

Integrating Framework for Application Programs with Network Services Using a Web Browser

Yutaka UENO, Kiyoshi ASAI, Masanori ARITA

A simple and general framework for integrating application programs with control through a local Web browser is described. Based on our simple inter-process message function, application programs are driven from an external process by means of command messages, which can also be prepared in a script file. A message dispatcher program sends these messages to the target application as a helper-application to a Web Browser by clicking a hyper-link in the associated Web document. Our method also serves a pluggable extension-module for an application program by dynamic linking. A prototype system was implemented on our molecular structure viewer program, MOSBY, which successfully featured an extension-module for the docking study of molecular fragments from a web site. Our method is also applicable for wide range of network computations for private data processing using a Web browser.

§ 1 Introduction

Large amount of recent genomic data have motivated bioinformatics projects not only for genetic sequence analysis but also the structure based function analysis of protein molecule coded in the sequence. The network services for these databases have been integrated on the World Wide Web, on its confirmed mechanism of Hypertext Markup Language (HTML) documents¹⁵⁾. Online search and related computations are usually provided by Common Gateway Interface (CGI) in a client-server model⁸⁾.

The dependence on the network service, however, makes it difficult to process private data on stand-alone personal computers. When a researcher in pharmacological industry tries to analyze a confidential sequence or atomic coordinates, a transaction through the Internet is unfavorable. Also, the security issues on the network make the configuration of a software much complicated in the client-server system. Since advanced hardware technologies have reinforced the performance

of personal computers, one solution for processing private data is the computation on the client side.

Java programming language addressed a network-based software framework with secure code execution in a client side applet. The applet run-time model, which has been deployed in most Web browsers, allows foreign code execution with restricted accesses to local resources. Although, this method raises a problem in saving a computed result on a local disk which is usually prohibited⁹⁾. In addition to practical disadvantages in speed and portability, its programming technique becomes too complex for users to adapt. In fact, a number of existing tools which were developed in previous programming environments are widely used in bioinformatics and need to be maintained. For these reasons, a simpler way to integrate native programs is expected.

We have enhanced a molecular structure viewer program, MOSBY¹³⁾ in respect of private data processing and software integration. This paper describes our simple inter-process message functions

added in MOSBY to accept commands from an external process, thereby facilitating advanced software integration. It runs on different platforms, and allows users to add new functions. The software architecture is maintained in our programming library named ASHLEY (Application Support Hybrid Library for Easy programming), which features pluggable extension modules¹²⁾, and a high-throughput graphics library¹³⁾ as reported previously. Introduced message functions are actually implemented in this library without loss of portability.

The software scripting tools automates computational tasks and integrates application software. As for scripting tools, Perl and Tcl already play an important role in customizing in-house software systems in bioinformatics. We have introduced our message function as an interface for an application program to receive commands from another scripting tools. It enables much fine-grind control or short-cut keys for the target application. This paper focuses on our method using HTML documents and Web browser as a scripting interface.

There had been other attempts to let processes to communicate through the network. EyeChem⁴⁾ is a molecular structure viewer with networking function using a Web browser. RasMol¹⁰⁾, are widely used as a helper application for viewing molecular structures among on Web pages with its scripting commands. In these cases, however, a Web page can not send a command to the currently running application program. Our novel idea is to prepare a 'steering' button for each application program. It is a kind of graphical user interfaces to integrate local application programs to a Web browser.

§ 2 Method

2.1 Overview

Our method is based on a simple message communication between application programs, passing commands in place of the keyboard input. We defined an application message protocol and its script file format.

A message dispatcher program CAY, which stands for Command to ASHLEY, reads messages in a script file in order to dispatch and send messages to application programs. The message script takes a form of a function with three arguments:

```
caymessage("name","command","parameter")
```

This script statement sends a message of "command" to the application "name" with data "parameter" as if it were typed from a keyboard. Usually a Web browser invokes a helper application, when it processes the data of a specific MIME (Multipurpose Internet Mail Extensions)¹⁴⁾ type in a Web page. Our protocol is also in MIME format so that a HTML document on a Web page can start a CAY script to control custom, or in-house, application programs (**Fig. 1**).

2.2 Inter-Process Message

The style of message-passing follows "a single listener and multiple senders" model. Each listener are registered with unique ascii name in the local system. Messages are queued in the system, and a sender receives a reply after its listener processes its message. The programming library ASHLEY provides its Application Program Interface (API) in C language. An application program becomes a message listener by an API YoyOnMessage() with its name and a handler function which is called when the application receives a message. The message handler takes its command and parameter as arguments in character strings.

An application can send a message to a specified listener by an API YiyMessage() with two character strings that will be passed to the handler function of the listener. Sending message blocks the execution of the calling process until the listener actually completes the handler function. The size of strings data in this message is limited, but another APIs support larger storage data.

2.3 ASHLEY programming library

Event driven programming style in ASHLEY library, which was designed supports graphics user interface,

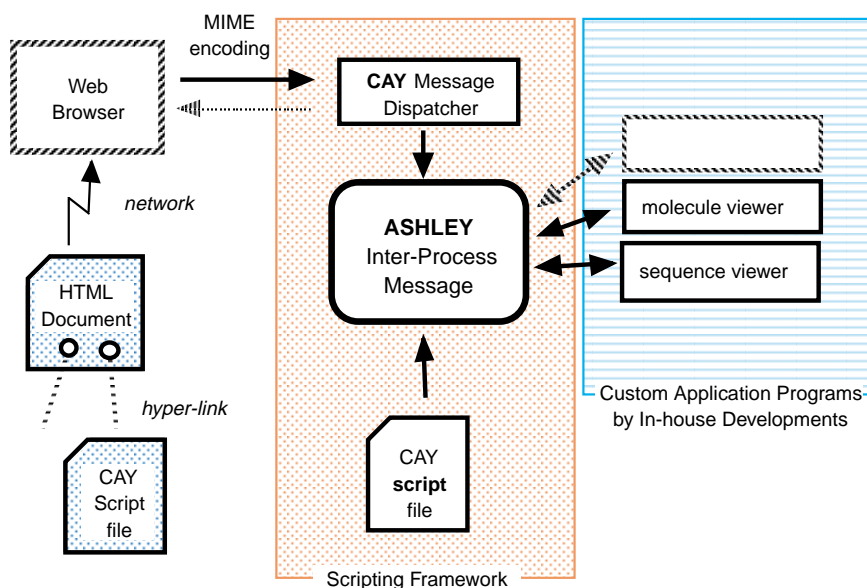


Fig.1 Overview of our integration method using a web browser

also facilitates implementing scripting function to the application. For example, selecting a menu-item generates corresponding command-strings and invokes a call-back procedure. The message handler function is also a kind of call-back procedure responding to a message with named command-strings.

The pluggable extension module is a dynamically linked code which append entries of the call-back procedure table when it is loaded. In other words, an application program can equip new commands by loading those modules.

2.4 The Message Script

There are standard commands for application programs: "open", "launch" and "quit". The "launch" command starts an application program if it is registered as a listener in your local system. If the program is already running, it just brings it to the foreground. A data file can be attached in a script file in following example:

```
caymessage("mosby","open",caystrip(".pdb"))
$endinput
.....
```

The function, `caystrip()`, creates a temporary file of the attached data after a keyword `$endinput` until the end of file. This command send a message with the file-name for a command "open" to an application "mosby". An argument to the function `caystrip()` is an extension to the temporary file so that application program can recognize it easily.

For sending binary data or an executable code, we embed the CAY script messages in PAX format. It is an archive of data indexed with ascii labels organized from the end of the file. When a message dispatcher program CAY opens a binary file in PAX format, it processes its archive record indexed as "caymagic" which contains CAY script. The next example is a program loader named "launch", which executes the code in the given file.

```
caymessage("launch","open",paxcode())
```

Function `paxcode()` creates a temporary file of the codes sent in PAX format. The PAX archive may also contain multiple binary executables, so that an appropriate code for each local operating system will be extracted. The use of the function `paxcode()` always spawns a dialog box to confirm the execution of the

code, because the foreign code potentially violate the security of the local system.

The next example sends a dynamically loadable code to the currently running application "mosby", which recognizes the command "plugin" to execute the code as a pluggable extension module.

```
caymessage("mosby","plugin".paxcode())
```

An application program has to guarantee the security of the execution of a foreign code in a local system. The function paxcode() can be implemented with some security checks for viral or malicious codes or with an appropriate castration.

§ 3 Implementation

3.1 A Message Dispatcher Program

A message dispatcher program, CAY was implemented with described scripting primitives : caymessage(), caystrip(), and paxcode(). It also overrides messages to program loader named "launch" in a script file to act as an installer for a binary program code in PAX format.

The code was developed on a workstation (Silicon Graphics Inc., Indy R4400SC, IRIX 6.4) and a personal computer (Dell Computer Corp., GXi5200, Linux 2.0.27). ASHLEY inter-process message function was implemented using the window property function in X-Window system. A message listener creates a property of the message name on the root window to wait a message by a Xlib function XSetSelectionOwner(). A sender creates an window with message context and send them using XConvertSelection() so that it can receive a XPropertyChangeEvent from the listener as a reply or error code on absence of the listener.

3.2 Cooperation With a Web Browser

Navigator 4.05 (Netscape Communications Corp.) was used for a Web browser. A message dispatcher program CAY was installed as a helper application in the MIME type application/x-cay for files with extension

".cay". After this setting, a local CAY script linked in a HTML file was processed correctly. Thus, it was extremely easy to build and maintain this system. To serve the script on network, the http server only requires this MIME type registration. When a browser handles a CAY script linked from a HTML document, the URL address should be specified to allow further improvement in security. It was provided by "%u" to the command line arguments to the helper application as well as "%s" for its cache file.

On the other hand, if an application program wants to send a message to a Web browser, the messaging function may differ according to the browser and the operating system^{11,16}. This process was delegated by another wrapper program which takes an ASHLEY message "www" and forward the message to the corresponding function of browsers. This is a portable solution to communicate with a selected browser on various operating systems. For example, command "openURL" request a Web browser to show up specified Uniform Resource Locator (URL) as its parameter.

§ 4 Result

4.1 Application to MOSBY

Our program MOSBY was modified using implemented message listener function to offer some scripting command. In addition to simple forwarding messages to internal command name, a command "plugin" is prepared to load a pluggable extension code. A demo Web page is available on <http://www.etl.go.jp/~ueno/demo99/> (**Fig. 2**). After installing our CAY dispatcher program to a Web browser, the page can demonstrate MOSBY program with interactive messages by the following links:

1. Downloading MOSBY: This link sends a binary executable of MOSBY, which will be launched automatically. This can be used for an alternative to a Java applet.
2. Browsing a PDB File: This link sends a CAY script file including a PDB file, which MOSBY will

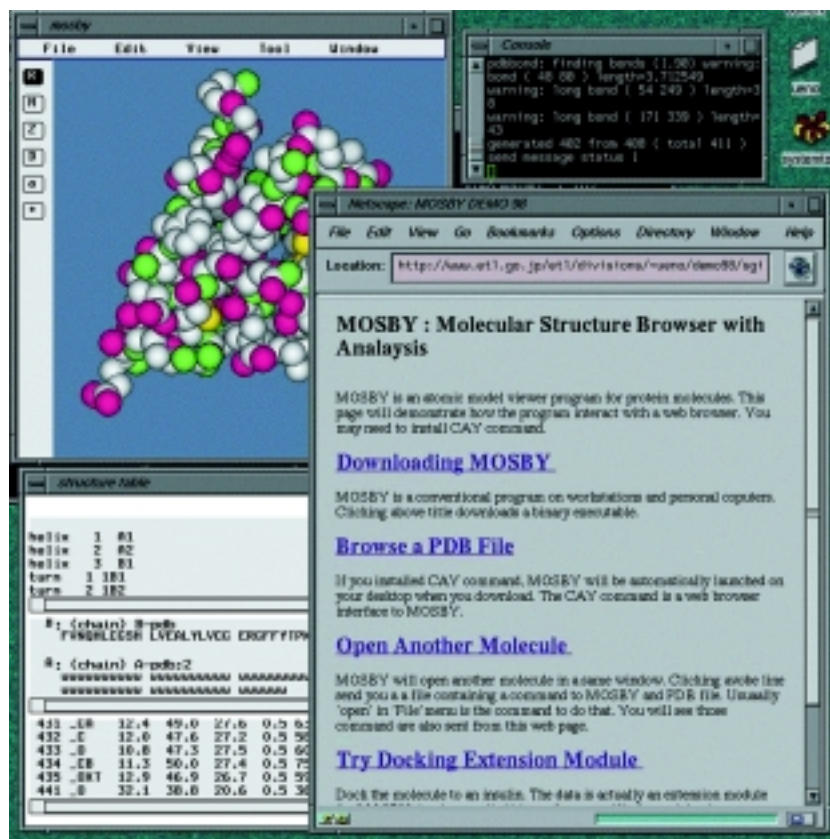


Fig.2 A snap shot of our mosby demonstration page

display.

3. Open Another Molecule: This link also sends a CAY file and appends a molecule to the current window of MOSBY.
4. Loading Docking Study Extension Module: It provides a pluggable application extension to MOSBY, which will be installed automatically. The second fragment loaded by the previous link can be rotated or moved by the switching mode of mouse operation.

Although this demo skips the calculation of collisions and of molecular force field energy for a docking study, different algorithms and energy calculations in pluggable extension modules were tested. Our work is the first to achieve the binding command sequences and code fragments for a molecular structure viewer with a Web page through a network. The distribution our code of MOSBY and ASHLEY will be released at <http://www.etl.go.jp/~ueno/bioinfo>.

§ 5 Discussions

5.1 The MIME Type and Helper Application

In a standard integration with Web browsers, a helper application like RasMol¹⁰⁾ can not interactively receive a message from a Web page. Our framework solved this problem by inter-process message passing. The support of one MIME type for each application programs⁴⁾ is another shortcoming of the standard helper application. Because increasing numbers of MIME types makes their file extensions confusing, Web browsers of end-users easily fail to synchronize. Since our CAY script uses only one MIME type, no more MIME types are required for the extended programs in a customized software system.

Multi-media contents are often handled by a plug-in module of a browser, in which the content is embedded in its window using an OBJECT tag. In this case, the information context is regulated by the Web browser,

and will disappear on closing or switching the window. In contrast, our framework let a helper application run regardless of the existence of a Web page. A Web page is merely a control panel with steering buttons, sending commands to the application. This is a more favorable scheme to integrate several applications through a browser.

5.2 Security Considerations

When a HTML document drives a local application program, two issues must be considered for security purpose: protection of a local resource and information leakage. Instead of grappling with their fundamental problem, we entrust these issues to an individual application program. For this reason, if an application program handles a command to delete files, it is subject to attack by a viral script. Fortunately, most bioinformatics computations do not need those risky command.

Limiting privileges for a general binary executable and pluggable application extension code for security reasons often sacrifices useful functions for an application program. The digital signature⁵⁾ used in Java for authentication could be employed, but may result in endless patch ups for security holes. In our framework, the code execution is application specific and they can also equip special defense module. It is easily to update, and the chance of the intrusion is much less. Unlike a class library in Java, a foreign code does not interfere among programs.

5.3 Comparisons with Related Works

Windows operating system (Microsoft Corp.) has a mechanism called ActiveX to embed a code in a HTML document to be executed in a local Web browser²⁾. Our method can also send a code fragments through a browser, but differs in a point that the execution is subject to the application program. The application program is responsible for whether taking a full advantage of computation with local resources or running a risk of executing code from the Internet.

There are also related works in terms of local code execution: WebApp⁹⁾ focused on the installation of programs, and Flypaper⁷⁾ uses AppleScript (Apple

Computer Inc.). However, they lack portability for UNIX operating systems where most programs for bioinformatics have been developed. Since ASHLEY programming library is already ported to Windows and MacOS, it is also possible to implement the described method for those platform. EyeChem⁴⁾ is also a molecular structure viewer which can communicate with a Web browser. But its software architecture depends on a specific visualization tool¹⁾ which lacks portability for practical use.

5.4 A Scriptable Application

A scriptable application is a terminology of Apple computer for a graphical user interface program, which accepts commands from a script file. In an object oriented framework like CORBA (Object Management Group)²⁾, the interface to a software component has to be pre-defined, which becomes sometimes inadequate during a software-development cycle. In addition, due to the considerable amount of knowledge required, it has not successfully been caught up by the real world bioinformatics community.

In contrast, our basic concept is to develop a framework alternative to the keyboard. This led us to more comprehensive way for a scriptable application. Our method stays in a conventional programming model, where a single thread process waits a command-input from a terminal. In terms of scripting languages, we are planning to embed LUA language⁶⁾ to our message dispatcher program for more detailed control of messages using variables, conditions and loops.

5.5 An Application to Gene Finding

Since our method is a general framework for scripting and networking application programs, we currently apply to our genomic sequence viewer. The program graphically displays annotation information to the given sequence such as protein coding regions predicted by our gene finding system³⁾. Since the gene finding system requires different levels of data set and knowledge for the target sequence, a computational server with a single or few algorithm does not always

give the best criteria for scientists. For example, viewing multiple results together with other related information in a local disks would be important. While server side computing has the limit in this respect, private computing can maximize the ability of programs to support human inference on undocumented knowledge in biology.

§ 6 Conclusion

Our simple scripting framework for application programs using inter process message implemented in a programming library ASHLEY was introduced. Through a message dispatcher program as a helper application to the MIME type, we were able to control local application programs from a Web browser by selecting links to its script data. The result showed that this method is applicable for a wide range of application systems for a private data processing together with network services.

Acknowledgments

We would thank to Dr.Katsutoshi Takahashi, Dr.Yutaka Akiyama for useful discussions, and Dr. Nobuyuki Otsu for continuous support and suggestions.

References

- 1) Abram, G. & Treinish, L. : An Extended Data-Flow Architecture for Data Analysis and Visualization, *Computer Graphics*, 29(2):17-21, 1995.
- 2) Adler, R.M. : Emerging Standards for Component Software, *IEEE Computer* 3:68-77, 1995.
- 3) Asai, K., Itou, K., and Ueno, Y. : Recognition of Human Genes by Stochastic Parsing, *Proceedings of the Pacific Symposium on Biocomputing* (Altman et al., eds.), World Scientific Press, 228-2239, 1998.
- 4) Casher, O. and Rzepa, H.S. : A Chemical Collaboratory using Explorer EyeChem and the Common Client Interface, *Computer Graphics* 29, 52-52,1995.
- 5) Gong, L. : New Security Architectural Directions for Java (Extended Abstract), *Proceedings of IEEE COMPCON*, San Jose, 97-102, 1997.
- 6) Ierusalimsky, R., de Figueiredo, L.H. and Filho, W.C.: Lua-an extensible extension language, *Software: Practice & Experience*, 26(6): 635-652, 1996.
- 7) Iversion, E. Flypaper, a shareware. <http://www.zeeland.k12.mi.us/~mcoleman/spanish/flypaper.html>
- 8) National Center for Supercomputing Applications, The Common Gateway Interface, 1995. <http://hoohoo.ncsa.uiuc.edu/webapp>
- 9) WebApp. The Institute for Academic Technology, The University of North Carolina at Chapel Hill, Durham, <http://www.iat.unc.edu/technology/software/webapp/index.html>
- 10) Sayle., R.A. and Milner-White, E.J. : RasMol: Biomolecular Graphics for All, *Trends in Biochemical Science*, 20, 374-376, 1995.
- 11) Thompson, D., NCSA Mosaic Common Client Interface, National Center for Supercomputing Applications. <http://www.ncsa.uiuc.edu/DG/Software/Mosaic/Docs/cci-spec.html>
- 12) Ueno, Y. and Asai, K. : A New Plug-in Software Architecture Applied for a Portable Molecular Structure Browser, *Proceedings of Intelligent Systems for Molecular Biology* (Gaasterland et al., eds.), AAAI Press, 329-332, 1997.
- 13) Ueno, Y., and Asai, K. : A High-Throughput Graphics Library Designed for a Portable Molecular Structure viewer, *Proceedings of the Pacific Symposium on Biocomputing* (Altman et al., eds.), World Scientific Press, 201-212, 1998.
- 14) Vaudreuil, G. CNRI : MIME: Multi-Media, Multi-Lingual Extensions for RFC 822 Based Electronic Mail, *ConneXions*, 36-39, (September) 1992
- 15) World Wide Web Consortium, Hypertext Markup Language (HTML), 1996. <http://www.w3.org/pub/WWW/MarkUp>
- 16) Zawinski,Z. and Netscape Communications Corp., Remote Control of Unix Netscape, Netscape Support Documentation. <http://home.netscape.com/newsref/std/x-remote.html>

(Accepted February 4, 2000)

