# IAMC Architecture and Prototyping: A Status Report

P. Wang*, S. Gray†, N. Kajler‡, D. Lin§, W. Liao, X. Zou
Institute for Computational Mathematics
Kent State University
Kent, Ohio 44242, U.S.A.

**Abstract**

Internet Accessible Mathematical Computation (IAMC) is a distributed framework to supply mathematical computing powers widely over the Internet. Presented are conceptual and experimental work on the IAMC architecture, a client prototype (*Dragonfly*), client GUI, a server prototype (*Starfish*), the *Mathematical Computation Protocol* (MCP), mathematical data encoding formats, and external compute engine interface.

## 1 Background

Making mathematical communication easy over the Internet has great potential. Ad hoc methods have been used to display mathematical formulas in Web pages and to make simple mathematical computations accessible via CGI programs or X Windows [29]. Yet, a general and effective system for accessing, producing, and delivering mathematical content is till the subject of research and development.

Investigators at the W3 Consortium and elsewhere are working to make *publishing* mathematical materials on the Web easy. MathML [4] defines an SGML language for markup of mathematical expressions. Both presentation (display layout) and content (computation semantics) markup are supported.

The IBM digital publishing group has released the experimental *Techexplorer* [9], a Web browser plug-in that dynamically formats and displays documents containing scientific and mathematical expressions coded in TEX/LATEX. Some MathML is also supported. Techexplorer also

*Work reported herein has been supported in part by the National Science Foundation under Grant CCR-9721343 and in part by the Ohio Board of Regents Computer Science Enhancement Funds.

†Ashland University, Ashland, OH USA.
‡Ecole des Mines de Paris, Paris, France.
§Institute of Systems Science, Academia Sinica, Beijing, China.

allows a user to send expressions to a fixed compute server for evaluation. MathType [25], from Design Science Inc., supports interactive creation of mathematical notations for web pages and documents. The same company also offers WebEQ [25] that provides a Java applet to display WebTeX and MathML in a browser. The W3 Consortium's Amaya Web browser demonstrates a prototype implementation of MathML which allows users to browse and edit Web pages containing mathematical expressions [20]. Together with the rest of the Web page, these expressions are manipulated through a WYSIWYG interface. The increasing acceptance and software support for MathML were evident at the recent *MathML International Conference 2000* [26].

Andrew Solomon and others have built JavaMath [28], a free software enabling Java-based mathematical programs to use the computational capabilities of existing compute engines. JavaMath can be used for stand-alone applications and for construction of Internet based client-server systems and Web pages.

## IAMC

*Mathematical content viewing* on a Web page is static. On the Internet, end-users and applications can make good use of *dynamic access to mathematical computing*. "Internet Accessible Mathematical Computation" was the focus of discussion at the 1999 *IAMC Workshop*, part of the International Symposium on Symbolic and Algebraic Computation (ISSAC'99), held in late July in Vancouver, Canada. The full-day workshop highlighted the importance of research and development in making mathematical information and computation easily available in the new communication age [17, 23]. For more background and related activities, please refer to the Proceedings of the IAMC'99 Workshop [30], the IAMC homepage [27], and the Workshop on *The Future of Mathematical Communication* [31].

A recent research focus at the Institute for Computational Mathematics (ICM/Kent) has been on efficient encoding and transmission of mathematical data among heterogeneous compute engines, and on the initial design and implementation of a distributed IAMC framework [24, 22].

As a distributed system IAMC, has these potential applications:

- Easily accessing remote mathematical computations
- Using remote mathematical databases
- Making parallel/super computing accessible
- Distance learning involving mathematics
- Internet/Web assisted mathematical education
- Establishing *problem solving environments* (PSE)

Many R&D efforts that facilitate our research can be found on the IAMC homepage. For data representation, in addition to MathML/WebEQ, the following are of particular interest:

- MP — the *Multi Protocol* [10, 11, 12] a mathematical data encoding/exchange format that uses a binary encoded annotated parse tree for efficiency.

- OpenMath — a protocol for representing semantically rich mathematical objects, allowing them to be exchanged between programs, stored in databases, and published in electronic form. The central notions in OpenMath [1] are Content Dictionaries, providing standard and application-defined vocabularies and definitions, and Phrasebooks, which provide translation between application concepts and those defined in the relevant Content Dictionaries. Translated data may be exchanged in either binary or XML form. The OpenMath Website `www.openmath.org` has links to software and activities.

Several other (early) works related to the exchange of mathematical data between scientific applications are reported in [2] and in [14]. Notable implementations of distributed architectures that provide some exchange of mathematical expressions include Polylith [18], CaminoReal [3], SUI [8], DSC [7], CAS/PI [13], and CC [6].

Based on our experiences and applying emerging technologies, we pursue the IAMC project with these goals:

- To make math-oriented data and services easily and widely accessible on the Internet in many contexts – directly, via the Web, by email, for distance learning, etc.
- To support interactive use of user-designated remote *compute servers* almost as if they were local programs
- To provide effective and efficient communication of mathematical data over the Internet
- To allow exchange and further processing of computational results among different compute servers

Reported here are advances in the design, architecture, protocol, data encoding, and software prototyping for the distributed IAMC framework.

# 2 IAMC Architecture

IAMC is a distributed framework to make mathematical computing easily accessible and interactively usable on the Internet [21, 22]. Figure 1 shows the overall IAMC architecture.
IAMC consists of these components:

1. IAMC client (Icl) — An end-user agent for accessing services provided by any IAMC server.
2. IAMC server (Isv) — A program to provide mathematical computation powers through the IAMC protocol layers. An Isv may or may not employ an external compute engine to perform mathematical computations.
3. Protocol layers — IAMC clients and servers are connected by well-defined protocols (Figure 2):

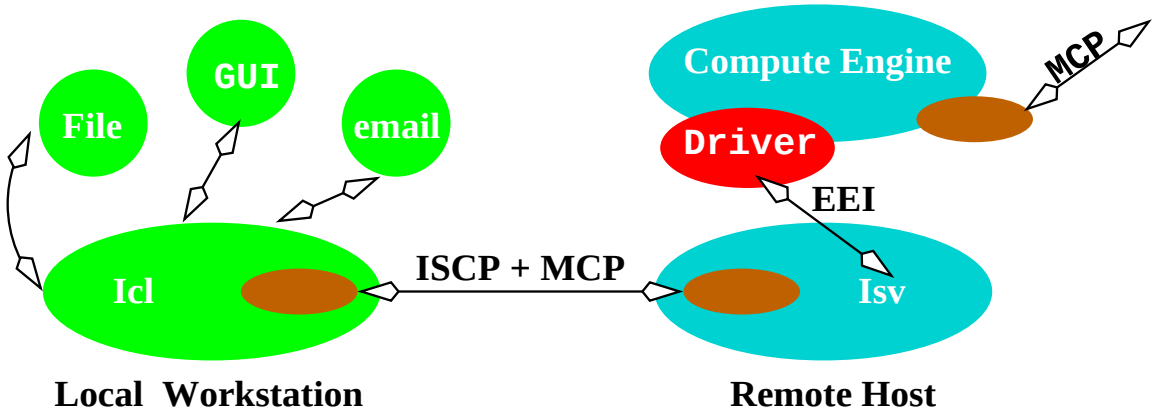


Figure 1: IAMC Architecture

- The Session Control layer provides the IAMC Session Control Protocol (ISCP) for asynchronous service invocation, callback, interrupting server-side computations, and IAMC URL resolution.
- The MCP layer (Mathematical Computation Protocol) supports interactive mathematical computation based on a persistent connection (Section 5).

4. Mathematical Data Encoding — Standard and user-defined mathematical data encoding formats can be used (Section 6). Standard formats allow heterogeneous mathematical programs to interoperate. Custom applications can easily build IAMC clients and servers to use any desirable encoding. The IAMC framework allows plug-in format converters, on the client and the server side, to support additional formats.

   Automatic negotiation between the Icl and Isv at the beginning of a computation session determines the encoding(s) used.

5. The External Engine Interface (EEI) — A specification and API implementation (Section 7) for binding existing compute engines to IAMC servers.

| IAMC Application Layer |
|---|
| IAMC Session Control Layer |
| Mathematical Computation Layer |
| Transport Layer |

Figure 2: Protocol Layers

Dragonfly, a client prototype, and Starfish [15], a server prototype, are implemented in Java. The prototypes allow us to test the effectiveness of IAMC and evaluate/improve the protocols.

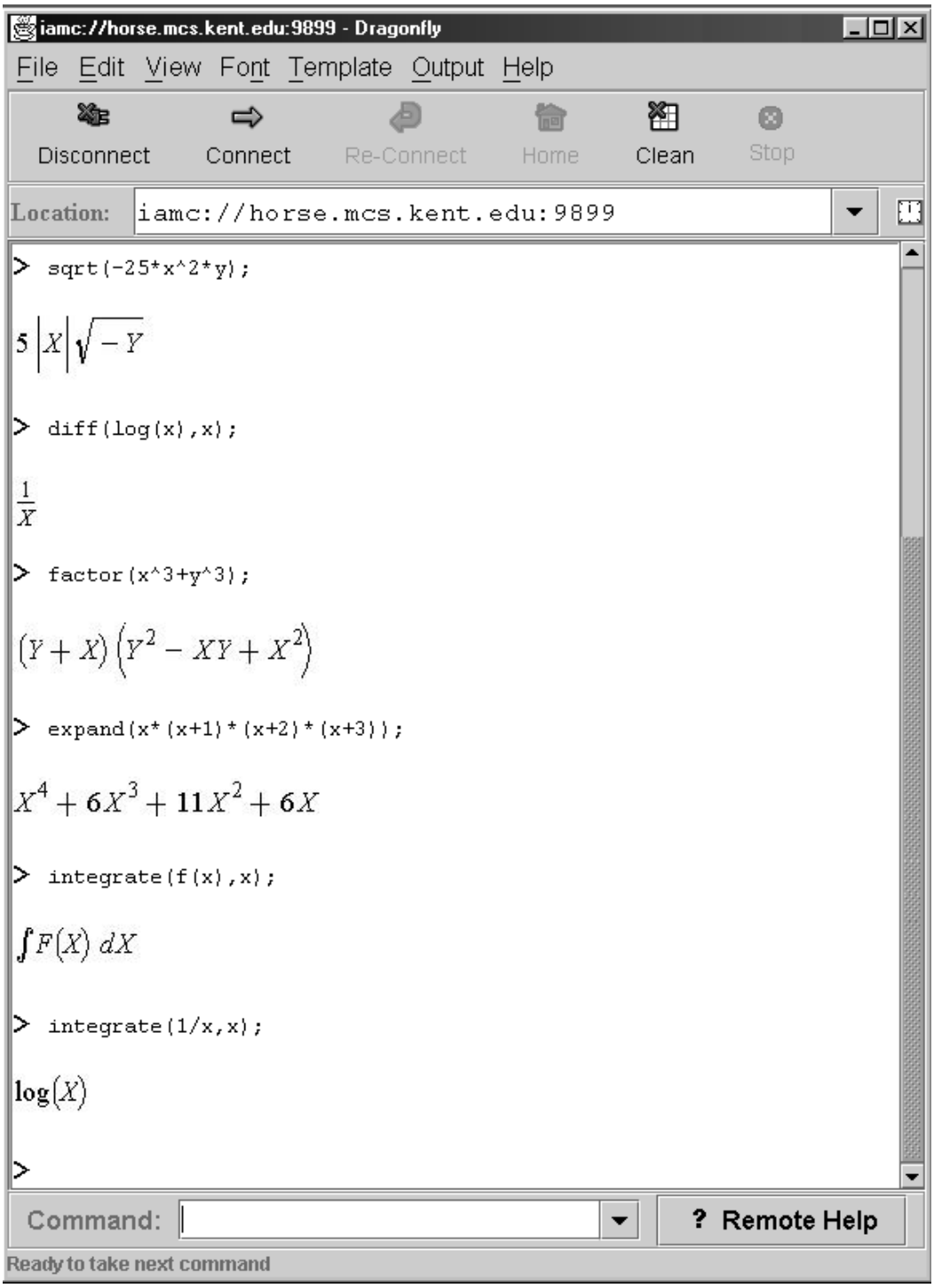


Figure 3: *Dragonfly* User Interface

# 3 Client Prototype: Dragonfly

Dragonfly [16] is a prototype IAMC client implemented in Java. It provides a convenient GUI for accessing computations supported by any IAMC compliant server. Dragonfly can

1. Connect to and communicate with any user-specified IAMC server via the ISCP-MCP protocol
2. Obtain capability and usage documentation from the IAMC server and make them available to the user
3. Receive computational, control and help commands from the user and send them to the server
4. Parse infix mathematical expressions entered by the user
5. Receive results from the server
6. Display mathematical symbols and expressions in textbook-like fashion
7. Select subexpressions with the mouse
8. Save mathematical results in files under well-defined formats such as MP, MathML, and OpenMath
9. Plot mathematical curves and surfaces
10. Present command templates from the server to the user as required
11. Display server dialog and relay user data thus obtained back to the server
12. Encode/decode mathematical and graphical data

Furthermore, Dragonfly is designed for extensibility allowing easy addition of features and functionalities.

Built on the Java 2 platform, Dragonfly employs:

- Swing to implement the GUI and the HTML-based help facility
- Java 2D to support 2-dimensional and 3-dimensional mathematical function plotting and manipulation
- WebEQ to render mathematical expressions in MathML presentation code
- XML to help encode/decode mathematical graphing data

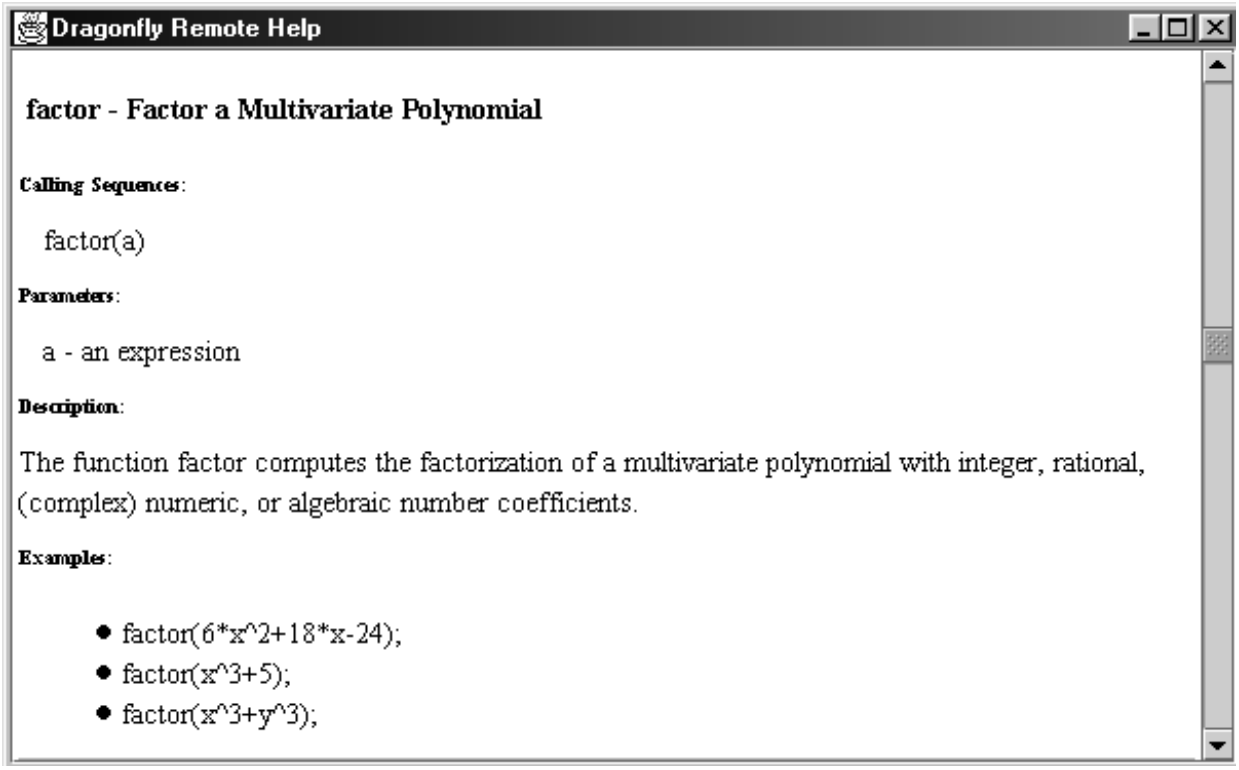


Figure 4: *Dragonfly* Remote Help Window

## Dragonfly Usage

When started, Dragonfly displays a browser window containing a *Location Box*, an *Output Panel* and a *Command Box*.

A user selects or types a desired IAMC URL in the Location Box:

`iamc://`*host*`:`*port*`/`*isv_name*

The URL identifies an IAMC server anywhere on the Internet by giving its name, host, and port. Dragonfly proceeds to connect with the given IAMC server. If the connection succeeds, a welcome message from the IAMC server (Isv) appears in the Output Panel. Otherwise, an error message appears.

User input to the Command Box will be echoed in the Output Panel and sent to the connected Isv. The computational result obtained is displayed in the Output Panel in 2-dimensional format.

An IAMC browser may connect to any compliant server with different capabilities. A way to obtain the supported commands and their usages is essential. The *Remote Help* button gets help on computational commands and their usage from the Isv. Dragonfly does this via a help request to the remote Isv.

As soon as Dragonfly connects to an Isv, the Remote Help button is enabled. A click on this button obtains the Isv-supplied help document (Figure 4) in HTML or other accessible formats. If

the Command Box is empty, the Remote Help button produces a list of available commands. If the Command Box contains a command, it displays details for that command.

The Command Template effectively guides the user to enter commands and arguments correctly. Simply put, a command template is a dialog for users to fill in the arguments needed for a mathematical operation. Figure 5 is a template for an integration operation.

A user may interrupt and abort a computation that is taking too long or becomes unnecessary. The MCP protocol supports such computation controls.

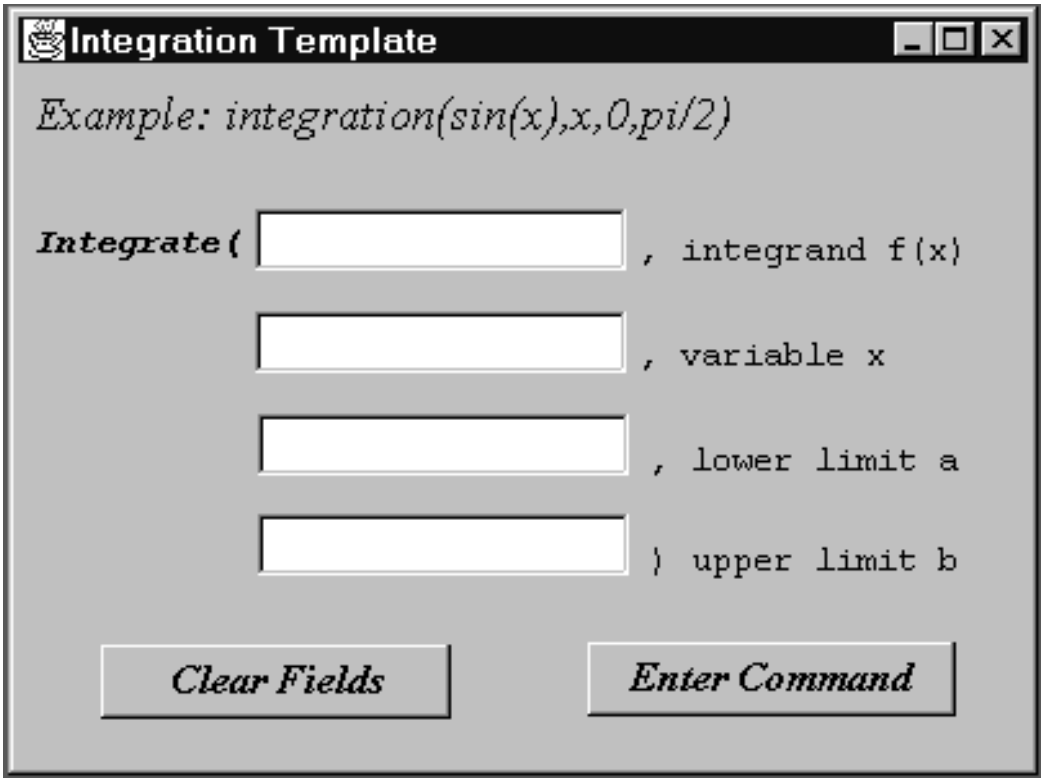


Figure 5: *Dragonfly* Command Template

## Displaying Expressions and Graphs

Dragonfly leverages the WebEQ class library to render mathematical expressions in 2-dimensional form. Sending the correct MathML presentation code to WebEQ produces the desired 2-dimensional display. Computational results, from IAMC servers, can be in any desired format: plain text, MathML, MP, OpenMath, Webtex, or any other format as specified in the MCP message header. MathML presentation or Webtex coded results can be passed to WebEQ directly for display. Other formats can be converted to MathML first.

Dragonfly also supports plotting of mathematical curves and surfaces. A separate window displays and interactively manipulates the graphics. The Dragonfly plotting module displays data

from any Isv in MathML/Graph, an XML-defined representation compatible with MathML [4, 26]. Figure 6 and 7 show the 2D and 3-D Graph Windows.

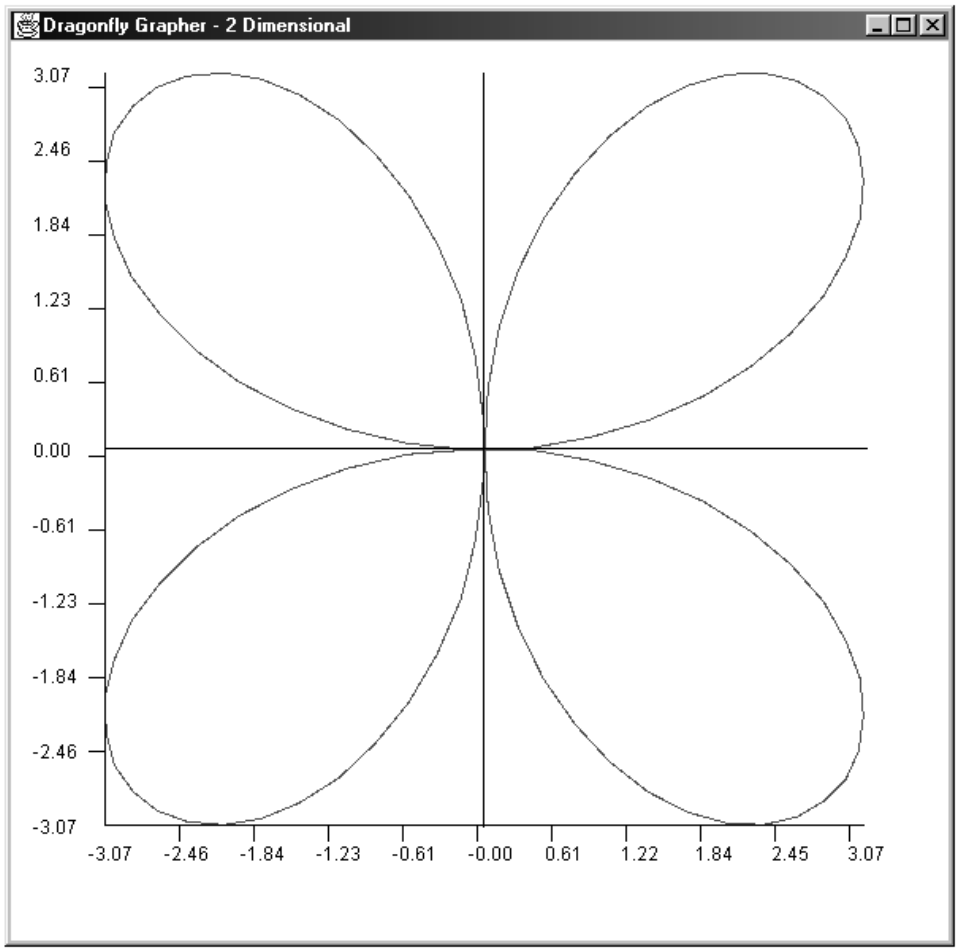


Figure 6: *Dragonfly* Plotting

The current version of Dragonfly connects to one server at a time. The ability to connect to multiple servers concurrently is important and will be added in the next version.

## Accessibility from the Web

Currently, Dragonfly is a stand-alone application that can be launched by the user. It would also be nice if Dragonfly could be activated by a hyperlink in a web page. For this purpose, well-established conventions on launching Web browser helper applications can be followed.

An IAMC-aware browser may launch its IAMC helper (Dragonfly) when following an IAMC URL. Alternatively, a hyperlink pointing to a text file containing one or more IAMC URLs can be used. A click on such a link gets an HTTP response with the content type, say,

```
Content-Type application/x-iamcurl
```

The `x-iamcurl` content type tells the browser to launch its user-configured IAMC helper application, handing to it the target URLs and optional initial display data. The latter arrangement does not require special browser treatment. It is worthwhile to make Dragonfly into a browser helper/plug-in.

Web pages containing formulas or equations can also make the mathematical expressions come alive through server-side programming (e.g. CGI, PHP, Servlet) that taps into local or remote IAMC servers. The MCP protocol is designed to also support such applications. But we still need good library modules, in the form of compact IAMC clients, that enable Web server-side programs to easily request and obtain mathematical results from IAMC servers. The results returned in MathML are readily included in a generated HTML document.

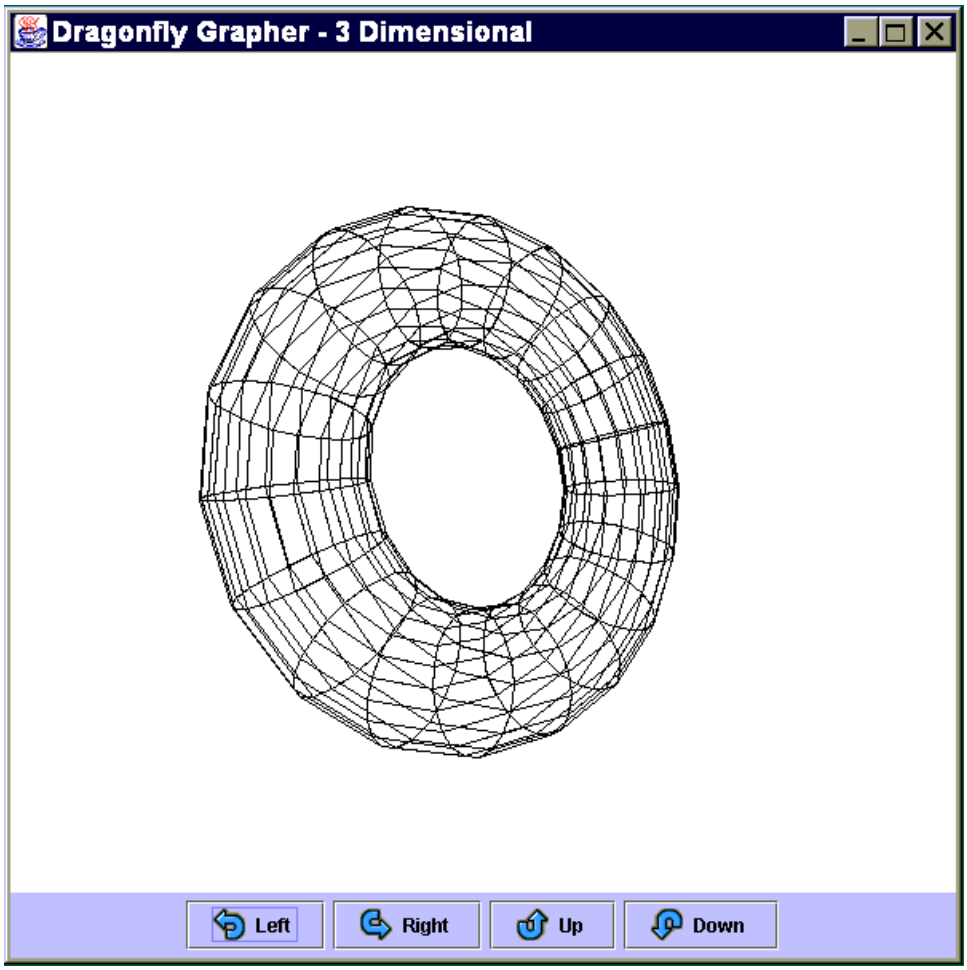


Figure 7: *Dragonfly* 3D Graphing

# 4 Server Prototype: Starfish

The IAMC server (Isv) is the program spawn in response to a TCP connection request, from an Icl, to the IAMC daemon listening at a specific port. Once connected, the Icl and Isv conduct business following the IAMC protocols. When the computation session is done, the Isv terminates.

Starfish is our IAMC server prototype, also implemented in Java. First, let's summarize the functional requirements. An IAMC server must

- Be MCP compliant
- Be customizable by configuration file
- Maintain the state of computation sessions
- Launch/control external compute engines dynamically
- Handle MCP control and computation requests either synchronously or asynchronously
- Perform callback (dialog) requests to the client
- Support client-generated interrupts
- Supply server administrative functions
- Allow added modules for extensibility

Starfish is implemented in Java. Through multi-threading, it can support multiple Icl connections concurrently. Starfish also has an administrative GUI for configuration and activities monitoring.

## Dynamic Service Binding

To dynamically bind compute engines, Starfish uses a property file to specify the available engines. The property file allows Starfish to locate and load *external engine drivers* (Section 7) for particular services. The Java Reflection API makes it rather simple to dynamically load classes for the engine driver.

Starfish uses an engine driver to access Maxima at ICM/Kent which has been modified to receive input in MathML (content codes) and to produce both MathML content and presentation code as output. The development of engine drivers for Maple and Mathematica is underway.

# 5 The MCP Protocol

To connect any IAMC server and client, we need a well-designed protocol that makes accessing/providing mathematical computation on the Internet effective and efficient. MCP, the Mathematical Computation Protocol, aims to be the core of such a protocol. Important characteristics of MCP include:

- Handling one-time computation requests and persistent computation sessions
- Placing no restrictions on content types or mathematical data representations
- Supporting the special needs of mathematical computations
- Permitting both server and client to send requests and to return responses
- Providing for both synchronous and asynchronous message exchanges
- Distinguishing protocol control from computation control
- Being simple and effective
- Allowing easy extensions

As described in its original design [22], MCP uses HTTP-style requests and responses to support mathematical computation sessions. Each MCP message consists of a *header* and an optional *body*. Each header entry is a *key-value pair* on one line terminated by a NEWLINE or a CR-NEWLINE. The header and body are separated by an empty line. The first line of an MCP message is a control line that specifies the *message type*, the *message class*, and a sequence number. The MCP message control line takes one of two forms:

`Request` *Class seqNo*

or

`Response` *Class seqNo statusCode* [ *statusString* ]

MCP requests belong to different classes that contain request methods to support well-defined operations. An MCP request identifies the request class and a sequence number. The request sequence number *seqNo* is generated by the request originator. A response indicates the class and *seqNo* to identify the request to which it is responding.

## MCP Message Classes

Following the OO philosophy, each MCP request class defines a set of methods that can be invoked. Standard MCP request classes include:

- `Initialization` — The initialization class supports session creation and configuration right after client-server connection
- `Control` — The control class supplies MCP protocol control and management methods for both the client and server side
- `Computation` — A class of server-side operations to perform application supported computations

- `Dialog` — A class of client-side methods to solicit information from the end user

As in HTTP, status codes and strings are used in responses to indicate error and exceptional conditions.

The `Method` header field indicates the class method to handle the request. The other header fields and any message body can be considered arguments to the method. A synchronous request returns with the response whereas an asynchronous request returns immediately and any response will be delivered when available.

A sample initialization request from the server to the client is

```
Request Initialization S1
Method: canDo
Version: MCP/1.0
Server-name: PolyFactor
Content-length: 128

"Calculus"=integrate diff taylor...
"Linear Algebra"=vector matrix determinant ...
"Complex Analysis"=absValue conjugate realPart ...

Response Initialization S1 100 OK
```

where the server informs the client what computation commands it supports. The `canDo` format

"*Area_name*"=*command1 command2* `...`

gives IAMC servers the freedom to name the areas and commands it supports. The `canDo` data can be collected by an IAMC search engine to make locating IAMC servers easy.

Other initialization methods include: `welcome`, `connect`, `version-negotiation`, `identificatio` `content-negotiation`, `dictionary-map`, `end-initialization`.

The `dictionaryMap` method allows the client and server to set up a *CD map*, an association of mathematical content dictionary (CD) names to integer indices. The indices can then be used in a mathematical encoding to attach CD tags to expressions. This scheme permits the use of any CD with a unique string name without bloating the mathematical encoding.

The detailed specification work on MCP is on-going.

# 6 Mathematical Data Encoding

The MCP protocol allows any content type to be transported within the MCP message envelop. In principle, IAMC servers and clients can use any mathematical data encoding they wish. In practice, clients and servers should use standardized and widely understood mathematical representations.

It is reasonable to assume, at this point, that MathML will be widely used and become a standard. Hence, MCP has some built-in support for MathML. It has a converter module MathML_MP

that can encode MathML content encoding into and from MP [11]. Because MP is a very compact binary encoding, it can perform *MathML compression* before network transmission. Any content dictionary information contained in MathML will be coded efficiently using *CD map* indices. Xiao Zhu is working on a MathML_MP converter in Java based on JMP, an object-oriented implementation of MP in Java [12, 19].

Here is the typical data flow we see through the IAMC system:

1. User enters input in Fortran-ish infix notation.
2. Icl parses user input into MathML content encoding.
3. MathML content data is sent through MCP with automatic MP compression/decompression.
4. Isv sends MathML content encoding to compute engine driver via EEI (Section 7).
5. Isv receives MathML content and/or MathML presentation result from engine driver.
6. The result is sent back to the IAMC client via MCP.
7. The Icl GUI gives the MathML presentation code to a renderer for display.

## MathML/Graph

In addition to expressions and formulas, IAMC also supports plotting of mathematical curves and surfaces. To represent graphing requests and results we devised an XML-based format called *MathML/Graph* which can be viewed as a *graphing extension* to MathML.

With XML, we define several new tags, including `<mathGraph>`, `<mathPlot>`, `<range>`, and `<coordinates>`. For issuing plotting requests, we use `<mathPlot>`. For representing a plotted graph, we use `<mathGraph>`.

Here is an example graph of $y = \cos(x)$ for the interval $0 \leq x \leq \pi$ :

```
<mathGraph name="mycosine" type=rectangular dimension=2
           plotcolor=black bgcolor=white>
  <equations type=normal>
     <apply> <eq/> <ci>y</ci>
             <apply> <cos/> <ci>x</ci> </apply>
     </apply>
  </equations>
  <range>
     <var type=independent> <ci> x </ci> </var>
     <lower> <cn type="integer">0</cn> </lower>
     <upper> <apply> <times/> <cn type="integer">2</cn>
                     <ci type="constant">&pi;</ci>
             </apply>
     </upper>
```

```
  </range>
   ...

   <coordinates type="rectangular" dim="2"
            valuetype="float" point="x,y" points=40>
     <cn type="integer">0</cn>
     <cn type="integer">1</cn>
     <cn>0.16041128085776021</cn>
     <cn>0.98716167535309518</cn>

     ...

   </coordinates>
</mathGraph>
```

Some of the elements in `<mathGraph>` are optional. With all coordinate points supplied, the equations are not necessary if they are not to be displayed on the graph. Equations can be of type `parametric` and, together with ranges, can be used in `<mathPlot>` requests. A curve or surface may involve many points and the representation can become huge. Binary encoding and data compression techniques [5] can be applied to significantly reduce the size. MathML/Graph can be compressed by the MathML_MP converter we are developing (Section 6). Work on MathML/Graph, which is useful in IAMC and independently, is on-going.

# 7 External Engine Interface

To make IAMC valuable, many useful computational services must be made available. Hence, it is important to be able to create IAMC services easily. The generic Starfish runs on any Java platform and goes a long way in this direction. But connecting to external computation engines must also be made easier. For example, JavaMath [28], supplies a uniform API to make it easy for Java-based programs to access existing compute engines.

We are developing an External Engine Interface (EEI) specification to govern how IAMC servers and external compute engines interface. Studying existing engines, we noted:

- A compute engine may execute some initialization commands before it begins to perform user-requested computations. Usually, the initialization sets up the input/output modes, processes customization/configuration parameters, etc.
- A compute engine may execute in several different modes such as normal mode and debug mode.
- A compute engine may send a prompt when it is ready for input. Different compute engines use different prompts. The prompt may also indicate the current engine status/mode.

- A compute engine normally takes one command at a time. Each command has a terminating character.
- A compute engine returns a result or an error message for a command. Results may contain text, mathematical expressions, and graphics. The beginning and end of the result are well-defined.
- A compute engine may ask for extra information from the user in order to complete a particular computation.
- A compute engine may support several types of user-generated interrupts.
- A compute engine may keep track of commands and results in generated variables.
- Help information and documentation can come from the engine or some other source.

The EEI being developed [15] is language independent and modeled after database connectivity systems such as Microsoft's ODBC and Sun's JDBC. The EEI specifies a standard API to create connections, obtain capabilities, retrieve help/documentation, formulate and execute commands, receive results, etc., for external compute engines. Developers can implement *EEI drivers* in different programming languages to support different applications. The engine end of a driver needs to be customized to control different compute engines. A compute engine designed to go online may support EEI directly, bypassing the EEI driver.

An IAMC server and the external compute engine can run as separate processes on the same host and perform interprocess communication, or reside in the same local area network and perform network communications. To launch a particular external compute engine, an Isv first invokes the correct EEI driver. The EEI driver in turn starts the target compute engine and controls it to perform computations. For prototyping, we have written an EEI driver in Java for MAXIMA (Figure 8). EEI drivers for Maple and Mathematica are being developed. Experiments will lead to further improvements in the EEI specification and driver implementation.

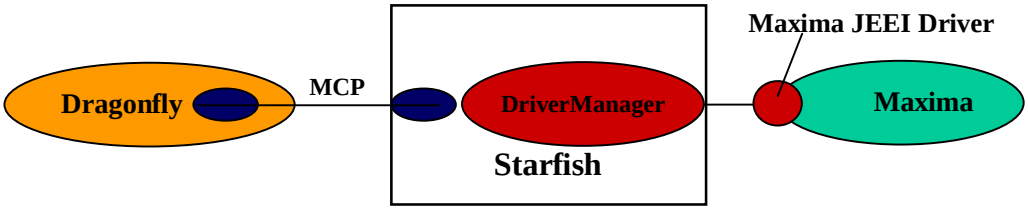


Figure 8: IAMC Prototyping Structure

# 8 Further Work

The IAMC project has come a long way since Norbert, Paul, and Simon started collaborating on the Multi Protocol (MP). The IAMC group at ICM plans to demonstrate the prototypes Dragonfly and Starfish at ISSAC'2001. Work on the specification and reference implementation of MCP, the specification and XML implementation of MathML/Graph, and the MathML_MP converter is ongoing. We are also considering a Web-based search engine for end-users to locate IAMC servers that support particular computations.

One of our long-term goals is to develop a large collection of reusable Java classes covering all aspects of IAMC implementation and, from there, to develop a collection of extensible Java frameworks to support the rapid development of clients, servers and external engine interfaces for particular compute engines. The development of Dragonfly and Starfish is a first step in this direction.

We hope to collaborate with more people in different parts of the world. The goal is to make mathematical computations easily accessible in many contexts on the Internet. Our progress can be tracked on the IAMC project homepage at Kent/ICM [30].

# References


[1] J. Abbott, A. Diaz. and R. S. Sutor, "Report on OpenMath", ACM SIGSAM Bulletin (Mar. 1996), 21-24.

[2] D. Arnon, "Report of the Workshop on Environments for Computational Mathematics", ACM SIGSAM Bulletin, volume 21, number 4, pp. 42-48.

[3] D. Arnon, R. Beach, K. McIsaac, C. Waldspurger, "CaminoReal: An Interactive Mathematical Notebook", In van Vliet, J. C., editor, Proc. of EP'88 International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice, France, pp. 1-18. Cambridge University Press, 1988.

[4] R. Ausbrooks, S. Buswell, S. Dalmas, S. Devitt, A. Diaz, R. Hunter, B. Smith, N. Soiffer, R. Sutor, S. Watt, "Mathematical Markup Language (MathML) Version 2.0", (`http://www.w3.org/TR/2000/CR-MathML2-20001113/`), Nov. 2000.

[5] R. Avitzur, O. Bachmann, N. Kajler, "From Honest to Intelligent Plotting", Proceedings, ISSAC'95, ACM Press, pp. 32-41, 1995.

[6] S. Dalmas, M. Gaëtano, A. Sausse, "Distributed Computer Algebra: the Central Control Approach", Proceedings, 1st Intl. Symp. on Parallel Symbolic Computation (PASCO'94), volume 5 of Lecture Notes Series in Computing, Hagenberg/Linz, Austria. World Scientific, 1994.

[7] A. Diaz, E. Kaltofen, K. Schmitz, T. Valente, M. Hitz, A. Lobo, and P. Smyth, "DSC: A System for Distributed Symbolic Computation", Proceedings, ISSAC'91, ACM Press, pp. 323-332, 1991.

[8] Y. Doleh, "The Design and Implementation of a System Independent User Interface for an Integrated Scientific Computing Environment", Ph.D. dissertation, Dept. Mathematics and Computer Science, Kent State University, May 1995.

[9] S. S. Dooley, "Coordinating Mathematical Content and Presentation Markup in Interactive Mathematical Documents", Proceedings, ISSAC'98, ACM Press, 1998.

[10] S. Gray, N. Kajler and P. S. Wang, "MP: A Protocol for Efficient Exchange of Mathematical Expressions", Proceedings, ISSAC'94, ACM Press, pp. 330-335, 1994.

[11] S. Gray, N. Kajler and P. S. Wang, "Design and Implementation of MP, a Protocol for Efficient Exchange of Mathematical Expressions", *Journal of Symbolic Computation*, Vol. 25, number 2, Feb. 1998, pp. 213-238.

[12] S. Gray, L. Tong, and P. S. Wang, "The MP Encoding for Distributed Mathematical Computations: An Object-oriented Design and Implementation", Proceedings, 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), June 28 - July 1, 1999, Monte Carlo Resort, Las Vegas, Nevada.

[13] N. Kajler, "CAS/PI: a Portable and Extensible Interface for Computer Algebra Systems", Proceedings, ISSAC'92, ACM Press, pp. 376-386, 1992.

[14] N. Kajler and N. Soiffer, "A Survey of User Interfaces for Computer Algebra Systems", *Journal of Symbolic Computation*, vol. 25, number 2, Feb. 1998, pp. 127-160.

[15] W. Liao and P. S. Wang, "Building IAMC: A Layered Approach", Proceedings, International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00), June 26-29, 2000, Csrea Press, pp. 1509-1516 (also: TR ICM-200003-0002, `http://icm.mcs.kent.edu/reports/index.html`).

[16] W. Liao and P. S. Wang, "Dragonfly: A Java-based IAMC Client Prototype", Proceedings of the 4th Asian Symposium on Computer Mathematics (ASCM'2000), Chiang Mai, Thailand, Dec. 17-21, 2000.

[17] S. Linton and A. Solomon, "GAP, OpenMath, and MCP", Proceedings, IAMC'99 Workshop, July 1999,

[18] J. M. Purtilo, "A Software Interconnection Technology to Support Specification of Computational Environments", PhD thesis, Dpt. of Computer Science, University of Illinois at Urbana-Champaign, 1986.

[19] L. Tong, "Object Oriented Design and Implementation of MP", Master's Thesis, Department of Mathematics and Computer Science, November, 2000.

[20] I. Vatton and V. Quint, “Editing MathML on the Web with Amaya”, W3C/INRIA, MathML International Conference 2000, UIUC Illinois USA, Oct. 20-21, 2000.

[21] P. S. Wang, “Internet Accessible Mathematical Computation”, Proceedings of the 3rd Asian Symposium on Computer Mathematics (ASCM’98), Lanzhou University, Lanzhou P. R. China, Aug. 6 1998, pp. 1-13.

[22] P. S. Wang, “Design and Protocol for Internet Accessible Mathematical Computation”, Proceedings, ISSAC’99, ACM Press, pp. 291-298, 1999.

[23] A. Weber and W. Küchlin, “A Framework for Internet Accessible Software Components for Scientific Computing”, Proceedings, IAMC’99 Workshop, July 1999, `http://icm.mcs.kent.edu/research/iamc99proceedings.html`.

[24] W. Wu, “Experiments with Internet Accessible Mathematical Computation”, Master’s Thesis, Department of Mathematics and Computer Science, May 1998, ICM/Kent technical report ICM-199805-0003.

[25] MathType and WebEQ, `http://www.mathtype.com/`.

[26] MathML International Conference 2000, `www.mathmlconference.org`, UIUC Illinois USA, Oct. 20-21, 2000.

[27] `http://icm.mcs.kent.edu/research/iamc/` (IAMC homepage), `http://icm.mcs.kent.edu/research/iamcproject.html` (IAMC project homepage).

[28] JavaMath, `http://javamath.sourceforge.net/`.

[29] SymbolicNet, `http://www.symbolicnet.org/systems/demos.html` live demos.

[30] Proceedings of the IAMC’99 Workshop, `http://icm.mcs.kent.edu/research/iamc99proceedings.html`, July 1999.

[31] Workshop on *The Future of Mathematical Communication* `http://www.msri.org/activities/events/9900/fmc99/`, Dec. 1999.