REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
10-26-2000	CONFERENCE PROCEEDINGS	

4. TITLE AND SUBTITLE
A CLIENT/SERVER B ASED APPLICATION USING A C/JAVA INTERFACE

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Marlin L. Gendron, Stephanie S. Edwards, Stephanie A. Myrick, Maura C. Lohrenz, and Michael E. Trenchard

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/PP/7440-00-1009

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NAVAIR

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

13. SUPPLEMENTARY NOTES

20010102 019

14. ABSTRACT

Since 1995, the Naval Research Laboratory (NRL) at the Stennis Space Center (NRLSSC) has been developing and enhancing a softwae application that allows naval AV-8B and F/A-18 aircraft mission planners and aircrew to design and build digital aeronautical chart coverages for cockpit moving-map displays. The application, known as the Moving-Map Composer (MMC), currently is implemented in the X-Windows graphical user interface (GUI) language and the C programming language.

Currently, MMC is only supported on Compaq Alpha computers running the OpenVMS operating system. This limitation requires end users to acquire Alpha stations, which are more expensive than standard Intel-based PC platforms. This paper will discuss reengineering methods being developed by NRLSSC to transform the existing MMC application to a Java and C-based program that will execute on many different hardware platforms and operating systems, including Linux, Windows NT and OpenVMS.

15. SUBJECT TERMS
graphic user interface, moving-map displays, aircraft optical disks, X-designer and X-library

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Marlin L. Gendron
Unclassified	Unclassified	Unclassified		1	19b. TELEPHONE NUMBER *(Include area code)* 228-688-4773

Proceedings of the Southern Conference on Computing
The University of Southern Mississippi, October 26-28, 2000

A CLIENT/SERVER BASED APPLICATION USING A C/JAVA INTERFACE

Marlin L. Gendron, Stephanie S. Edwards, Stephanie A. Myrick, Maura C. Lohrenz, and Michael E. Trenchard[1]

Abstract. Since 1995, the Naval Research Laboratory (NRL) at the Stennis Space Center (NRLSSC) has been developing and enhancing a software application that allows naval AV-8B and F/A-18 aircraft mission planners and aircrew to design and build digital aeronautical chart coverages for cockpit moving-map displays. The application, known as the Moving-Map Composer (MMC), currently is implemented in the X-Windows graphical user interface (GUI) language and the C programming language.

Currently, MMC is only supported on Compaq Alpha computers running the OpenVMS operating system. This limitation requires end users to acquire Alpha stations, which are more expensive than standard Intel-based PC platforms. This paper will discuss reengineering methods being developed by NRLSSC to transform the existing MMC application to a Java and C-based program that will execute on many different hardware platforms and operating systems, including Linux, Windows NT and OpenVMS.

As part of this redesign, the new MMC will have the added capability to not only execute locally, but also reside on a centralized server. This Internet-based design will make MMC more accessible to a substantially greater number of users. To accomplish this, the new software architecture will be client/server based, with the server end implemented in Java. Existing C code will be linked into the Java server as a C library to maintain all current low-level MMC functionalities. Methods to easily extract C routines from the existing MMC software with minimal changes will be addressed, as well as portability issues such as file and path naming conventions, system specific calls, logicals, and compiler directives.

The client side of the new architecture can be implemented either as a Java GUI application, browser applet, or command line program that will run remotely on any desired workstation. The development of the client application will be discussed. A text-based query language will be developed to handle the communication between the server and client. This paper will provide a description of this query language.

1. Introduction. This paper will discuss methods used by the Naval Research Laboratory (NRL) at the Stennis Space Center (NRLSSC) to transform an existing software program known as the Moving-Map Composer (MMC) from a standalone, operating system dependent application to a web-based client/server application. Also discussed are software changes that allow the program to run on any Internet capable computer or as a standalone application on Unix, Microsoft Windows, or OpenVMS architectures. Examples of these software changes will be given with the goal of keeping modifications to a minimum and avoiding major rewrites that might introduce new software "bugs".

Currently, MMC will only work on an Alpha computer running the OpenVMS operating system. Alpha computers tend to be more expensive than standard Intel-based PCs and most users are unfamiliar with the OpenVMS operating system. While some users of MMC might prefer to use Linux, others may prefer to deal with Microsoft Windows. In general, the task of reengineering software to run on many different operating systems can be daunting. Discussed here is one solution to this dilemma.

There exist several limitations within the MMC software that prohibit porting. All low-level functionality of MMC is written in the C programming language and resides in a static C library. This library is used by a graphical user interface (GUI) implemented in the X-Windows language, which will work on computers running Unix-based and OpenVMS operating systems. X-Windows will not run on Microsoft Windows machines. The low-level C code within the library contains operating system calls and file naming conventions specific to the OpenVMS operating system. Although the X-Windows GUI language will run under a Unix-based operating system, like Linux, it will not work under the Microsoft Windows operating system.

NRLSSC's solution to the problem is to replace the existing X-Windows GUI with one written in Java to serve as the client side of a client/server architecture. The low-level C library is linked into a Java "server" with only minor changes. This allows MMC to run on many different hardware platforms and operating systems, including Linux, Windows NT and OpenVMS.

2. Background. In the early 90's, NRLSSC was tasked with building digital map images using a map system designed by a contractor. The application allowed Naval AV-8B and F/A-18 aircraft mission planners and aircrew to design and build digital aeronautical chart coverages for cockpit moving-map displays.

The program was a DOS-based application with a crude GUI. The software was extremely "buggy" and areas of coverage in the Polar Regions, e.g. areas above and below 50.0 latitude, could not be designed at all due to software problems. There were also limitations in the methods users had to define map coverages. They were given a base

[1] Naval Research Laboratory, Code 7440.1, Stennis Space Center, MS 39529

map and allowed to define areas of coverage with rectangles via stretch box implementations. This was not adequate for selecting areas along a diagonal line.

To correct these problems, NRLSSC began developing standalone programs to supplement the map design software. Eventually NRLSSC combined these standalone applications into a user-friendly graphical interface program called MMC that enabled users to define available coverage using polygons. For this innovation, NRLSSC obtained a patent [6]. Because of a great demand for the MMC program, NRLSSC began looking for ways to port MMC to other architectures. NRLSSC solved this problem by going to a client/server type architecture using Java. This paper is a result of that effort.

3. GUI Changes. The client side of the new architecture is implemented either as a Java GUI application, a browser applet, or a command line program that will run remotely on any Java capable workstation. To obtain the Java GUI, the existing X-Windows GUI was converted to Java. The methods used to accomplish this task are discussed in a companion paper entitled "The Design and Development of an Internet-based Graphical User Interface Using a Commercial Design Tool in Java" [9].

4. Low-level C Library Changes. The existing MMC low-level C code is robust and well tested. In order to minimize modifications to the software so as not to introduce new bugs, a few key changes were made. One main modification dealt with file naming conventions. Fortunately, the existing C code did not have pathnames "hard-coded", but rather logical names were used. These logicals were defined in an ASCII configuration file. This was done initially so different directories and devices could be substituted in without having to recompile the code. Table 4.1 shows the current logical definitions listed in the configuration file for OpenVMS and the C system calls to define them. Sections for Windows and Unix were added and are shown in Table 4.2 and Table 4.3, respectively.

CONFIGURATION FILE (OpenVMS Section)

VMS	
MPS_HD_FILES	DKA100:[mmc.mps.]
AOD_HD_FILES	DKA100:[mmc.aods.]
(Etc)	(Etc)
System ("DEFINE/NOLOG/JOB/TRANS=CONCEAL MPS_HD_FILES DKA100:[mmc.mps.]");	

Table 4.1

CONFIGURATION FILE (Windows NT Section)

WINNT	
MPS_HD_FILES	J:\mps\
AOD_HD_FILES	J:\aods\
(Etc)	(Etc)
putenv *(MPS*_HD_FILES=J:\mps);	

Table 4.2

CONFIGURATION FILE (UNIX Section)

UNIX	
MPS_HD_FILES	/mmc/mps/
AOD_HD_FILES	/mmc/mps/
(Etc)	(Etc)
setenv (MPS_HD_FILES, /mmc/mps, TRUE);	

Table 4.3

Once the logicals are defined, the low-level C code is modified and path and filenames are replaced with function calls. These functions will create names appropriately depending on the operating system being used. This avoids the use of compiler directives like #ifdef and #endif that can clutter up the code and make for poor readability. Listed below is the sequence of calls to create a pathname. Table 4.4 shows the results for each operating system.

(4.1) multidir_init ("MPS_HD_FILES");

(4.2) multidir_send ("odi");

(4.3) multidir_send ("images");

(4.4) multidir_file ("images.mps");

(4.5) strcpy (path, multidir_ret());

	OpenVMS Results	WinNT Results	UNIX Results
4.6	MPS_HD_FILES:[000000	%MPS_HD_FILES%\	$MPS_HD_FILES/
4.7	MPS_HD_FILES:[000000.odi	%MPS_HD_FILES%\odi\	$MPS_HD_FILES/odi/
4.8	MPS_HD_FILES:[000000.odi.images	%MPS_HD_FILES%\odi\images\	$MPS_HD_FILES/odi/images/
4.9	MPS_HD_FILES:[000000.odi.images]img.mps	%MPS_HD_FILES%\odi\images\img.mps	$MPS_HD_FILES/images/img.mps

Table 4.4

Another area in which the software had to be modified was where "find file" operations occurred. Table 4.5 shows the original function calls to find files. Table 4.5 show the system calls for each operating system that are embedded inside the functions for each operating system.

```
find_context = findfile_init (path);
while ((filename=findfile_ret (find_context, FALSE) != NULL)
{
        /* Do something here */
        free (filename);
}
findfile_end (find_context);
```

Table 4.5

WINNT	UNIX	OPENVMS
_findfirst	opendir	lib$find_file
_findnext	readdir	lib$find_file_end

Table 4.6

5. Java Server/C Interface. The functionality of the new MMC will rely heavily on the ability to use both Java code for the client side, and the low-level C code on the server side. This interface between the Java code and the C code is implemented through the Java Native Interface (JNI). The Java Development Kit (JDK) requires a three-step process to produce an interface between Java and any other native code.

1. Generate a C stub for a function that translates between the Java call and the actual C function. The stub does this translation by taking information off the Java stack and passing it to the compiled C function.

2. Create a special shared library and export the stub from it.
3. Use a special Java method, called System.LoadLibrary to tell the Java run time to load the library from step 2.

In order to simplify the interface between languages, any necessary interaction has been funneled through a single Java class and a single C function. Table 5.1 shows the C function and Table 5.2 shows the Java class definition. This class receives a query from the Java application, and passes that query, along with other pertinent information, to the C function, which then calls any other necessary C functions. By filtering all client-server communication through this simple interface, the "Three Tier Architecture" concept is preserved.

```
JNIEXPORT jint JNICALL Java_ThreadedEchoServer_runMMC
(JNIEnv *env, jclass c1, jstring command, jint count)
{
   void map_composer_main(char , int);          /* The name of the C function to be called */

   const char *str = (*env)->GetStringUTFChars(env, command, 0);

   map_composer_main (str, count);
   (*env)->ReleaseStringUTFChars(env, command, str);
   return (1);
}
```

Table 5.1

```
public class ThreadedEchoServer
{
        public static int counter=0;

        public native static int runMMC(String command, int i);
        static
        {
                System.loadLibrary("ntsrc");     /*  Loads the dll produced from the C code  */
        }
        .
        .
        .
}
```

Table 5.2

6. Query Language. A mechanism is needed by which the client can request tasks for the server to perform and also receive messages from the server. By using sockets, which allow information to be passed in and out of a Java application, the MMC client can communicate with its server. A set of commands, or language, understood both by the client and the server provides one method to link them together. An ASCII-based language is currently in development capable of forming task-oriented queries. The language listed below is not complete at this time and needs to be extended to include commands sent from the server to the client, including error and informational messages, GUI control changes like button changes and text. Attempts are currently being made to keep the language terminology as consistent, reasonable, complete and sound [2].

Table 6.1 lists examples of commands sent by the client to the server and the resulting action performed by the server. Table 6.2 describes the language thus far in Backus Naur Form (BNF).

CLIENT REQUEST	SERVER ACTION
execute connect	Connects client.
execute modify scale[value=500000]	Changes map scale to 1:500K.
execute modify projection[value=NP]	Changes projection to northern polar.
build picture[type=gif,size=768,1024] wvs bounds	Builds GIF file with world vector shoreline and boundaries.
build file[type=ascii] template[location=hd,type=final,name=all]	Creates a file containing template names on the hard drive.
execute modify zoom[value=5,center=0.0,0.0]	Changes the zoom factor to 5 and the center lat/long.
build picture template[location=hd,type=final,name=spain]	Builds GIF file containing the "spain" template.
execute logout	Disconnects client.

Table 6.1

QUERY LANGUAGE DESCRIPTION

syntax ::= { query }
query ::= <execute_statement> | <build_statement>
execute_statement ::= "execute" <space> <execute_modifiers>
execute_modifiers ::= "connect" | "modify" <space> <modify_what> | "logout"
modify_what ::= <scale_expression> | <projection_expression> | <zoom_expression>
zoom_expression ::= "zoom[" <zoom_attributes> "]"
zoom_attributes ::= "value=" <zoom_value> { ",center=" <latitude> "," <longitude> }
zoom_value ::= 0 | <integer> | -<integer>
scale_expression ::= "scale["<scale_attributes>"]"
scale_attributes ::= "value="<scales>
projection_expression ::= "projection ["<projection_attributes>"]"
projection_attributes ::= "value=" <projections>
build_statement ::= "build" <space> <build_modifiers>
build_modifiers ::= <picture_statement> | <file_statement>
file_statement ::= "file" { "[" <filetype> "]" } <space> {<contents>}
picture_statement ::= "picture" { "["<filetype>{","<size>}"]" } <space> {<contents>}
contents ::= { "wvs" } {<space>"bounds"}{<space>"template_statement"}
template_statement ::= "template["<template_attributes>"]"
template_attributes ::= "["<location_statement> {","<template_type_statement>}{","<name_statement>}{","<id_statement>}"]"
location_statement ::= "location="<location_type>
template_type_statement :: = "type="<template_type>
name_statement ::= "name="<name_type>
id_statement ::= "id=<id_type>
filetype ::= "type=" <filetype_modifier>
filetype_modifier ::= <graphics_format> | "ascii" | "binary"
size ::= "size="<integer>","<integer>
scales ::= "50000" | "100000" | "250000" | "500000" | "1000000" | "2000000"
projections ::= "SP" | "Mercator" | "NP"
latitude ::= -90.0 – 90.0
longitude ::= -180.0 – 180.0
location_type ::= "hd" | "cdrom"
template_type ::= "working" | "final"
graphics_format ::= "gif" | "tiff" | "jpeg"
name_type ::= any valid id string | "all"
id_type ::= any valid id | "all"
integer ::= any positive integer
space ::= " "

Table 6.2

7. Summary. Several details of porting an existing software package written in C and X-Windows to many different operating systems and ultimately the Internet are discussed. Specific examples including path, filenaming and system calls for finding files are given. Code examples are listed with the goal in mind of keeping modifications to a minimum and avoiding major rewrites of the software, which might introduce new software "bugs". Finally, the client/server architecture is also discussed, and a query language developed to allow the client and server to communicate.

Although this process was for the most part unproblematic for NRLSSC, much success can be attributed to sound software engineering techniques used in the original MMC program design. The initial choice to separate the GUI from the low-level C code, and the use of logical names, instead of "hard-coding" pathnames, aided in the smooth transition to a multi-platform Internet-based application.

8. Acknowledgements. This work is funded by the NAVAIR FA-18 program (program elements 204136N). The authors thank the sponsoring program manager, Sharam Bavani, at NAVAIR for his support of this project.

PUBLICATION OR PRESENTATION RELEASE REQUEST

SSC-138-00

NRLINST 5600.2

1. REFERENCES AND ENCLOSURES	2. TYPE OF PUBLICATION OR PRESENTATION	3. ADMINISTRATIVE INFORMATION
Ref: (a) NRL Instruction 5600.2 (b) NRL Instruction 5510.40D Encl: (1) Two copies of subject paper (or abstract)	( ) Abstract only, published ( ) Book ( ) Conference Proceedings (refereed) ( ) Invited speaker ( ) Journal article (refereed) ( ) Oral Presentation, published ( ) Other, explain ( ) Abstract only, not published ( ) Book Chapter (X) Conference Proceedings (not refereed) ( ) Multimedia report ( ) Journal article (not refereed) ( ) Oral Presentation, not published	STRN NRLPP/7440-00-1009 Route Sheet No. _____ Job Order No. _____ Classification X U _____ C Sponsor NAVAIR PA18 approval obtained _____ yes _____ no

4. AUTHOR

Title of Paper or Presentation

A CLIENT/SERVER BASED APPLICATION USING A C/JAVA INTERFACE

Author(s) Name(s) *(First, Mi, Last)*, Code, Affiliation if not NRL

MARLIN L. GENDRON (Code 7440.1), STEPHANIE S. EDWARDS (Code 7440.1), STEPHANIE A. MYRICK (Code 7440.1), MAURA C. LOHRENZ (Code 7440.1), and MICHAEL E. TRENCHARD (Code 7440.1)

It is intended to offer this paper to the SOUTHERN CONFERENCE ON COMPUTING
(Name of Conference)

THE UNIVERSITY OF SOUTHERN MISSISSIPPI, OCTOBER 26-28, 2000, HATTIESBURG, MISSISSIPPI
(Date, Place and Classification of Conference)

and/or for publication in _____ _____
(Name and Classification of Publication) *(Name of Publisher)*

After presentation or publication, pertinent publication/presentation data will be entered in the publications data base, in accordance with reference (a).

It is the opinion of the author that the subject paper (is____) (is not X) classified, in accordance with reference (b).

This paper does not violate any disclosure of trade secrets or suggestions of outside individuals or concerns which have been communicated to the Laboratory in confidence. This paper (does____) (does not X) contain any militarily critical technology.

This subject paper (has____) (has never X) been incorporated in an official NRL Report.

MARLIN L. GENDRON, NRL Code 7440.1 Marlin Lee Gendron
Name and Code *(Principal Author)* (Signature)

5. ROUTING/APPROVAL

CODE	SIGNATURE	DATE	COMMENTS
Author(s) Gendron	Marlin Lee Gendron	26 Oct 2000	
Section Head Lohrenz			
Branch Head Harris	W. Eric Reeves	26 Oct 00	
Division Head Acting 7400 Valent Eppert	Philip J. Valent	10/26/00	1. Release of this paper is approved. 2. To the best knowledge of this Division, the subject matter of this paper ~~this~~ _____ (has never X) been classified.
Security, Code ~~1221~~ 7031	David K Cardinez	10/31/00	1. Paper or abstract was released. 2. A copy is filed in this office. SSC-138-00
Office of Counsel, Code ~~1~~ 1008.2		11/17/00	
ADOR/Director NCST			
Public Affairs *(Unclassified/ Unlimited Only)*, Code ~~1230~~ 7030.4	Paula Patanella	11/1/00	
Division, Code			
Author, Code 7440.1			

HQ-NRL 5511/6 (Rev. 12-98) (e) THIS FORM CANCELS AND SUPERSEDES ALL PREVIOUS VERSIONS

6. DISTRIBUTION STATEMENTS *(Author to check appropriate statement and fill in reason as required)*

☒ A - Approved for public release, distribution is unlimited.

☐ B - Distribution authorized to U.S. Government agencies only *(check reason below):*

☐ Foreign Government Information ☐ Contractor Performance Evaluation ☐ Critical Technology
☐ Proprietary Information ☐ Administrative/Operational Use ☐ Premature Dissemination
☐ Test and Evaluation ☐ Software Documentation ☐ Cite "Specific Authority _____"
(Identification of valid

Date statement applied _____

Other requests for this document shall be referred to _____
(Insert Controlling DOD

☐ C - Distribution authorized to U.S. Government agencies and their contractors *(check reason below):*

☐ Foreign Government Information ☐ Software Documentation
☐ Administrative/Operational Use ☐ Critical Technology ☐ Cite "Specific Authority _____"
(Identification of valid

Date statement applied _____

Other requests for this document shall be referred to _____
(Insert Controlling DOD

☐ D - Distribution authorized to DOD and DOD contractors only *(check reason below):*

☐ Foreign Government Information ☐ Critical Technology
☐ Software Documentation ☐ Cite "Specific Authority _____"
(Identification of valid
☐ Administrative/Operational Use

Date statement applied _____

Other requests for this document shall be referred to _____
(Insert Controlling DOD

☐ E - Distribution authorized to DOD components only *(check reason below):*

☐ Critical Technology
☐ Proprietary Information ☐ Premature Dissemination ☐ Direct Military Support
☐ Foreign Government Information ☐ Software Documentation ☐ Test and Evaluation
☐ Administrative/Operational Use ☐ Contractor Performance Evaluation ☐ Cite "Specific Authority _____"
(Identification of valid

Date statement applied _____

Other requests for this document shall be referred to _____
(Insert Controlling DOD

☐ F - Further dissemination only as directed by _____
(Insert Controlling DOD

Date statement applied _____ or higher DOD authority _____

☐ X - Distribution authorized to U.S. Government agencies and private individuals or enterprises eligible to obtain export-controlled technical data in accordance with regulations implementing 10 U.S.C. 140c.

Date statement applied _____

Other requests for this document shall be referred to _____
(Insert Controlling DOD

*For NRL publications, this is usually the Commanding Officer, Naval Research Laboratory, Washington, DC 20375-5320

7. OTHER LIMITATION

31 Oct 00
Classification Review
(Initial/Date)

☐ Classification only ☐ NOFORN ☐ DTIC exempt (explain) _____

Substantive changes made in this document after approval by Classification Review and Public Release invalidate these reviews. Therefore, if any substantive changes are made by the author, Technical Information, or anyone else, the document must be returned for another Classification Review and Public Release.

8. INSTRUCTIONS

Author completes and submits this form with the manuscript via line channels to the division head for review and approval according to the routing in section 4.

1. NRL Reports.....Submit the diskette (if available), manuscript, typed double-spaced, complete with tables, illustrations, references, draft SF 298, and proposed distribution list.

2. NRL Memorandum Reports.....Submit a copy of the original, typed manuscript complete with tables, illustrations, references, draft SF 298, and proposed distribution list.

3. NRL Publications or other books, brochures, pamphlets, proceedings, or any other printed publications.....Handled on a per case basis by Site Technical Information Office.