

Version: 1.0
Date: March 12th 2003

Mathematical Service Description Language: Final Version

Stephen Buswell¹, Olga Caprotti² and Mike Dewar³

¹Stilo Ltd ²Technical University of Eindhoven ³NAG Ltd

Deliverable D14 (Public)

Contents

1	Introduction	2
2	Abstract model for Mathematical Service Description	2
2.1	The Service Description Itself	2
2.2	The Problem Description	3
2.3	Taxonomies	4
2.4	Directives	4
2.5	Algorithms	4
2.6	Actions	4
2.7	Service Binding Description	5
2.8	Broker Interface	5
A	A Sample Problem Description	5
B	Sample MONET Taxonomy	8

1 Introduction

This document describes the language to be used to define mathematical services in the MONET project. For an overview of the interactions between the different components in the MONET architecture and definitions of terms used in this document, the reader is referred to the separate document “MONET Architecture Overview” (deliverable D04).

There are a number of components to MSDL: the service description itself which is registered with a broker, abstract descriptions of problems which provide a terminology for referring to different parts of a problem, vocabularies for referring to algorithms etc. In practice we anticipate that a broker would be dealing with:

- a collection of descriptions of particular services;
- a problem description library, containing abstract problem descriptions;
- one or more *taxonomies* for categorising mathematical problems;
- ...

The concrete implementation of this language is as a set of schema documents, available on request.

2 Abstract model for Mathematical Service Description

2.1 The Service Description Itself

A Mathematical Service Description consists of the following components:

Classification: a specification of what the service does in terms of

- references to a problem description library;
- references to taxonomies;
- the semantics it supports (a list of OpenMath CDs, or fixed semantics of MathML, Maple etc.);
- a detailed semantic description in a suitable framework such as RDF;
- the *directives* it supports (*find*, *prove*, etc.).

All of these are defined as optional, however a completely empty classification would not be very interesting. It is possible to have multiple instances of any component.

Implementation Details: information about the specific service.

- A reference to a description of the algorithm(s) used.
- Software details.
- Hardware details.
- Algorithmic properties (such as accuracy, resource usage etc.)
- Actions needed to solve a problem (*initialise*, *exec*, *stop*, ...)

All of these are optional, and multiple instances of any component can occur. An empty implementation part is quite reasonable.

Service Interface Description: details of the interfaces exposed by the service. This would typically be a WSDL document, but could also be IDL or some other mechanism.

Service Binding Description: a mapping from abstract problem components and actions to elements of the service interface, allowing calls to the service to be constructed.

Broker Interface: This is the API exposed to the broker by the service, to allow it to generate service instances, perform housekeeping actions etc. Typically it would consist of a service URI and an interface description.

Service Metadata: This is information about the service expressed in any generic (i.e. not mathematics-specific) formalism. For example author information might be encoded using the Dublin Core and access policies could be encoded using the Grid Economic Service Architecture (GESA).

In addition the service description includes a name for the service.

2.2 The Problem Description

A problem description consists of two parts: a header and a body. The header contains metadata such as bibliographical references and pointers to the equivalent problem in various taxonomies. It may also contain references to other problems which it is a generalisation or specialisation of.

The body has four parts:

Inputs: the name and type of each of the initial objects in the problem description.

Outputs: the name and type of each component of the problem solution.

Pre-conditions: a collection of mathematical statements about the inputs.

Post-conditions: a collection of mathematical statements about the inputs and outputs.

Not all of these have to be given. In practice it may be useful simply to provide the inputs and outputs to create a common vocabulary to refer to objects in the problem.

For example, minimisation of a multivariate function over the real numbers could be described as:

Input:

1. $F : \mathbb{R}^n \rightarrow \mathbb{R}$

Output:

1. $x \in \mathbb{R}^n$
2. $f \in \mathbb{R}$

Post-conditions:

1. $F(x) = f$
2. $\nexists y \in \mathbb{R} \mid F(y) < f$

Note that in this case there are no pre-conditions.

2.3 Taxonomies

The easiest way to describe the task which a service performs is by referring to a node in a published taxonomy. MONET will develop such a taxonomy based on the Guide to Available Mathematical Software (GAMS) from NIST. We have chosen GAMS because it describes existing software packages and is quite mature however we recognise that in some cases it is not detailed enough (for example there are no sub-categories under *symbolic computation*) and in some of the area of numerical analysis the descriptions are far too detailed. Our rule of thumb will be that the descriptions should be based on information which is explicit in the query submitted by the user, rather than based on information which must be extracted via computation, to allow a broker to identify the closest match between a user's problem and a service.

The GAMS taxonomy is arranged as a tree (although there are a few cross references) so that one may view the children of a node as sub-classes (or specialisations) of that node. So for example one may say that solving one-dimensional integrals over a finite interval is a sub-class of quadrature which is a sub-class of integration.

MSDL does not provide a representation for taxonomies as there are plenty of existing mechanisms available. A good candidate is RDFS. An example of a piece of GAMS described in RDFS is given in Appendix B.

2.4 Directives

These are verbs used to describe the type of action the client would like the service to perform. Examples include:

Solve: Given a set of inputs and their values, some outputs and pre- and post-conditions, return values for the outputs which satisfy the post-conditions.

Decide: Given a statement return true, false or don't know.

Prove: Given a statement return a proof that it is true or false.

...

For the MONET project we will produce an OpenMath Content Dictionary containing the definitions of the directives which we will use.

2.5 Algorithms

An algorithm description must have a name and a reference to the problem (as specified in the MPDL) which it is designed to solve. It may also contain bibliographic and complexity information.

For the MONET project we will produce an OpenMath Content Dictionary containing symbols for describing algorithmic complexity.

2.6 Actions

Sometimes a service will require several steps to perform a computation and these should be made known to the user. For example an initialisation call followed by a number of compute

steps and finally a termination call to release resources being held. MSDL handles this in two stages, in the implementation section a sequence of *actions* are defined in terms of their rôles. These are then mapped into concrete operations in the service binding section.

2.7 Service Binding Description

This has two sections. The first maps abstract actions to operations for each problem type handled by the service. The second describes how to construct/deconstruct the messages understood by the service from the elements of the problem description.

For example suppose that the Service Interface Description is a WSDL document, and the classification of the service includes a reference to an entry in the Mathematical Problem Library. Every WSDL message is named, and consists of a sequence of named parts. Then every element of the problem description will be related to a named part of a particular message.

2.8 Broker Interface

This is the interface exposed to the broker by the service. This can be used for almost anything — the extent to which a given broker manages a given service is not proscribed by the MONET architecture. Some examples of operations which might be offered include:

Status-report

Create service instance if the service uses a factory model.

Terminate service instance in the case where the broker manages the service life-cycle.

Resource Usage Report in the case where the broker charges the client for resources used.

...

A A Sample Problem Description

In 2.2 we gave an example of a minimisation problem. The corresponding XML is given below, using a number of namespaces as follows:

```
msdl  http://monet.nag.co.uk/monet/msdl
om    http://www.openmath.org/OpenMath
xsi   http://www.w3.org/2001/XMLSchema-instance
```

```

<msdl:problem name='unconstrained_multivariate_optimisation'>
  <msdl:header>
    <msdl:classification taxonomy="http:monet.nag.co.uk/gams.rdf code="G1b"/>
  </msdl:header>
  <msdl:body>
    <msdl:inputs>
      <msdl:input name='msdl_umo_F'>
        <msdl:signature>
          <om:MOBJ>
            <om:OMA>
              <om:OMS cd="sts" name="mapsto"/>
            <om:OMA>
              <om:OMS cd="arith1" name="power"/>
              <om:OMS cd="setname1" name="R"/>
            </om:OMA>
            <om:OMS cd="setname1" name="R"/>
          </om:OMA>
        </om:MOBJ>
      </msdl:signature>
    </msdl:input>
  </msdl:inputs>
  <msdl:outputs>
    <msdl:output name='msdl_umo_x'>
      <msdl:signature>
        <om:MOBJ>
          <om:OMA>
            <om:OMS cd="arith1" name="power"/>
            <om:OMS cd="setname1" name="R"/>
          </om:OMA>
        </om:MOBJ>
      </msdl:signature>
    </msdl:output>
    <msdl:output name='msdl_umo_f'>
      <msdl:signature>
        <om:MOBJ>
          <om:OMS cd="setname1" name="R"/>
        </om:MOBJ>
      </msdl:signature>
    </msdl:output>
  </msdl:outputs>
  <msdl:pre-conditions/>
  <msdl:post-conditions>
    <msdl:post-condition>
      <om:MOBJ>
        <om:OMA>
          <om:OMS cd="relation1" name="eq"/>
        <om:OMA>
          <om:OMV name='msdl_umo_F' />
          <om:OMV name='msdl_umo_x' />
        </om:OMA>
      </om:MOBJ>
    </msdl:post-condition>
  </msdl:post-conditions>
</msdl:body>
</msdl:problem>

```

```

        </om:OMA>
        <om:OMV name='msdl_umo_f' />
    </om:OMA>
</om:OMOBJ>
</msdl:post-condition>
<msdl:post-condition>
    <om:OMOBJ>
        <om:OMA>
            <om:OMS cd="logic1" name="not" />
            <om:OMBIND>
                <om:OMS cd="quant1" name="exists" />
                <om:OMBVAR>
                    <om:OMV name="y" />
                </om:OMBVAR>
            </om:OMBIND>
            <om:OMA>
                <om:OMS cd="logic1" name="and" />
                <om:OMA>
                    <om:OMS cd="set1" name="in" />
                    <om:OMV name="y" />
                    <om:OMS cd="setname1" name="R" />
                </om:OMA>
            </om:OMA>
            <om:OMA>
                <om:OMS cd="relation1" name="lt" />
                <om:OMA>
                    <om:OMV name='msdl_umo_F' />
                    <om:OMV name="y" />
                </om:OMA>
            </om:OMA>
            <om:OMV name='msdl_umo_f' />
        </om:OMA>
    </om:OMOBJ>
</msdl:post-condition>
</msdl:post-conditions>
</msdl:body>
</msdl:problem>

```

B Sample MONET Taxonomy

```

<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdf:Description ID="gams">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>GAMS Classification</rdfs:label>
  </rdf:Description>

  <rdf:Description ID="gamsA">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Arithmetic, error analysis</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gams"/>
  </rdf:Description>

  <rdf:Description ID="gamsA1">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Integer</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gamsA"/>
  </rdf:Description>

  <rdf:Description ID="gamsA2">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Rational</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gamsA"/>
  </rdf:Description>

  <rdf:Description ID="gamsA3">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Real</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gamsA"/>
  </rdf:Description>

  <rdf:Description ID="gamsA3a">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Standard precision</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gamsA3"/>
  </rdf:Description>

  <rdf:Description ID="gamsA3c">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>Extended precision</rdfs:label>
    <rdfs:subClassOf rdf:resource="#gamsA3"/>
  </rdf:Description>
</rdf:RDF>

```