Enhancing Interactivity of Software and Data Repositories with Java

Ronald F. Boisvert, Bruce Miller
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA.
email: boisvert@nist.gov, bruce.miller@nist.gov

Keywords: Java, matrices, math software, online catalog, testing data, World Wide Web

ABSTRACT

Network-based computing, such as that facilitated by languages like Java, has the potential to radically change how scientific software is developed, distributed, and used. In this paper we describe two projects in which experimental Java-based tools have significantly improved the interactivity of network services related to scientific computing. The first is the Guide to Available Mathematical Software, a cross-index and virtual repository of reusable software components for computational science. The second is the Matrix Market, a repository of test data for matrix algorithms.

INTRODUCTION

The Internet is changing how scientific software is developed, distributed and used. Software and associated data are now routinely made available for downloading via the World Wide Web. This has been so successful, in fact, that the volume of this material now demands increasingly sophisticated techniques for resource location and selection. The Web is also providing new opportunities to exercise scientific software itself, either via network-accessible computational services or via executable Web content such as provided by Java [1] and related technologies.

Java is an object-oriented programming language resembling C++. A Java compiler produces byte-codes which are executed by a Java Virtual Machine (JVM). The byte-codes are platform independent, running on any computer which has a JVM implementation. Since the JVM executes the byte-codes from within a secure environment, the danger of running Java programs from unknown sources can be minimized. Further, along with the installed JVM, a computer will have a standard library of Graphical User Interface (GUI) elements and utilities. Thus, *applets*, small but complete applications, can be embedded in web pages and executed by a JVM within a user's web browser, giving rise to so-called executable content.

It is this notion of executable content that first suggests Java's usefulness as a smart front-end to online databases or computational services. But the promise of portability, security and adherence to standards, such as IEEE-754 arithmetic, offers the potential for much more. The distinction between client and server can become blurred when significant computation takes place in the user's web browser.

In this paper we describe how Java is being used to improve the interactivity of two services currently provided by NIST. In the next section we describe the first of these, the Guide to Available Mathematical Software (GAMS), a cross-index and virtual repository of reusable software components for computational science. We then describe HotGAMS, a Java-based client for GAMS which improves its user interface and adds new functionality. Next, we outline the second service, the Matrix Market, a repository of test data for matrix algorithms. This is followed by a description of the Matrix Market Deli, a collection of Java applets which enhance the interactivity of the service by generating user-specified test matrices on demand inside a Web browser. Finally, we make observations on the state of software development using Java.

THE GUIDE TO AVAILABLE MATHEMATICAL SOFTWARE

The Guide to Available Mathematical Software (GAMS) [3] is an on-line cross-index and virtual repository of reusable mathematical software components (which we term *modules*) of use in computational science and engineering research. GAMS performs the function of an inter-repository and inter-package cross-index, collecting and maintaining data about software available from external repositories and presenting it as a homogeneous whole. It also provides the functions of a repository itself (i.e., retrieval). But, instead of maintaining the cataloged software itself, it provides transparent on-demand access to repositories managed by others.

GAMS currently contains information on nearly 10,000 problem-solving modules from 100 packages. The index includes software in use by NIST laboratory scientists (both public-domain and commercial) as well as software distributed by *netlib*, the premier archive of research-grade mathematical software developed by the numerical analysis community. Although both public-domain and commercial software are cataloged, source code of proprietary software are not available through GAMS; but items such as documentation and example programs often are.

All problem-solving software modules in GAMS are assigned one or more problem classifications from a 736-node tree-structured taxonomy of mathematical and statistical problems developed as part of the project [2]. Users can browse through modules in any given problem class. To find an appropriate class, one can utilize the taxonomy as a decision tree, or enter keywords. Keywords can be mapped either to nodes in the classification system, or to individual software modules. In addition, users can browse through all modules in a given package, or all modules with a given name. Each module's abstract lists the retrievable objects associated with the module, such as documentation, examples, test programs, source code and dependencies. (More than 32,000 such objects can be retrieved.)

At the core of the GAMS system is a relational database of information about available software. This database is maintained at NIST, which provides a classification service for the repositories it indexes. The GAMS network server provides this information to network clients using a specialized protocol over TCP/IP connections. Most users access GAMS via a World Wide Web gateway at `http://math.nist.gov/gams` which maps URLs to GAMS server requests in real time via http's Common Gateway Interface (CGI).

HOTGAMS

While hierarchical, problem-oriented, classification of software is natural and powerful, it can lead to difficulties for the user in isolating the best module for a given application. HotGAMS (`http://math.nist.gov/HotGAMS/`) attempts to remedy these shortcomings by using the capabilities of Java to provide a more flexible and capable front-end to the GAMS server.

The set of available mathematical software is very dynamic, with new algorithms and implementations constantly appearing, and new computational problems becoming feasible. It is simply not possible to foresee all possible problem classes and define the taxonomy once and for all. It is inevitable, then, that problem classes representing active areas of mathematics will be over-filled with related, but increasingly disparate, modules. GAMS already has 22 problem classes with at least 200 modules. One remedy is to spawn new problem subclasses. This is not always reasonable, however, since not all meaningful distinctions between modules indicate distinct problem classes. Not subdividing leaves a *resolution* problem for the user, while subdividing can create *navigation* problems.

A ragged hierarchy that is shallow in some places and arbitrarily wide or deep in other places can be confusing to navigate as a decision tree. It can be particularly awkward when browsing through a maze of Web pages acquired from a possibly sluggish network. This problem is overcome in HotGAMS by providing a completely different user interface not solely based on hypertext. We provide a 'tree widget' for exploring the classification scheme, much like the file system browser of many operating systems. Each problem class can be opened or closed, in place, to reveal or conceal its subclasses, giving access to the modules and other information associated with each class. This approach is responsive and it is easy for the user to backtrack. HotGAMS obtains the necessary information directly from the native GAMS server using TCP/IP socket connections.

The resolution difficulty presents a more challenging problem. Once users have found the most appropriate problem class, they may still be faced with an overwhelming number of modules. Hopefully a few of these modules will solve the problem at hand; many others in the list solve slightly different problems, or are inapplicable because of preconditions or restrictions, or perhaps they are more general than necessary. We would like to assist users without forcing them to examine each module in turn.

To resolve modules in a class we employ a hybrid faceted/hierarchical scheme; simply put, we add various descriptive attributes to each module, characteristics which distinguish it from other modules in the same class. Ideally, user specification of desired attribute values would reduce the set of modules to a manageable number of equivalent modules. Given these distinguishing characteristics, we could present the user with a series of questions, but it is not clear that any given ordering of the questions would make sense to all users; a misunderstood question early in the query would lead to a fruitless search. A huge HyperText Markup Language (HTML) form could be presented to the user, but this would involve a slow back and forth interaction with the server; the user would get little feedback about the process.

A better solution, we feel, involves bringing the list of modules, along with their attributes, into an applet within the browser. A GUI for selecting the desired attributes is provided and a dynamically updated list of currently selected modules are simultaneously visible. Choosing any module characteristic causes the list of modules to be contracted (or expanded) immediately, providing feedback on the process. Users can focus on the attributes most relevant, or understandable, immediately seeing whether a given combination of features is unavailable. Users need only answer questions (i.e. make attribute selections) until the set of modules has been resolved to a manageable number. The user interface presented during the refinement process is illustrated in Figure 1.

Currently, only a limited attribute set is present in the database — primarily the programming language used, the numerical precision and whether the source is publicly available. However, we have implemented the resolution mechanism in HotGAMS in a generic fashion, lacking only the availability of more distinguishing characteristics provided by the database. The next phase of the project will be to add such data into the GAMS database.

THE MATRIX MARKET

In order to make reliable, reproducible quantitative assessments of the value of new algorithmic developments in numerical linear algebra it is useful to have a common collection of representative problems through which methods can be compared. For sparse matrices the Harwell-Boeing Sparse Matrix Collection [5] has served this purpose for some time. One of the difficulties with such collections is that their size and diversity makes them unwieldy to manage and use effec-

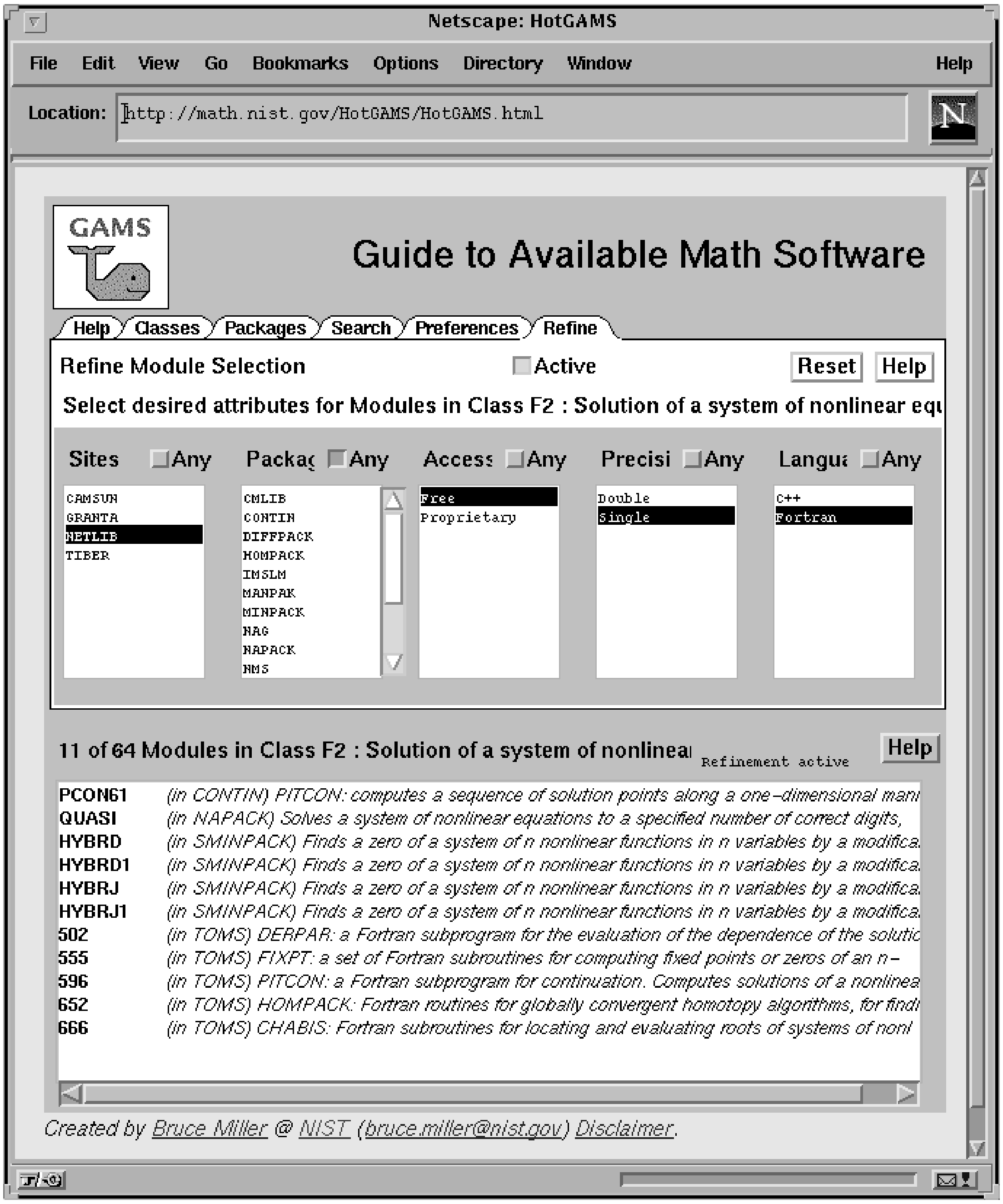

Figure 1: The HotGAMS User Interface: a sophisticated Java applet

tively. Recent developments in communications infrastructure, such as the World Wide Web, are opening up new possibilities for improving the access to and usability of such test corpora.

The Matrix Market [4] (`http://math.nist.gov/MatrixMarket/`) is such a Web-based database of matrix test data. Each matrix is documented by a Web page in HTML format outlining its properties and displaying graphical representations of its structure, such as density maps and spectral portraits. The matrices are of a wide variety of types, e.g., real, complex, symmetric, nonsymmetric, Hermitian. Some are representations of nonzero patterns only. Others include supplementary data such as right-hand sides and solution vectors. Matrices are distributed as compressed ASCII files in both the well-known Harwell-Boeing format, as well as a new Matrix Market format, which includes an exchange format for dense matrices. Alternatively, matrices may be available implicitly via a code which generates them. Use of such codes allow one to easily generate a large set of test matrices depending on one or more parameters.

Matrices are gathered together into *sets* related by application area or contributed from a single source. Each set also has its own Web page which gives its background (e.g., source and application area), references, as well as a thumbnail sketch of each matrix's nonzero pattern. Similarly, each matrix generator code also has a Web page. Sets and generators are grouped further into *collections* managed by a single group, such as the Harwell-Boeing collection. At this writing, the Matrix Market contains 482 individual matrices and 24 matrix generators comprising four separate collections.

We maintain a database containing all of the information on these pages in a highly structured form. This allows us to manipulate the data in various ways; for example, all of the matrix and set Web pages are automatically generated from this database. The database also supports both structured and free-text retrieval. Such tools which allow users to request matrices satisfying very specific criteria, such as arithmetic field, symmetry, size, and density, are a key to the usefulness of this service. Related tools allow the identification of matrices and generators by application field or contributing institution.

THE MATRIX MARKET DELI

The Matrix Market Deli (`http://math.nist.gov/cgi-bin/mmdeli/`) is a collection of Java applets which are used to construct matrices useful for testing linear algebra software according to user specified parameters. As applets, these generators perform all computations within the user's browser. They have a graphical user interface for specifying parameters, allowing for a more informative parameter validity check. Related objects can also be computed by the applet. For example, some test matrices have inverses or eigenvalues of known explicit form; these can be computed exactly by the applet for use in testing inversion or eigensystem routines.

The applets also provide means for inspecting the resulting matrices: as tabular data, as density or magnitude maps or computed indicators such as measures of density, bandedness, diagonal dominance and so forth. Thus, to the extent the applet is capable of computing the relevant qualities, the user may explore different parameter values to design a matrix satisfying specific criteria. The generated matrices can be saved to the local system through the browser.

At the present time, 15 applets are available for use, including classic examples due to Clement, Dorr, Forsythe, Frank, Gear, Kahan, Lauchli, Lotkin and Wilkinson. The specific implementations have been inspired by Nicholas Higham's Test Matrix Toolbox for Matlab (see `ftp://ftp.ma.man.ac.uk/pub/narep/narep276.ps.gz`).

The approach we have taken is to define an abstract (i.e. incomplete) applet class that presents the user interface and implements a common display and analysis environment. Each generator applet extends this abstract class. In this way, individual generators need only implement methods to define relevant parameters and to compute the matrix from those parameters. Thus, only a minimal amount of code is necessary to define new generators.

OBSERVATIONS

Java provides exciting opportunities for delivering software and data, in quite new ways. The JVM design, with wide availability of implementations, promises portability and reproducibility across all platforms. Also, the fact that applets are automatically downloaded on each use, provides a new mechanism for developers to provide instant updates to their users.

Yet, as fast as the technology is developing, it is not quite mature. It promises greater interactivity, allowing users to have more control over what they are getting and more feedback about the process. However, the range of qualities of implementations as well as the currently restrictive security model sometimes require very awkward procedures to achieve simple results. For example, at the time of this writing, saving a matrix from Matrix Market Deli generators requires us to write an HTML page — a technique that only works in one brand of browser — and then have the user save that page via the browser's menu. One must balance writing well-designed, maintainable Java code against having too many Java class files; the multiple connections required can delay the startup of the applet and many users will give up before they have even seen it.

Having the applets provided by the server but executed by the client browser greatly simplifies updating the code: one simply installs the new version on the server; clients will get the new version automatically. However, there is still the problem of keeping track of which version of Java and its Application Programmer's Interface (API) each client can support — if the client supports Java at all! By the same token, having most computation take place in the client reduces the load on the server. Yet, given the range of capabilities of client machines, from low-end personal computers to high-end workstations, one must not make too many assumptions about what a client can successfully compute. A further complication is that to maintain security and portability, a developer must use pure Java code for the client applets; we cannot simply link in `LAPACK`, for example. Thus, in Matrix Market Deli, while we might like to carry out more extensive matrix generation and analysis requiring sophisticated numerics, we must face the lack of existing linear algebra software in Java and the uncertainty about whether a given client could actually carry out the computations even if we did have the software.

Many of these problems will be resolved as richer APIs are developed — they may even be resolved by the time you are reading this — and, as the security model is validated, many restrictions will be relaxed. Other problems have less obvious solutions. While one can clearly deliver interesting and useful services using Java at present, if a broad audience is to be reached, those services will probably have to be duplicated in more traditional form for a while longer.

ACKNOWLEDGMENTS

This work is supported in part by Defense Advanced Research Projects Agency under contract DAAH04-95-1-0595, administered by the U.S. Army Research Office. Roldan Pozo and Karin Remington are co-developers of the Matrix Market. Java is a registered trademark of Sun Microsystems, Inc. This paper is a contribution of the National Institute of Standards and Technology and is not subject to copyright.

REFERENCES

[1] B. Joy and G. Steele, *The Java Language Specification*, ACM Press, New York, 1996. See also `http://java.sun.com/`.

[2] R. F. Boisvert, S. E. Howe and D. K. Kahaner, *The Guide to Available Mathematical Software Problem Classification System*, Comm. Stat. 20, p. 811 (1991).

[3] R. F. Boisvert, *The Architecture of an Intelligent Virtual Mathematical Software Repository System*, Math and Comp. in Simul. 36, p. 269 (1994).

[4] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett and J. J. Dongarra, The Matrix Market: a Web Resource for Test Matrix Collections, in *The Quality of Numerical Software: Assessment and Enhancement*, (R. Boisvert, ed.), Chapman & Hall, London, p. 125 (1997).

[5] I. S. Duff, R. G. Grimes and J. G. Lewis, *Sparse Matrix Test Problems*, ACM Trans. Math. Softw. 15, p. 1 (1989).

This paper appeared in *15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Volume 4: Artificial Intelligence and Computer Science*, A. Sydow, ed., Wissenshaft & Technik Verlag, Berlin, August 1997, pp. 767-772.