker";

        public MathBrokerPublish(){
        }

        public static void main( String[] args ) throws Exception {
                String queryPublishUrl =
                                "http://perseus.risc.uni-linz.ac.at:8080/ebxmlrr/registry/soap";
                MathBrokerPublish mathbrokerPublish = new MathBrokerPublish();
                mathbrokerPublish.makeConnection(queryPublishUrl);
                mathbrokerPublish.publish(args);
        }

        /**
         * Establishes a connection to a registry.
         * @param queryUrl        the URL of the query registry
         */
        public void makeConnection(String queryPublishUrl) {
         /*
                * Define connection configuration properties.
                */

                // To publish, you need both the query URL and the publish URL
                 Properties props = new Properties();
                 props.setProperty("javax.xml.registry.queryManagerURL", queryPublishUrl);
                 props.setProperty("javax.xml.registry.lifeCycleManagerURL", queryPublishUrl);
                 props.setProperty("javax.xml.registry.factoryClass",
                       "com.sun.xml.registry.ebxml.ConnectionFactoryImpl" );
                 try {

                  // Create the connection, passing it the configuration properties
                  ConnectionFactory factory = MathBrokerConnectionFactoryImpl.newInstance();
                  System.out.println("NewInstance is " + factory);
                  factory.setProperties( props );
                  MathBrokerConnection connection =
                                 (MathBrokerConnection)factory.createConnection();
                  System.out.println( "---Created connection to registry---" );
                  mrs = connection.getMathBrokerRegistryService();
                  mlcm = mrs.getMathBrokerLifeCycleManager();
                  fqm = mrs.getMathBrokerFocusedQueryManager();
                  System.out.println( "---Got Mathbroker registry service and  manager---" );
                  // Get authorization from the registry
                  PasswordAuthentication passwdAuth =
                         new PasswordAuthentication( username, password.toCharArray() );
                  Set creds = new HashSet();
                  creds.add( passwdAuth );
                  connection.setCredentials( creds );
                  System.out.println( "---Established security credentials---" );
                 }catch (Exception e) {
                                 e.printStackTrace();
                                 if (connection != null) {
                                                 try {
                                                                 connection.close();
                                                 } catch (JAXRException jaxre) {}
                                 }
                 }
        }

```

```
        public void publish(String [] args) {
         try {
                        String fileName = args[0];
                        //InputStreamReader input = new InputStreamReader(System.in);
                        //BufferedReader reader = new BufferedReader(input);
                        //System.out.println("Enter xml file name ");
                        //fileName = reader.readLine();
                        System.out.println("Filename reads: "+ fileName);
                        File repositoryItemFile = new File (fileName);
                        // File repositoryItemFile = new File ("risch.xml");
                        javax.activation.DataHandler repositoryItem =
                                        new DataHandler(new FileDataSource(repositoryItemFile));
                        mlcm.publishMathBrokerObject(fqm, repositoryItem);
         }catch (JAXBException je) {
                        je.printStackTrace();
         }catch (JAXRException jre) {
                        jre.printStackTrace();
         }catch (IOException ioe) {
                        ioe.printStackTrace();
         }
        }
}

```

B.2 Query Example

```
import javax.xml.registry.*;
import javax.xml.registry.infomodel.*;
import javax.activation.DataHandler;
import java.io.*;
import java.net.*;
import java.util.*;
import com.sun.xml.registry.ebxml.*;
import com.sun.xml.registry.ebxml.infomodel.*;
import at.ac.uni_linz.risc.mathbroker.registry.infomodel.*;
import at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.*;
import javax.xml.bind.JAXBContext;
import com.sun.msv.grammar.*;

/**
 * The MathBrokerQuery class consists of a main
 * method, a makeConnection method a getManagers method,
 * a makeSelection method,  and executeQuery method. It
 * searches a registry for information about
 * Mathbroker Object(s)
 *
 */

public class MathBrokerQuery {

                public MathBrokerRegistryService rs;
                public MathBrokerLifeCycleManager mlcm;
                public MathBrokerFocusedQueryManager fqm;
                MathBrokerDeclarativeQueryManager dqm;
                MathBrokerConnection connection=null;
                RegistryObject response=null;

  // constructor
```

```
    public MathBrokerQuery() {
    }

    public static void main(String[] args) {
                String queryURL =
                         "http://perseus.risc.uni-linz.ac.at:8080/ebxmlrr/registry/soap";
      MathBrokerQuery mbQuery = new MathBrokerQuery();
      mbQuery.makeConnection(queryURL);
                mbQuery.makeSelection();
          }

           /**
     * Establishes a connection to a registry.
     * @param queryUrl        the URL of the query registry
     */
    public void makeConnection(String queryUrl) {
      /*
       * Define connection configuration properties.
       */
                        Properties props = new Properties();
                        props.setProperty("javax.xml.registry.queryManagerURL", queryUrl);
                        props.setProperty("javax.xml.registry.factoryClass",
                        "com.sun.xml.registry.ebxml.ConnectionFactoryImpl");

          try {
                        // Create the connection, passing it the configuration properties
                        ConnectionFactory factory =
                                        MathBrokerConnectionFactoryImpl.newInstance();
                        factory.setProperties(props);
                        connection = (MathBrokerConnection)factory.createConnection();
                        System.out.println("---Created connection to registry---");
                        rs = connection.getMathBrokerRegistryService();
                        mlcm = rs.getMathBrokerLifeCycleManager();
                        fqm = rs.getMathBrokerFocusedQueryManager();
                        System.out.println( "--Got Mathbroker registry service and  managers--");
          } catch (Exception e) {
                        e.printStackTrace();
                        if (connection != null) {
                                        try {
                                                        connection.close();
                                        } catch (JAXRException jaxre) {}
                        }
          }
    }

          public void makeSelection() {
           try {

                String argument = null;
                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Enter selection: 1 to query by Id, 2 to query by Name,
                 or 3 to query by Classification ");
                String str =in.readLine();
                StringTokenizer st =new StringTokenizer(str);
                int selection=Integer.parseInt(st.nextToken());
                switch (selection){
                 case 1 : System.out.println("Enter id.");
                                 argument = in.readLine();
```

```
                                 fqm.executeQueryById(argument);
                                 break;
                 case 2 : System.out.println("Enter name. You may use % wildcard.");
                                 argument = in.readLine();
                                 fqm.executeQueryByName(argument);
                                 break;
                 case 3 : System.out.println("Enter classification. You may use % wildcard.");
                     argument = in.readLine();
                     fqm.executeQueryByClassification(argument);
                                 break;
                default: System.out.println("Selections are 1 to 3 only");
                }
         }catch (IOException e){
                         e.printStackTrace();
         }catch (JAXRException jaxre){
                         jaxre.printStackTrace();
         }catch (javax.xml.bind.JAXBException jaxbe){
                         jaxbe.printStackTrace();
         }
        }
}
```

C MathBroker Registry API

C.1 Interfaces

C.1.1 Interface **MathBrokerAlgorithm**

Declaration

public interface MathBrokerAlgorithm **implements** MathBrokerObject

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.1.2 Interface **MathBrokerConnection**

This class represents a connection between a JAXR client and a JAXR provider.

Declaration

public interface MathBrokerConnection

Methods

- *getMathBrokerRegistryService*
  `public MathBrokerRegistryService` **getMathBrokerRegistryService**`( )`
  - **Usage**
    * Gets the MathBrokerRegistryService interface associated with the Connection. If a Connection property (e.g. credentials) is set after the client calls getMathBrokerRegistryService then the newly set Connection property is visible to the MathBrokerRegistryService previously returned by this call.
  - **See Also**
    * `MathbrokerRegistryService`

C.1.3 Interface MathBrokerDeclarativeQueryManager

This interface provides the ability to execute declarative queries (e.g. SQL)

Declaration

public interface MathBrokerDeclarativeQueryManager

C.1.4 Interface MathBrokerFocusedQueryManager

This is the interface exposed by the MathBroker Registry Service.

Declaration

public interface MathBrokerFocusedQueryManager **implements** MathBrokerQueryManager

Methods

- *executeQueryByClassification*
  `public void` **executeQueryByClassification**`( java.lang.String` **concept** `)`
- *executeQueryById*
  `public void` **executeQueryById**`( java.lang.String` **id** `)`
- *executeQueryByName*
  `public void` **executeQueryByName**`( java.lang.String` **name** `)`
- *findMathBrokerObjectByName*
  `public Key` **findMathBrokerObjectByName**`( java.lang.String` **name** `)`
- *findMathBrokerObjectByName*
  `public boolean` **findMathBrokerObjectByName**`( java.lang.String` **name**`,`
  `java.lang.String` **type** `)`

C.1.5 Interface MathBrokerImplementation

DECLARATION

public interface MathBrokerImplementation
implements MathBrokerObject

METHODS

- *setRepositoryItem*
  `public void` **setRepositoryItem**`(` `javax.activation.DataHandler` **repItem** `)`

C.1.6 INTERFACE MathBrokerLifeCycleManager

This is the interface exposed by the Registry Service that implements the life cycle management functionality of the Registry.

DECLARATION

public interface MathBrokerLifeCycleManager

METHODS

- *createMathBrokerAlgorithm*
  `public MathBrokerAlgorithm` **createMathBrokerAlgorithm**`(` `InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem** `)`
- *createMathBrokerAssociations*
  `public Collection` **createMathBrokerAssociations**`(` `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**, `monet.openmath.lang.impl.ImplementationImpl` **element** `)`
- *createMathBrokerImplementation*
  `public MathBrokerImplementation` **createMathBrokerImplementation**`(` `InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem** `)`
- *createMathBrokerMachine*
  `public MathBrokerMachine` **createMathBrokerMachine**`(` `InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem** `)`
- *createMathBrokerProblem*
  `public MathBrokerProblem` **createMathBrokerProblem**`(` `InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem** `)`
- *createMathBrokerService*
  `public MathBrokerService` **createMathBrokerService**`(` `InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem** `)`

- *deleteIncomingAssociations*
public void **deleteIncomingAssociations**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, java.lang.String **uid**)
- *deleteMathBrokerObject*
public void **deleteMathBrokerObject**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerFocusedQueryManagerImpl
fqm, java.lang.String **name**, java.lang.String **type**)
- *deleteMathBrokerObjectAndAssociations*
public void **deleteMathBrokerObjectAndAssociations**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, java.lang.String **uid**)
- *deleteOutgoingAssociations*
public void **deleteOutgoingAssociations**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, java.lang.String **uid**)
- *publishMathBrokerObject*
public void **publishMathBrokerObject**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, javax.activation.DataHandler **repItem**)
- *saveMathBrokerAlgorithm*
public void **saveMathBrokerAlgorithm**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, monet.openmath.lang.impl.AlgorithmImpl **element**, javax.xml.bind.JAXBContext
context, java.lang.String **uri**, java.lang.String **uid**)
- *saveMathBrokerImplementation*
public void **saveMathBrokerImplementation**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, monet.openmath.lang.impl.ImplementationImpl **element**,
javax.xml.bind.JAXBContext **context**, java.lang.String **uri**, java.lang.String **uid**)
- *saveMathBrokerMachine*
public void **saveMathBrokerMachine**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, at.ac.uni_linz.risc.mathbroker.lang.impl.MachineHardwareImpl **element**,
javax.xml.bind.JAXBContext **context**, java.lang.String **uri**, java.lang.String **uid**)
- *saveMathBrokerObject*
public void **saveMathBrokerObject**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, java.lang.Object **element**, javax.xml.bind.JAXBContext **context**,
java.lang.String **uri**, java.lang.String **uid**, java.lang.String **type**)
- *saveMathBrokerProblem*
public void **saveMathBrokerProblem**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, monet.openmath.lang.impl.ProblemImpl **element**, javax.xml.bind.JAXBContext
context, java.lang.String **uri**, java.lang.String **uid**)
- *saveMathBrokerService*
public void **saveMathBrokerService**(
at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager
fqm, monet.openmath.lang.impl.ServiceImpl **element**, javax.xml.bind.JAXBContext
context, java.lang.String **uri**, java.lang.String **uid**)

C.1.7 Interface MathBrokerMachine

The MathBrokerMachine interface is used to represent machine elements in the Mathbroker information model

Declaration

public interface MathBrokerMachine
implements MathBrokerObject

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.1.8 Interface MathBrokerObject

Declaration

public interface MathBrokerObject

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`
  - **Usage**
    * Shows the contents of the repository item using the JAXB generated MSDL API.
  - **See Also**
    * `null`

C.1.9 Interface MathBrokerProblem

Declaration

public interface MathBrokerProblem
implements MathBrokerObject

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.1.10 Interface **MathBrokerQueryManager**

This is the interface exposed by the MathBroker Registry Service.

Declaration

public interface MathBrokerQueryManager

Methods

- *getMathBrokerRegistryService*
  `public MathBrokerRegistryService` **`getMathBrokerRegistryService`**`( )`

C.1.11 Interface **MathBrokerRegistryService**

Declaration

public interface MathBrokerRegistryService

Methods

- *getMathBrokerDeclarativeQueryManager*
  `public MathBrokerDeclarativeQueryManager` **`getMathBrokerDeclarativeQueryManager`**`( )`
  - **Usage**
    * Returns the DeclarativeQueryManager interface implemented by the JAXR provider.
- *getMathBrokerFocusedQueryManager*
  `public MathBrokerFocusedQueryManager` **`getMathBrokerFocusedQueryManager`**`( )`
  - **Usage**
    * Returns the MathBrokerQueryManager interface implemented by the JAXR provider
- *getMathBrokerLifeCycleManager*
  `public MathBrokerLifeCycleManager` **`getMathBrokerLifeCycleManager`**`( )`
  - **Usage**
    * Returns the MathBrokerLifeCycleManager interface implemented by the JAXR provider
  - **See Also**
    * `LifeCycleManager`

C.1.12 Interface **MathBrokerService**

Declaration

public interface MathBrokerService **implements** MathBrokerObject

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2 Classes

C.2.1 Class MathBrokerAlgorithmImpl

Declaration

public class MathBrokerAlgorithmImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerAlgorithm

Constructors

- *MathBrokerAlgorithmImpl*
  `public` **MathBrokerAlgorithmImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm `)`
- *MathBrokerAlgorithmImpl*
  `public` **MathBrokerAlgorithmImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  lcm`, ExtrinsicObjectType` **tEobj** `)`

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`

Methods inherited from class
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl`

(in C.2.9, page 42)

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2.2 Class MathBrokerConnectionFactoryImpl

Class Declaration

Declaration

public class MathBrokerConnectionFactoryImpl
extends ConnectionFactoryImpl

CONSTRUCTORS

- *MathBrokerConnectionFactoryImpl*
  `public` **MathBrokerConnectionFactoryImpl**`( )`

METHODS

- *createConnection*
  `public Connection` **createConnection**`( )`
  - **Usage**
    * Create a named connection. Such a connection can be used to communicate with a JAXR provider.

- *createMathBrokerConnection*
  `public MathBrokerConnection` **createMathBrokerConnection**`( )`
- *newInstance*
  `public static ConnectionFactory` **newInstance**`( )`

C.2.3 CLASS MathBrokerConnectionImpl

MathBrokerConnectionImpl

DECLARATION

public class MathBrokerConnectionImpl
extends ConnectionImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerConnection

CONSTRUCTORS

- *MathBrokerConnectionImpl*
  `public` **MathBrokerConnectionImpl**`( ConnectionFactoryImpl` **factory** `)`

METHODS

- *getMathBrokerRegistryService*
  `public MathBrokerRegistryService` **getMathBrokerRegistryService**`( )`
  - **Usage**
    * Gets the MathBrokerRegistryService interface associated with the Connection.
  - **See Also**
    * `null`

C.2.4 CLASS MathBrokerDeclarativeQueryManagerImpl

Class Declaration

DECLARATION

public class MathBrokerDeclarativeQueryManagerImpl
extends DeclarativeQueryManagerImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerDeclarativeQueryManager

CONSTRUCTORS

- *MathBrokerDeclarativeQueryManagerImpl*
  `public` **MathBrokerDeclarativeQueryManagerImpl**(`RegistryServiceImpl` **regService**, `BusinessLifeCycleManagerImpl` **lcm**)
- *MathBrokerDeclarativeQueryManagerImpl*
  `public` **MathBrokerDeclarativeQueryManagerImpl**(`RegistryServiceImpl` **regService**, `BusinessLifeCycleManagerImpl` **lcm**, `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerRegistryServiceImpl` **mRegService**, `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl` **mlcm**)

C.2.5 CLASS MathBrokerFocusedQueryManagerImpl

Class Declaration

DECLARATION

public class MathBrokerFocusedQueryManagerImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerQueryManagerImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager

CONSTRUCTORS

- *MathBrokerFocusedQueryManagerImpl*
  `public` **MathBrokerFocusedQueryManagerImpl**(`RegistryServiceImpl` **regService**, `BusinessLifeCycleManagerImpl` **lcm**, `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerRegistryServiceImpl` **mRegService**, `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl` **mlcm**)

METHODS

- *executeQueryByClassification*
  `public void` **executeQueryByClassification**(`java.lang.String` **concept**)
  - **Usage**
    * Searches for Mathbroker objects corresponding to the given math classification concept and displays information about them from each coressponding repository item.
  - **Parameters**

    * `concept` - is /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+typename typename can be problem, algorithm, service, implementation, or machine.
  - **Exceptions**
    * `JAXRException.` -
    * `JAXBException.` -
    * `IOException.` -

- *executeQueryById*
  `public void` **executeQueryById(** `java.lang.String` **id** `)`
  - **Usage**
    * Searches for Mathbroker object corresponding to the given id and displays information about it from the coressponding repository item.
  - **Parameters**
    * `id` - = key.getId().
  - **Exceptions**
    * `JAXRException.` -
    * `JAXBException.` -
    * `IOException.` -

- *executeQueryByName*
  `public void` **executeQueryByName(** `java.lang.String` **name** `)`
  - **Usage**
    * Searches for Mathbroker object corresponding to the given name and displays information about it from the coressponding repository item.
  - **Parameters**
    * `name` - is formed from the targetNamespace, "/", and the name of the element to be found.
  - **Exceptions**
    * `JAXRException.` -
    * `JAXBException.` -
    * `IOException.` -

- *findMathBrokerObjectByName*
  `public Key` **findMathBrokerObjectByName(** `java.lang.String` **name** `)`
  - **Usage**
    * Finds the math object that matchs the name specified by the parameter of this call.
  - **Parameters**
    * `name` - is formed from the targetNamespace, "/", and the name of the element to be found.
  - **Returns** - Key of the found object.
  - **Exceptions**
    * `JAXRException.` -

- *findMathBrokerObjectByName*
  `public boolean` **findMathBrokerObjectByName(** `java.lang.String` **name**, `java.lang.String` **type** `)`
  - **Usage**
    * Searches for Mathbroker object corresponding to the given parameters and displays information about it from the coressponding repository item.
  - **Parameters**
    * `name` - is formed from the targetNamespace, "/", and the name of the element to be found.
    * `type` - is problem, algorithm, implementation, service, or machine.
  - **Exceptions**
    * `JAXRException.` -
    * `JAXBException.` -
    * `IOException.` -

METHODS INHERITED FROM CLASS
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerQueryManagerImpl`

(in C.2.11, page 43)

- *getMathBrokerRegistryService*
  `public MathBrokerRegistryService` **getMathBrokerRegistryService**`( )`

C.2.6 CLASS MathBrokerImplementationImpl

DECLARATION

public class MathBrokerImplementationImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerImplementation

CONSTRUCTORS

- *MathBrokerImplementationImpl*
  `public` **MathBrokerImplementationImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm `)`
- *MathBrokerImplementationImpl*
  `public` **MathBrokerImplementationImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm`, ExtrinsicObjectType` **tEobj** `)`

METHODS

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`

METHODS INHERITED FROM CLASS
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl`

(in C.2.9, page 42)

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2.7 CLASS MathBrokerLifeCycleManagerImpl

DECLARATION

public class MathBrokerLifeCycleManagerImpl
extends LifeCycleManagerImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerLifeCycleManager

Methods

- *createMathBrokerAlgorithm*
  `public MathBrokerAlgorithm` **createMathBrokerAlgorithm**`( InternationalString` **name**`, InternationalString` **description**`, javax.xml.registry.infomodel.Classification` **classification**`, javax.activation.DataHandler` **repItem** `)`
  - **Usage**
    * Creates a MathBrokerAlgorithm instance using the specified parameters.
  - **Parameters**
    * `name` - is extracted from the repository item.
    * `description` - is extracted from the repository item.
    * `classification` - is specified by: /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+algorithm
  - **Returns** - the MathBrokerAlgorithm instance created.
  - **Exceptions**
    * `JAXRException` - if the JAXR provider encounters an internal error

- *createMathBrokerAssociations*
  `public Collection` **createMathBrokerAssociations**`( at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, monet.openmath.lang.impl.ImplementationImpl` **element** `)`
  - **Usage**
    * Adds outgoing association(s) to the given Implementation source object based on the hrefs included in the element and using the given parameters.
  - **Exceptions**
    * `JAXRException` -

- *createMathBrokerImplementation*
  `public MathBrokerImplementation` **createMathBrokerImplementation**`( InternationalString` **name**`, InternationalString` **description**`, javax.xml.registry.infomodel.Classification` **classification**`, javax.activation.DataHandler` **repItem** `)`
  - **Usage**
    * Creates a MathBrokerImplementation instance using the specified parameters.
  - **Parameters**
    * `name` - is extracted from the repository item.
    * `description` - is extracted from the repository item.
    * `classification` - is specified by: /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+implementation
    * `repositoryItem` - the DataHandler for the repository item. Must not be null.
  - **Returns** - the MathBrokerImplementation instance created.
  - **Exceptions**
    * `JAXRException.` -

- *createMathBrokerMachine*
  `public MathBrokerMachine` **createMathBrokerMachine**`( InternationalString` **name**`, InternationalString` **description**`, javax.xml.registry.infomodel.Classification` **classification**`, javax.activation.DataHandler` **repItem** `)`
  - **Usage**
    * Creates a MathBrokerMachine instance using the specified parameters.
  - **Parameters**

    * `name` - is extracted from the repository item.
    * `description` - is extracted from the repository item.
    * `classification` - is specified by by:
      /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+machine
    * `repositoryItem` - the DataHandler for the repository item. Must not be null.
  - **Returns** - the MathBrokerMachine instance created
  - **Exceptions**
    * `JAXRException` - if the JAXR provider encounters an internal error

---

- *createMathBrokerProblem*
  `public MathBrokerProblem` **createMathBrokerProblem**`( InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem**)
  - **Usage**
    * Creates a MathBrokerProblem instance using the specified parameters.
  - **Parameters**
    * `name` - is extracted from the repository item.
    * `description` - is extracted from the repository item.
    * `classification` - is specified by:
      /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+problem
    * `repositoryItem` - the DataHandler for the repository item. Must not be null.
  - **Returns** - the MathBrokerProblem instance created
  - **Exceptions**
    * `JAXRException` - if the JAXR provider encounters an internal error

---

- *createMathBrokerService*
  `public MathBrokerService` **createMathBrokerService**`( InternationalString` **name**, `InternationalString` **description**, `javax.xml.registry.infomodel.Classification` **classification**, `javax.activation.DataHandler` **repItem**)
  - **Usage**
    * Creates a MathBrokerService instance using the specified parameters.
  - **Parameters**
    * `name` - is extracted from the repository item.
    * `description` - is extracted from the repository item.
    * `classification` - is specified by:
      /urn:uuid:f2552642-97a3-4e40-95da-b3ef92ab34ab/Entities/"+algorithm
    * `repositoryItem` - the DataHandler for the repository item. Must not be null.
  - **Returns** - the MathBrokerService instance created
  - **Exceptions**
    * `JAXRException.` -

- *deleteIncomingAssociations*
  `public void` **deleteIncomingAssociations**(`at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**, `java.lang.String` **uid**)
  - **Usage**
    * Delete all associations where the element is a target object using the given parameters.
  - **Parameters**
    * `uid` - is formed from the target name space + "/" + the name of the element.
  - **Exceptions**
    * `JAXRException` -

- *deleteMathBrokerObject*
`public void` **deleteMathBrokerObject**`(`
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerFocusedQueryManagerImpl`
fqm`, java.lang.String` **name**`, java.lang.String` **type** `)`
  - **Usage**
    * Delets a MathBrokerObject using the specified parameters.
  - **Exceptions**
    * `JAXRException.` -

- *deleteMathBrokerObjectAndAssociations*
`public void` **deleteMathBrokerObjectAndAssociations**`(`
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager`
fqm`, java.lang.String` **uid** `)`
  - **Usage**
    * Delete mathbroker element and its incoming/outgoing associations based on the given parameters.
  - **Parameters**
    * `uid` - is formed from the target name space + the name of the element.
  - **Exceptions**
    * `JAXRException` -

- *deleteOutgoingAssociations*
`public void` **deleteOutgoingAssociations**`(`
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager`
fqm`, java.lang.String` **uid** `)`
  - **Usage**
    * Delete all associations where the element is a source objec using the given parameters.
  - **Parameters**
    * `uid` - is formed from the target name space + "/" + the name of the element.
  - **Exceptions**
    * `JAXRException` -

- *publishMathBrokerObject*
`public void` **publishMathBrokerObject**`(`
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager`
fqm`, javax.activation.DataHandler` **repItem** `)`
  - **Usage**
    * Publishes a MathBroker object instance using the specified parameters. It takes an xml repository item which should conform to mathbroker schema definition, unmarshals it, processes and determines each mathbroker element and forms a unique name for each mathbroker element/entity. The unique name/identifier (uid) is formed by concatenating the element name to the target name space. It creates individaul xml repository items for each element and calls the saveMathbrokerObject method to save the newly created element and repository item in the registry.
  - **Parameters**
    * `repositoryItem` - the DataHandler for the repository item. Must not be null.
  - **Returns** - the MathBrokerService instance created.
  - **Exceptions**
    * `JAXRException` - if the JAXR provider encounters an internal error.
    * `JABException.` -

* IOException. -

- *saveMathBrokerAlgorithm*
  `public void` **saveMathBrokerAlgorithm**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, monet.openmath.lang.impl.AlgorithmImpl` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid** `)`
  - **Usage**
    * Saves a MathBrokerAlgorithm instance using the specified parameters. if the object already exits in the registry it updates that object.
  - **Exceptions**
    * JABException. -
    * JAXRException. -
    * FileNotFoundException. -
    * javax.xml.bind.PropertyException -

- *saveMathBrokerImplementation*
  `public void` **saveMathBrokerImplementation**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, monet.openmath.lang.impl.ImplementationImpl` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid** `)`
  - **Usage**
    * Saves a MathBrokerImplementation instance using the specified parameters. if the object already exits in the registry it updates that object.
  - **Exceptions**
    * JABException. -
    * JAXRException. -
    * FileNotFoundException. -
    * javax.xml.bind.PropertyException -

- *saveMathBrokerMachine*
  `public void` **saveMathBrokerMachine**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, at.ac.uni_linz.risc.mathbroker.lang.impl.MachineHardwareImpl` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid** `)`
  - **Usage**
    * Saves a MathBrokerMachine instance using the specified parameters. if the object already exits in the registry it updates that object.
  - **Exceptions**
    * JABException. -
    * JAXRException. -
    * FileNotFoundException. -
    * javax.xml.bind.PropertyException -

- *saveMathBrokerObject*
  `public void` **saveMathBrokerObject**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, java.lang.Object` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid**`, java.lang.String` **type** `)`
  - **Usage**
    * Saves a MathBroker object instance using the specified parameters. if the object already exits in the registry it updates that object.

- **Exceptions**
  - `JABException.` -
  - `JAXRException.` -

- *saveMathBrokerProblem*
  `public void` **saveMathBrokerProblem**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, monet.openmath.lang.impl.ProblemImpl` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid** `)`
  - **Usage**
    - Saves a MathBrokerProblem instance using the specified parameters. if the object already exits in the registry it updates that object.
  - **Exceptions**
    - `JABException.` -
    - `JAXRException.` -
    - `FileNotFoundException.` -
    - `javax.xml.bind.PropertyException` -

- *saveMathBrokerService*
  `public void` **saveMathBrokerService**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerFocusedQueryManager` **fqm**`, monet.openmath.lang.impl.ServiceImpl` **element**`, javax.xml.bind.JAXBContext` **context**`, java.lang.String` **uri**`, java.lang.String` **uid** `)`
  - **Usage**
    - Saves a MathBrokerService instance using the specified parameters. if the object already exits in the registry it updates that object.
  - **Exceptions**
    - `JABException.` -
    - `JAXRException.` -
    - `FileNotFoundException.` -
    - `javax.xml.bind.PropertyException` -

- *saveObject*
  `public void` **saveObject**`( java.util.Collection` **mathbrokerObjects** `)`
  - **Usage**
    - Does the actual saving when called by the saveMathBroker element methodes using the specified parameters.
  - **Exceptions**
    - `JAXRException.` -

C.2.8 Class MathBrokerMachineImpl

Declaration

---

public class MathBrokerMachineImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerMachine

Constructors

- *MathBrokerMachineImpl*
  `public` **MathBrokerMachineImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm `)`
- *MathBrokerMachineImpl*
  `public` **MathBrokerMachineImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm`, ExtrinsicObjectType` **tEobj** `)`

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`

Methods inherited from class
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl`

(in C.2.9, page 42)
- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2.9 Class MathBrokerObjectImpl

Declaration

public abstract class MathBrokerObjectImpl
extends ExtrinsicObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerObject

Constructors

- *MathBrokerObjectImpl*
  `public` **MathBrokerObjectImpl**`( LifeCycleManagerImpl` **lcm** `)`
- *MathBrokerObjectImpl*
  `public` **MathBrokerObjectImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm`, ExtrinsicObjectType` **tEobj** `)`

Methods

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2.10 Class MathBrokerProblemImpl

DECLARATION

public class MathBrokerProblemImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerProblem

CONSTRUCTORS

- *MathBrokerProblemImpl*
  `public` **MathBrokerProblemImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm `)`
- *MathBrokerProblemImpl*
  `public` **MathBrokerProblemImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm`, ExtrinsicObjectType` **tEobj** `)`

METHODS

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`

METHODS INHERITED FROM CLASS
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl`

(in C.2.9, page 42)

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

C.2.11 CLASS MathBrokerQueryManagerImpl

DECLARATION

public abstract class MathBrokerQueryManagerImpl
extends BusinessQueryManagerImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerQueryManager

CONSTRUCTORS

- *MathBrokerQueryManagerImpl*
  `public` **MathBrokerQueryManagerImpl**`( RegistryServiceImpl` **regService**`,`
  `BusinessLifeCycleManagerImpl` **lcm**`,`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerRegistryServiceImpl`
  mRegService`,`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  mlcm `)`

Methods

- *getMathBrokerRegistryService*
  `public MathBrokerRegistryService` **getMathBrokerRegistryService**`( )`

C.2.12 Class MathBrokerRegistryServiceImpl

Class Declaration

Declaration

public class MathBrokerRegistryServiceImpl
extends RegistryServiceImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerRegistryService

Constructors

- *MathBrokerRegistryServiceImpl*
  `public` **MathBrokerRegistryServiceImpl**`( ConnectionImpl` **c** `)`

Methods

- *getMathBrokerConnection*
  `public MathBrokerConnectionImpl` **getMathBrokerConnection**`( )`
  - **Usage**
    * Returns the MathBroker connection.

- *getMathBrokerDeclarativeQueryManager*
  `public MathBrokerDeclarativeQueryManager` **getMathBrokerDeclarativeQueryManager**`( )`
  - **Usage**
    * Returns the MathBrokerDeclarativeQueryManager interface implementation.

- *getMathBrokerFocusedQueryManager*
  `public MathBrokerFocusedQueryManager` **getMathBrokerFocusedQueryManager**`( )`
  - **Usage**
    * Returns the MathBrokerFocusedQueryManager interface implementation.

- *getMathBrokerLifeCycleManager*
  `public MathBrokerLifeCycleManager` **getMathBrokerLifeCycleManager**`( )`
  - **Usage**
    * Returns the MathBrokerLifeCycleManager interface implementation.

C.2.13 Class MathBrokerServiceImpl

DECLARATION

public class MathBrokerServiceImpl
extends at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl
implements at.ac.uni_linz.risc.mathbroker.registry.infomodel.MathBrokerService

CONSTRUCTORS

- *MathBrokerServiceImpl*
  `public` **MathBrokerServiceImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  lcm `)`
- *MathBrokerServiceImpl*
  `public` **MathBrokerServiceImpl**`(`
  `at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerLifeCycleManagerImpl`
  lcm`, ExtrinsicObjectType` **tEobj** `)`

METHODS

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`
- *showContent*
  `public void` **showContent**`( )`

METHODS INHERITED FROM CLASS
`at.ac.uni_linz.risc.mathbroker.registry.infomodel.impl.MathBrokerObjectImpl`

(in C.2.9, page 42)

- *setRepositoryItem*
  `public void` **setRepositoryItem**`( javax.activation.DataHandler` **repItem** `)`

References

[1] Olga Caprotti and Wolfgang Schreiner. Towards a Mathematical Service Description Language. In *International Congress of Mathematical Software ICMS 2002*, Bejing, China, August 17–19, 2002. World Scientific Publishing, Singapore.

[2] Mike Dewar, David Carlisle, and Olga Caprotti. Description Schemes for Mathematical Web Services. In *EuroWeb 2002: The Web and the Grid: From e-Science to e-Business*, Oxford, UK, December 2002. Britisch Computer Society Electronic Workshops in Computing.

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

[4] ebXML Registry Services Specification v2.0. OASIS, April 2002. http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebRS.pdf.

[5] Java Architecture for XML Binding (JAXB). Sun microsystems, April 2004. http://java.sun.com/xml/jaxb/.

[6] Java API for XML Registries (JAXR) v0.9. Sun microsystems, April 2002. http://java.sun.com/xml/jaxr/.

[7] Java Web Services Developer Pack (JWSDP). Sun microsystems, April 2004. http://java.sun.com/webservices/webservicespack.html.

[8] MathBroker — A Framework for Brokering Distributed Mathematical Services. Reseach Institute for Symbolic Computation (RISC), April 2004. http://www.risc.uni-linz.ac.at/projects/basic/mathbroker.

[9] MONET — Mathematics on the Web. The MONET Consortium, April 2004. http://monet.nag.co.uk.

[10] Mathematical Services Description Language (MSDL). Reseach Institute for Symbolic Computation (RISC), April 2004. http://poseidon.risc.uni-linz.ac.at:8080/mathbroker/results/xsd.html.

[11] Mathematical Services Description Language Library API. Reseach Institute for Symbolic Computation (RISC), April 2004. http://poseidon.risc.uni-linz.ac.at:8080/results/xsd/monet-based/api/index.html.

[12] Mathematical Services Description Language (MSDL) Schema. Reseach Institute for Symbolic Computation (RISC), April 2004. http://poseidon.risc.uni-linz.ac.at:8080/results/xsd/monet-based/mathbroker.xsd.

[13] MathBroker Registry API. Reseach Institute for Symbolic Computation (RISC), April 2004. http://poseidon.risc.uni-linz.ac.at:8080/results/registry/MBregistryAPI/.

[14] MathBroker Samples. Reseach Institute for Symbolic Computation (RISC), April 2004. http://perseus.risc.uni-linz.ac.at:8080/openmath/.

[15] Semantic Web. World Wide Web Consortium, March 2004. http://www.w3.org/2001/sw.

[16] Apache Tomcat. The Apache Jakarta project, April 2004. http://jakarta.apache.org/tomcat/.

[17] UDDI Version 2.04 API Specification. OASIS, July 2002. http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf.