A Framework for Publishing and Discovering Mathematical Web Services

Thesis - January 2006

1 author:

Rebhi Baraka
Islamic University of Gaza

Some of the authors of this publication are also working on these related projects:

- ontology engineering, big data analysis

All content following this page was uploaded by Rebhi Baraka on 03 June 2014.

The user has requested enhancement of the downloaded file.
A Framework for Publishing and Discovering Mathematical Web Services

Rebhi Baraka
rbaraka@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at

August 11, 2006
Outline

1 Foundation
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2 The Framework for Service Publishing and Discovering
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQL
   - The Overall Framework

3 Performance Analysis

4 Conclusion
The foundation of this work is an infrastructure for service development and description consisting of:

- A mathematical Web services architecture.
- A language for describing mathematical services: the Mathematical Services Description Language (MSDL).
Based on standards such as XML, SOAP, and WSDL and mathematical standards and software such as OpenMath, computer algebra and automated reasoning systems.
Service Description: The Information Model

- Algorithm
- Problem
- Implementation
- Realization
- Service (WSDL)
- Machine Type

Relationships:
- Algorithm solves Problem
- Implementation implements Algorithm
- Realization is based on Algorithm
- Service (WSDL) is bound to service port
- Service (WSDL) is located on Machine Type
- Problem is special version of Problem
MSDL is an implementation of the information model.

It is an XML schema-based language.

Mathematical content in MSDL is represented in a semantics preserving way using the OpenMath standard.
A Skeleton of a Service Description in MSDL

```xml
<monet:definitions>
  <monet:problem name=‘‘indefinite-integration’’>
    ...
  </monet:problem>
  <monet:algorithm name=‘‘RischAlg’’>
    ...
  </monet:algorithm>
  <monet:implementation name=‘‘RImpl’’>
    ...
  </monet:implementation>
  <monet:service name=‘‘RRISC’’>
    ...
  </monet:service>
  <mathb:machine_hardware name=‘‘perseus.risc.uni-linz.ac.at’’>
    ...
  </mathb:machine_hardware>
</monet:definitions>
```
1. **Foundation**
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2. **The Framework for Service Publishing and Discovering**
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQL
   - The Overall Framework

3. **Performance Analysis**

4. **Conclusion**
A registry is a Web-based shared resource for publishing and discovering Web services.

The mathematical registry provides a set of functionalities such that MSDL descriptions can be submitted, updated, associated, classified, and discovered.

Based on OASIS ebXML registry standard. Extends its information model and functionality.

Implements the extension on top of the ebXML reference implementation.
Extending ebXML Information Model

Inheritance:  
Association:  

RegistryObject
ExtrinsicObject
MathbrokerObject

Machine  Realization  Implementation  Algorithm  Problem
RunsOn  IsBasedOn  Implements  Solves  IsSpecVersOf

ebXML objects
MathBroker objects

Rebhi Baraka  Publishing and Discovering Mathematical Web Services
Implemented as the Mathematical Registry API.

Math classification scheme (GAMS) is submitted.

Client applications for publishing and discovering.
Publishing and Discovering Service Descriptions

A Registry for Publishing Service Descriptions
The Mathematical Services Query Language (MSQL)
Semantic Extension to MSQL
The Overall Framework

Object Metadata (stored in the registry database)

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>repItem</th>
<th>association(s)</th>
<th>classification(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RischAlg</td>
<td>Algorithm</td>
<td>URI</td>
<td>Solves</td>
<td>Symbolic comp.</td>
</tr>
</tbody>
</table>

Classification Schemes
- MathBroker
  - Realization
  - Implement
  - Algorithm
  - Problem
- GAMS
  - Special functions
  - Symbolic computation
- Integrals of elementary fnc
- CPU
  - Intel
  - Celeron

Extrinsic Objects
- Mathematical Objects
  - RischAlg
  - Solves
  - Indefinite integration
  - Implements
  - Rlmpl
  - IsBasedOn
  - RRISC
  - RunsOn
  - perseus

Repository Item
```xml
<problem name="indefinite-integration">
  <header/>
  <body>
    <input name="F" />
    <OMOBJ>
      <OMS cd="sts" name="mapsto"/>
    </OMOBJ>
    <OMS cd="setname1" name="R"/>
    <OMS cd="setname1" name="R"/>
  </body>
</problem>
```

Repository
- RischAlg description
- Indefinite-integration description
- Rlmpl description
- RRISC description
- perseus description
- URL:
Querying Service Descriptions

- **Metadata-based querying**: such as querying registry objects by name, or by classification. This is already supported by the registry.
  
  **Example**: Find a problem with name \texttt{lcm} and is classified under the classification concept “GAMS/Symbolic Computation”.

- **Content-based querying**: Involves documents containing MSDL descriptions of published objects.
  
  **Example**: Find a problem classified under the classification concept “GAMS/Symbolic Computation” whose first input argument has type integer.

The Mathematical Services Query Language (MSQL) is developed to facilitate the second type of querying on top of the first one.
1 Foundation
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2 The Framework for Service Publishing and Discovering
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQL
   - The Overall Framework

3 Performance Analysis

4 Conclusion
MSQL is a light-weight query language for querying the contents of MSDL documents published in the registry.

Characteristics of MSQL:
- Functional.
- Precise Semantics.
- Registry Interfacing.
- Result Ordering.
The general structure of a query in MSQL is:

```
SELECT EVERY|SOME <entity>
FROM <classificationConcept>
WHERE <expression>
ORDERBY <expression> ASCENDING|DESCENDING
```
MSQL Expressions

MSQL uses a minimal set of expressions that are necessary to address the contents of MSDL documents.

- Path expressions.
- Arithmetic, logical, and comparison expressions.
- Function expressions.
- Variable binding expressions.
- Conditional expressions.
- Quantified expressions.
- Semantic expressions.
Example 1: Path Expressions

Find all problems under the classification concept "/GAMS/Symbolic Computation" with first input having type integer and order them according to their names in descending order.

```
SELECT EVERY problem
FROM /GAMS/Symbolic Computation
WHERE //body/input[1]/signature/OMOBJ/
    OMS[1][(( @name = 'Z' )
    and ( @cd = 'setname1' ))]
ORDERBY /problem/@name DESCENDING
```
Example 2: Conditional Expressions

Find every service in “/GAMS/Linear Algebra” such that, if it has an implementation, it runs on a machine called “perseus”, otherwise its interface is on the said machine.

```
SELECT EVERY service  
FROM /GAMS/Linear Algebra  
WHERE  
  if not (/service[empty(/implementation)])  
  then let $d := doc(/implementation/@href) in  
      $d/hardware[contains(@name, ‘perseus’)]  
  else //service-interface-description[  
      contains(@href, ‘perseus’)]
```
MSQL Engine Architecture

- MSQL Engine constitutes the functionality of MSQL.
- Its components correspond to the MSQL `select-from-where-orderby` clauses.
- Implemented as the MSQL API.
A Prototype Implementation

- Based on the denotational semantics of MSQL.
- Denotational semantics clarifies the intended meaning of the language constructs and provides a reference for implementation.
- For example, the semantic function
  \[ E[V] d = lookup(d, [V]) \]
  is implemented by the Java method

```java
static private Value evaluateVarExpr(ChildAST expr, Declaration declaration) throws MsqlException {
    VarExpr varExpr = (VarExpr)expr;
    return declaration.lookup(varExpr);
}
```
1. Foundation
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2. The Framework for Service Publishing and Discovering
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQML
   - The Overall Framework

3. Performance Analysis

4. Conclusion
Semantic Extension to MSQIL

- MSDL descriptions contain semantic information represented in OpenMath.
- Semantic-based Queries are formed based on such semantics.
- Semantic expressions are expressed in predicate logic.
- Defining new constructs to express predicate logic formulae as OpenMath objects.
- Adding a semantic evaluator to the MSQL engine to process these formulae with the help of an automated reasoner.
Mathematical content in MSDL is represented in a semantics preserving way using the OpenMath standard.

The following pieces of semantic information for a problem specification are expressed in OpenMath (next slide):

- **Input**: $f : \mathbb{R} \rightarrow \mathbb{R}$
- **Output**: $i : \mathbb{R} \rightarrow \mathbb{R}$
- **Post-condition**: $i = \text{indfint}(f)$
<problem name="‘indefinite-integration’">
<header></header>
<body>
  <input name="‘f’">
  <signature>
    <OMOBJ>
      <OMA>
        <OMS cd="‘sts’" name="‘mapsto’"></OMS>
        <OMS cd="‘setname1’" name="‘R’"></OMS>
        <OMS cd="‘setname1’" name="‘R’"></OMS>
      </OMA>
    </OMOBJ>
  </signature>
</input>
  <output name="‘i’">
    ...
  </output>
</body>
A Problem Description (cont.)

```xml
<output>
<post-condition>
<OMOBJ>
  <OMA>
    <OMS cd='relation1' name='eq'></OMS>
    <OMV name='i'></OMV>
  </OMA>
  <OMA>
    <OMS cd='calculus1' name='indefint'></OMS>
    <OMV name='f'></OMV>
  </OMA>
</OMOBJ>
</post-condition>
</body>
```

Rebhi Baraka
Publishing and Discovering Mathematical Web Services
Example 3: Semantic Expressions

- **Problem Specification:**
  - Input: \( a : \mathbb{R} \rightarrow \mathbb{R} \)
  - Output: \( b : \mathbb{R} \rightarrow \mathbb{R} \)
  - Post-condition: \( \text{diff}(b) = a \)

- **Knowledge:** \( \text{diff}(\text{indefint}(a)) = a \)

- **Find some service with problem \( p \) such that**
  \[
  \text{type}[\text{input}_p] = \mathbb{R} \rightarrow \mathbb{R} \land \\
  \text{type}[\text{output}_p] = \mathbb{R} \rightarrow \mathbb{R} \land \\
  \forall a \in \mathbb{R} \rightarrow \mathbb{R}, \ b \in \mathbb{R} \rightarrow \mathbb{R} \ (\text{post}_p[a, b] \Rightarrow \text{diff}(b) = a)
  \]
SELECT SOME service
FROM /GAMS/Symbolic computation
WHERE let $p:= \text{doc}(/\text{problem}@\text{href})$ in
  let $a:= p/input/@name,$
  $b:= p/output/@name,$
  $ta:= p/input/signature/OMOBJ,$
  $tb:= p/output/signature/OMOBJ,$
  $post:= p/post-condition/OMOBJ$ in
  (typematch($ta, \text{oma}(\text{oms:sts:mapsto}($
                oms:setname1:R, oms:setname1:R))))$ and
  (typematch($tb, \text{oma}(\text{oms:sts:mapsto}($
                oms:setname1:R, oms:setname1:R))))$ and
  (satisfy(ombind(\text{oms:quant1:forall}$
                [omvar:$a@(\text{oms:sts:type}, \text{\$ta)},$
                 omvar:$b@(\text{oms:sts:type}, \text{\$tb})]$
                \text{oma}(\text{oms:logic1:implies, \$post, \text{oma}(\text{oms:relation1:eq,}$
                    \text{oma}(\text{oms:calculus1:diff, omv:}}\text{\$b), omv:}\text{\$a))))))
MSQL Engine Extended Architecture.
1 Foundation
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2 The Framework for Service Publishing and Discovering
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQL
   - The Overall Framework

3 Performance Analysis

4 Conclusion
The Overall Framework.
1. **Foundation**
   - Service Architecture
   - Service Description: The Information Model
   - The Mathematical Services Description Language (MSDL)

2. **The Framework for Service Publishing and Discovering**
   - A Registry for Publishing Service Descriptions
   - The Mathematical Services Query Language (MSQL)
   - Semantic Extension to MSQL
   - The Overall Framework

3. **Performance Analysis**

4. **Conclusion**
The aim is to measure the relative increments/decrements in time when certain factors (e.g., framework configuration, repository size, and query type) change.

Measurements:
- Publishing time
- Querying time
  - Total time: time starting from parsing the query to getting the result.
  - Retrieval time: time taken to retrieve candidate documents.
  - Evaluation time: time to evaluate the WHERE expression

\[ T_{evaluation} = T_{total} - T_{retrieval} \]
Performance Analysis (Retrieval and Evaluation time for some queries)
Based on the obtained performance results the following can be concluded:

- The *publishing time* is much longer than the *querying time*.
- The *retrieval time* is an order of magnitude larger than the *evaluation time*.
- The *retrieval time* is proportional to the repository size.
- The *evaluation time* is affected by the type of the query (the WHERE expression).
1 Foundation
   • Service Architecture
   • Service Description: The Information Model
   • The Mathematical Services Description Language (MSDL)

2 The Framework for Service Publishing and Discovering
   • A Registry for Publishing Service Descriptions
   • The Mathematical Services Query Language (MSQL)
   • Semantic Extension to MSQL
   • The Overall Framework

3 Performance Analysis

4 Conclusion
Conclusion

The main contributions of this dissertation are:

- A registry for publishing and discovering mathematical Web services.
- A content-based query language for discovering registry-published services.
- An extension to the query language supporting semantic-based discovery of registry-published services.

Recommended extensions include:

- A software agent that receives user computational requests, discovers suitable services (or service compositions) using the framework, invokes these services to perform the request and returns the answer to the user.
- A user-friendly tool for forming queries on target MSDL descriptions.
These contributions have been documented and published through:

- Three RISC technical reports,
- Two software packages (the Mathematical Registry API and the MSQL API), and
- Three papers at the following refereed international conferences:
  - The IEEE 20th International Conference on Advanced Information Networking and Applications (AINA 2006).

Thank You.