

Version: 1.4
Date: March 12th, 2003

MONET Architecture Overview

The MONET Consortium

Deliverable D04 (Public)

Contents

1	Introduction	2
2	MONET and web services: <i>Declaration of Linz</i>	2
3	Architecture of the MONET framework	3
3.1	Entities	3
3.2	The Rôle Of The Broker	4
4	Ontologies	5
4.1	Mathematical Problem Ontology	5
4.2	Mathematical Query Ontology	6
4.3	Mathematical Service Description Language (MSDL)	6
4.4	Mathematical Explanation Ontology	7
5	Use Case	7

1 Introduction

This document describes the rôles and interactions of the different ontologies and components in the MONET framework. This framework aims at providing sufficient tools for the advertisement and discovery of mathematical web services which are the key goals for the MONET project. Section 2 presents the project in the larger context of web services through the *Declaration of Linz*.

In Section 3, we describe the framework architecture and introduce the essential rôle played by the broker in registering and discovering services. We define the main entities involved in the framework and explain the different aspects of the broker.

Section 4 provides an overview of the ontologies used in the system: Mathematical Problem Ontology, Mathematical Query Ontology, Mathematical Service Description Language, and Mathematical Explanation Ontology. These ontologies are used in conjunction with OpenMath to express both concrete mathematical problems as posed by the client, and abstract problems descriptions as advertised by the services. They are also used to represent logistical information. It is important to note, however, that OpenMath could be replaced by other formalisms if needed.

2 MONET and web services: *Declaration of Linz*

On November 11-12 2002 the MONET, SWADE, OpenMath-TN, Calculemus (Fifth Framework), Math-Broker (Austrian Science Foundation) and MathWeb projects met at RISC-Linz to coordinate their approach to deploying mathematical web services. The following key points were agreed on and informally labelled *The Declaration of Linz*:

- Mathematical Web Services are specialised Web Services such that:
 - each service is described by XML-based meta-information;
 - this information is published and is meant to support discovery and query (and not e.g. to describe the error behaviour of a service);
 - this information is sufficient on its own to allow clients to find and connect to appropriate services.
- The necessary architecture to support such services should be built on existing and emerging technologies as far as possible (XML, SOAP, WSDL, RDF, OWL, OpenMath, ...) by using these technologies to construct mathematics-specific definitions and extending them where necessary to provide extra mathematics-related functionality.
- A problem or service description consists of one or more inter-related parts, each pertaining to a particular aspect of the problem or service. Each aspect may have many levels of formalisation from informal human-readable text up to formal machine-processable descriptions with precise semantics.
- Descriptions may be organised or attributed according to multiple classification schemes to help different stages of the discovery process.
- When querying, the client may specify whatever information is available, e.g.
 - a simple taxonomy concept;
 - multiple taxonomy concepts;
 - a formal problem definition.

The broker can return results with varying degrees of confidence and possibly rate them, e.g.

- proved correspondence of input problem versus problem solved ***
- matched taxonomy entries **
- semi-matching entries (more-specific problem solved) *

The MONET architecture should as far as possible satisfy the W3C *Web Services Architecture*, the latest version of which is at <http://www.w3.org/TR/ws-arch/>. We note that the architecture itself is technology-neutral but that the reference architecture is couched in terms of SOAP, WSDL etc.

3 Architecture of the MONET framework

The MONET project aims at providing a framework for the use and discovery of mathematical web services. In this section, we provide a description of the overall picture of the system we envision.

3.1 Entities

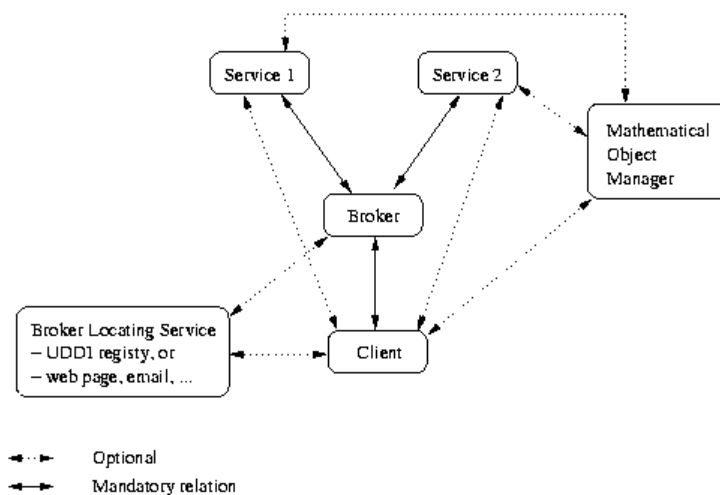


Figure 1: System architecture

In Figure 1, we distinguish five entities:

- **Client.** A client in the MONET framework can range from a servlet or CGI script accessible through a web page to a piece of software. The client initiates a call to a service by contacting a broker and asking it either to discover an appropriate service or to provide access to a specific service.
- **Service.** In this project, we concentrate on a mathematical services accessible as web services. This entity publicises its functionality by registering with a broker, which then negotiates access by potential clients on its behalf.

- **Broker.** Central to the framework, the broker acts as an intermediary between clients and services. On the one hand it assists clients in identifying suitable services for the problem they wish to address, on the other it negotiates access to the service on the client's behalf. It may also perform some housekeeping and lifecycle-management functions for the service. The functionality of the broker is detailed in Section 3.2.
- **Mathematical Object Manager.** Certain mathematical objects may be very large and should not be sent over the network unless it is necessary. They may also exist in different locations using different concrete representations at different times. By registering an object with the Mathematical Object Manager it can be referenced using an abstract URI which is distinct from its physical location, and only retrieved by the client or the service when required.
- **Broker Locating Service.** Although optional in this picture, this service represents the starting point of any process involving this framework. Indeed, before discussing with a broker to locate services, a client needs to know where and how to access the broker. The Broker Locating service can take several forms from a URL written on a piece of paper to a UDDI registry.

3.2 The Rôle Of The Broker

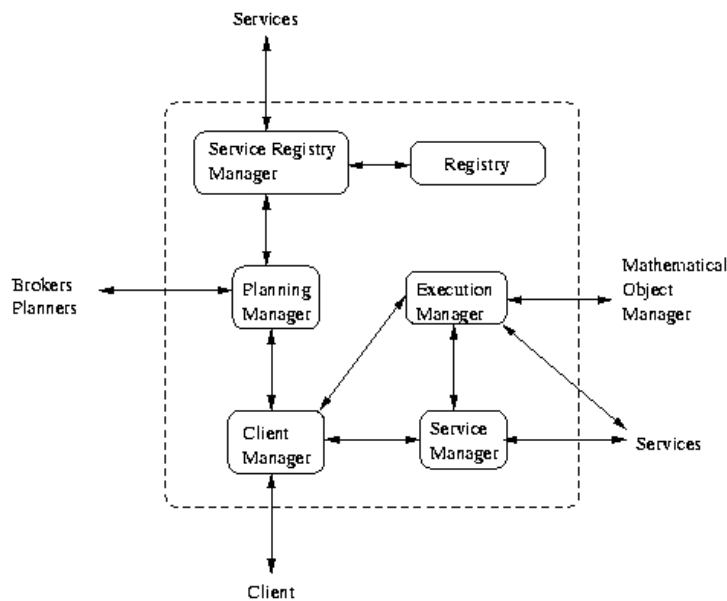


Figure 2: Broker architecture

Figure 2 presents the architecture of the logical entity we call the **broker**. A broker is composed of the following entities that model its functionalities:

- **Client Manager.** This is the front-end module of the framework that receives messages from the client and contacts the Planning Manager for creating an execution plan. The Client Manager may then send the plan back to the client or to an Execution Manager.

- **Service Registry Manager (SRM).** This service handles mathematical service registration and discovery. Discovery is done by consulting the services advertised in the Registry. Semantic matching is done using the ontologies and taxonomies developed in the MONET project.
- **Planning Manager.** The Planning Manager either forwards the query directly to the Service Registry Manager, or it performs some analysis on the problem and consults one or more planners. In this instance, the Planning Manager would also query the Service Registry Manager in order to construct an execution plan based on availability and suitability of services.
- **Execution Manager.** This entity executes plans created by the Planning Manager. For each element of the plan, it chooses a URI in the associated list and contacts the Service Manager to obtain an instance of the service. The Execution Manager could also use the Mathematical Object Manager to store references to intermediate results.
- **Service Manager.** The Service Manager is used to negotiate for a handle on an instance of a mathematical service on behalf of the Execution Manager or the Client Manager.

Note that, at a software level, these entities may be used as distinct services or assembled into any combination. For example, the Client Manager, Planning Manager, and SRM could be combined into one piece of software that would handle and analyze requests to locate appropriate mathematical software.

4 Ontologies

In this section, we describe the ontologies used in the context of the MONET project. These are used in the description of problems, queries, and services, and we expect various aspects of the ontologies to coincide.

4.1 Mathematical Problem Ontology

The mathematical problem ontology is meant to describe the mathematical aspect of a problem posed by the client to the broker or planner. The ontology must satisfy three functional requirements:

1. It must allow the user to pose his or her problem in a natural way (although we do not anticipate humans creating documents in the problem ontology).
2. It must convey enough information for a broker to be able to identify suitable services based on its own knowledge about those which are available plus some inference rules.
3. It must be structured in such a way that the concrete “parameters” which a service requires can be extracted from a problem description. This is necessary to construct XML messages of the form expected by the service.

While it would be possible to define OpenMath CDs containing symbols for representing the canonical interface to different kinds of services, or equivalently designing a set of XML Schemas for the interfaces as WSDL messages, this forces a particular interface on each service and makes it hard to provide extra control over particular algorithms through the provision of extra “parameters” controlling memory usage, iteration limits, discrimination criteria etc. The approach we propose may seem more complicated but we believe it allows for more flexibility.

We note that OpenMath is a declarative language consisting of nouns only, so that e.g. one may create an *integral* but cannot say *integrate*.

A mathematical problem consists of the following identifiable components:

1. A directive such as *find*, *prove* or *decide*. These are meant to describe the kind of action the service will perform and will of course determine the type of answer returned.
2. A set of inputs expressed as OpenMath objects. These could be parts of the problem (e.g. integrand, bounds of integration) or constraints on the solution process (e.g. iteration limits, discrimination criteria) or on the solution itself (e.g. accuracy).
3. A single object representing the output (e.g. an integral).

The task of preparing a problem is similar to that of a phrasebook in that the names of operations and the understanding of their semantics is likely to be hard-coded in the client. Note that there is no canonical representation for a problem since for example both “integrate the function $x \rightarrow \sin(x)$ ” and “integrate f where $f = x \rightarrow \sin(x)$ ” are both equally valid. However the latter may be more flexible when we address the problem of binding to a specific service as we shall see later.

4.2 Mathematical Query Ontology

The mathematical query ontology describes a general query which will be submitted to a broker. In addition to the problem itself (or a general description of it), the query contains relevant information about the client (e-mail address, location, PGP key etc.), and extra constraints on the solution which do not vary between specific problems (user preferences). Ideally a generic mechanism will emerge for dealing with these issues but attempts so far (such as Microsoft Passport) have been a failure.

A mathematical query therefore consists of:

1. Either:
 - (a) A mathematical problem as defined above.
 - (b) A description of a service described in the Mathematical Service Description Language.
2. Extra static (i.e. not problem-specific) information about the user and his or her preferences.

The query is sent to a broker which will return a (possibly ordered) list of services which can handle the problem.

4.3 Mathematical Service Description Language (MSDL)

The Mathematical Service Description ontology defines the metadata which a developer of a service provides when publishing the service to a MONET broker.

MSDL contains mechanisms for categorising services according to the mathematical problems which they solve as well as making statements about both the mathematical (e.g. algorithms used) and non-mathematical (e.g. access policy) properties of a mathematical web service. In addition it has to provide information about how to invoke a service.

The simplest way of providing service descriptions will be by reference to a set of informal descriptions of the problems the service solves. A given set of descriptions will form a taxonomy and a broker will need to understand the relationship between nodes in the taxonomy to be able to do more than the most trivial matching. For example a taxonomy for numerical computation could be based on GAMS which has a tree structure where each node represents a particular mathematical problem and its children represent more specialised instances of that problem.

It is suggested that the organisation of such a taxonomy should be based on information which is expected to be explicit in the mathematical query. Further information about a particular service should be represented separately, in a form which can be used by a broker. So for example a service might declare that it can solve problems of GAMS category G2h3a (optimisation of functions with non-linear equality constraints) and also advertise that it uses the sequential quadratic programming method.

Reference to algorithms could be handled informally by giving the name as a text string, or more formally by reference to an *algorithm registry* where each algorithm would be associated with a URI. It is possible that we will need to define a number of such vocabularies.

The informal descriptions of mathematical problems represented by the taxonomy described above could be associated with formal descriptions described in a *mathematical problem description library* (MPDL) where each problem would be described in terms of its inputs, its outputs and a set of pre- and post-conditions. The labels on the inputs and outputs would have URIs and could be referred to in both the problem description and also the description of the interface to the service. Since this is potentially an important rôle, and in addition defining the pre- and post-conditions for some mathematical algorithms is extremely hard, there is no requirement that the entries in the MPDL be complete. In the time available the MONET project will only be able to develop a rudimentary MPDL.

An MSDL document also contains technical information about the service itself (software used, hardware, algorithmic properties such as accuracy, and so on). The MSDL also includes a Service Interface Description typically of the form of a WSDL document, a Service Binding Description to map actions to operations for each problem type handled by the service, and a Broker Interface that allows a broker to generate service instances, perform housekeeping action etc.

4.4 Mathematical Explanation Ontology

The Mathematical Explanation Ontology allows services to provide a detailed description of the executed plan. The explanation ontology will be used when a query was not successful. In this case, it would be possible to rerun parts of the process and restate the query. Alternatively, an explanation of a successful result could provide the client with useful information to assist with future queries.

5 Use Case

In this section, we describe the process of registering services, making queries, and receiving answers. Here we describe how the entities shown in Figure 1 interact:

- Mathematical services register their offer of a service with the Service Registry Manager which is a logical part of the broker. The offer of a service is formalised through an MSDL document as described in Section 4.3. This document specifies syntactic and semantic information about the service as well as implementation details. The Service Registry Manager may periodically contact its registered services as a house-keeping activity.
- In order to obtain a reference to a MONET broker, a client should use a “locating” mechanism. This can be done through any means available, although we expect a specialised service such as a UDDI registry [19] to be used. The protocol for this activity is not defined in the context of MONET.
- A client may register any objects with the Mathematical Object Manager (MOM) which will issue unique URIs by which these objects can be identified. The services may also use the MOM to retrieve and store objects.

- The client submits a query to the Client Manager to find a mathematical service or services to solve a general or specific problem. The query is formed using the Mathematical Query Ontology for both logistical and mathematical (through the Mathematical Problem Ontology) questions.
- The Client Manager contacts a Planning Manager that analyses the problem and constructs an execution plan or plans (where each step of a plan could refer to a list of zero or more services). The Planning Manager contacts the Service Registry Manager to obtain the list of references of services that match the query both at a mathematical and at a logistical level.
- The Planning Manager returns the execution plans to the Client Manager which forwards them to either the client or an Execution Manager.
- Plans are carried out by the client or the Execution Manager by contacting the Service Managers associated to each of the chosen services. Services are chosen either explicitly by the client or by the Execution Manager on behalf of the client.

Glossary

- **Calculemus.** The CALCULEMUS interest group is a loosely coupled network of research groups and individuals interested in joining forces for the design of a new generation of mathematical software systems and computer-aided verification tools based on the integration of the deduction and the computational power of deduction systems and computer algebra systems respectively. See [2].
- **GAMS.** Guide to Available Mathematical Services. Taxonomy for mathematics. See [8].
- **OpenMath.** Extensible semantic format for unambiguously describing mathematical objects. See [14].
- **OpenMath CD.** OpenMath Content Dictionary. Collection of symbols used to describe mathematical concepts. Each CD is associated to a particular area of mathematics. See [14].
- **OpenMath-TN.** OpenMath Thematic Network. See [14].
- **MathBroker.** Software framework for brokering mathematical services that are distributed among networked servers. See [10].
- **MathWeb.** A project developing a plug-and-play architecture for mathematical software, see [11].
- **MONET.** Mathematics On the NET. The MONET project is a two-year investigation into mathematical web services funded by the European Commission, as part of the Information Society Technologies (IST) Programme of the Fifth Framework. See [12].
- **OASIS.** Organization for the Advancement of Structured Information Standards. See [13].
- **OWL.** Web Ontology Language. “The Web Ontology Language (OWL) is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications.” (from the W3C page). See [15].
- **RDF.** Resource Description Framework. “Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web.” (from the W3C page). See [16].
- **RDFS.** Resource Description Framework Schema. See [17].
- **SOAP.** Simple Object Access Protocol. SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. See [18].
- **SWADE.** Semantic Web Advanced Development Europe, a project supporting the spread of semantic web applications.
- **UDDI.** Universal Description, Discovery and Integration of web services. Framework used for the discovery of web services matching a given query. See [19].

- **URI**. Uniform Resource Identifier. The generic set of all names/addresses that are short strings that refer to resources (*from the W3C page*). See [1].
- **URL**. Uniform Resource Locator. An informal term (no longer used in technical specifications) associated with popular URI schemes: http, ftp, mailto, etc (*from the W3C page*). See [1].
- **W3C**. World Wide Web Consortium. See [20].
- **WSDL**. Web Service Description Language. WSDL is an XML format for describing network services as a set of endpoints operating on messages which contain either document-oriented or procedure-oriented information. See [21].
- **XML**. eXtensible Markup Language. XML is a text format derived from SGML. It was originally designed for large-scale electronic publishing, and plays an important rôle in the exchange of a data on the Web. See [22].

References

- [1] W3C, *Web Naming and Addressing Overview*, <http://www.w3.org/Addressing>
- [2] *The CALCULEMUS Project*, <http://www.calculumus.net>
- [3] Mike Dewar, *Mathematical Explanation Ontology*, The MONET Consortium, Deliverable D07. Available from <http://monet.nag.co.uk>
- [4] Olga Caprotti, *Mathematical Problem Ontology*, The MONET Consortium, Deliverable D11. Available from <http://monet.nag.co.uk>
- [5] Stephen Buswell, Phillip Lord, *Mathematical Query Ontology*, The MONET Consortium, Deliverable D13. Available from <http://monet.nag.co.uk>
- [6] Stephen Buswell, Olga Caprotti, Mike Dewar, *Mathematical Service Description Language: Final Version*, The MONET Consortium, Deliverable D14. Available from <http://monet.nag.co.uk>
- [7] *Dublin Core Metadata Initiative*, <http://dublincore.org>
- [8] National Institute of Standards and Technology, *GAMS: Guide to Available Mathematical Software*, <http://gams.nist.gov>
- [9] *Grid Economic Services Architecture*, http://www.gridforum.org/3_SRM/gesa.htm
- [10] RISC-Linz, *Mathbroker: A Framework for Brokering Distributed Mathematical Services*, <http://poseidon.risc.uni-linz.ac.at:8080>
- [11] The MathWeb Project <http://www.mathweb.org/>.
- [12] The MONET Consortium, *MONET Home page*, <http://monet.nag.co.uk>
- [13] OASIS, *Organization for the Advancement of Structured Information Standards*, <http://www.oasis-open.org>
- [14] OpenMath, *The OpenMath website*, <http://www.openmath.org>
- [15] W3C, *Web Ontology Language (OWL) Guide Version 1.0*, <http://www.w3.org/TR/2003/WD-owl-guide-20030210>

- [16] W3C, *Resource Description Framework*, <http://www.w3.org/RDF>
- [17] W3C, *RDF schema*, <http://www.w3.org/TR/rdf-schema>
- [18] W3C, *Simple Object Access Protocol (SOAP) 1.1*, <http://www.w3.org/TR/SOAP>
- [19] OASIS, *Universal Description, Discovery and Integration of Web Services*, <http://www.uddi.org/>
- [20] W3C, *World Wide Web Consortium*, <http://www.w3.org>
- [21] W3C, *Web Services Description Language (WSDL) 1.1*, <http://www.w3.org/TR/wsdl>
- [22] W3C, *Extensible Markup Language (XML)*, <http://www.w3.org/XML/>