Technical Report ITL-96-3
May 1996

US Army Corps of Engineers
Waterways Experiment Station

Investigation of Issues for Conversion of the Information System Modernization Program (ISMP) Automated Information Systems (AIS) to Use a Graphical User Interface

by Michael E. George, Julia A. Baca

Prepared for Headquarters, U.S. Army Corps of Engineers

PRINTED ON RECYCLED PAPER

Technical Report ITL-96-3
May 1996

Investigation of Issues for Conversion of the Information System Modernization Program (ISMP) Automated Information Systems (AIS) to Use a Graphical User Interface

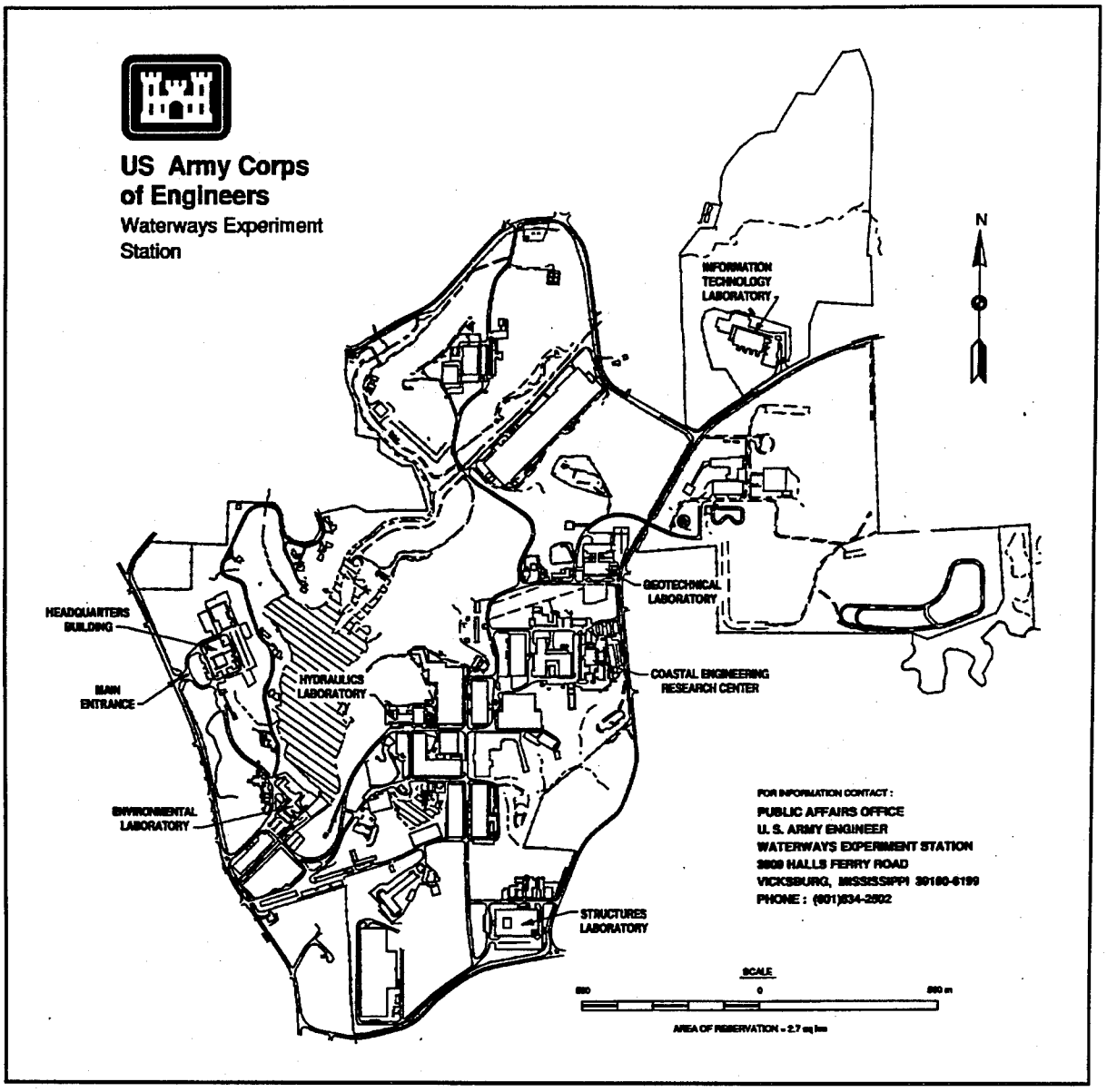
by Michael E. George, Julia A. Baca

U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000



Waterways Experiment Station Cataloging-in-Publication Data

George, Michael E.

Investigation of issues for conversion of the Information System Modernization Program (ISMP) Automated Information Systems (AIS) to use a graphical user interface / by Michael E. George, Julia A. Baca ; prepared for U.S. Army Corps of Engineers.

42 p. : ill. ; 28 cm. — (Technical report ; ITL-96-3)

1. Graphical user interfaces (Computer systems) 2. User interfaces (Computer systems) 3. Client/server computing. 4. Computer system conversion — United States — Army —Corps of Engineers. I. Baca, Julia A. II. United States. Army. Corps of Engineers. III. U.S. Army Engineer Waterways Experiment Station. IV. Information Technology Laboratory (U.S. Army Engineer Waterways Experiment Station) V. Title. VI. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; ITL-96-3.

TA7 W34 no.ITL-96-3

TABLE OF CONTENTS

PREFACE

This report documents an investigation of issues that must be addressed if the U.S. Army Corps of Engineers (USACE) Information System Modernization Program (ISMP) Automated Information Systems (AIS) are to be converted to a graphical user interface (GUI) environment. The investigation was sponsored by the USACE Headquarters (HQUSACE). This report was prepared by Mr. Michael E. George with assistance from Ms. Julia A. Baca of the Information Technology Laboratory (ITL) of the US Army Engineer Waterways Experiment Station (WES) under the direction of Dr. N. Radhakrishnan, Director, ITL; and Dr. Windell Ingram, Chief, Computer Science Division.

The Director of WES is Dr. Robert W. Whalin. The Commander is COL Bruce K. Howard, EN.

I. EXECUTIVE SUMMARY

A. Introduction

The Information System Modernization Program (ISMP) is currently comprised of four Automated Information Systems (AIS), including the Corps of Engineers Financial Management System (CEFMS), the Real Estate Management Information System (REMIS), the Resident Management System (RMS), and the Programs and Project Management Information System (PROMIS). All of these systems were designed and developed to meet the needs of specific user communities. The first three systems, CEFMS, REMIS, and RMS, employed the standard user interface technology available at the time of development, character-based displays. However, rapidly occurring changes in desktop computing technology have since made the use of a graphical user interface (GUI) possible for these systems. Indeed, the user community has begun to expect this type of interface which has now become the defacto standard for software purchased off-the-shelf. A standard GUI for ISMP systems would offer advantages, allowing the user, for example, to simply click on an icon to move from one application to another or to perform numerous other functions within the interface. Nonetheless, despite the ease-of-use offered by a GUI, the conversion from a character-based user interface to a GUI environment requires significant effort and thus, should be undertaken only after careful consideration of the costs and benefits. This report presents the results of investigating three alternative approaches to a GUI implementation, based on current technology. These include 1) an ORACLE SQL*Forms approach with emulation of an X-Windows environment, 2) an ORACLE SQL*Forms approach in a Microsoft Windows client-server environment, and finally 3) a generic Microsoft Windows client-server environment which does not depend on a single vendor, such as ORACLE. Several criteria were used to evaluate each approach. Although the evaluations apply generally to any ISMP AIS, the investigation focused primarily on CEFMS. The remainder of this executive summary discusses the evaluation criteria and synopsizes the evaluation of each approach.

B. Evaluation Criteria

The successful conversion of existing character-based applications to a GUI environment requires a number of issues to be addressed. These issues, listed below, comprised the criteria along which the three approaches were evaluated:

1) Characteristics of the GUI. This entails how the interface looks and feels to the user. The term GUI is so loosely defined and used that many products can claim the status of a GUI. If the interface does not meet user expectations, however, it will not be accepted.

2) Performance. Will a GUI environment improve or degrade performance for the user community? Conversion to a GUI will be futile if the application becomes unacceptably slower for the users.

3) Network Traffic. How will a GUI environment impact network traffic if GUI images must be stored on remote machines and transmitted to individual desktop machines for local display?

4) Version Control. How can the current version of the application be distributed to the user community if changes to the application occur frequently? This is affected by the methodology used to implement a GUI.

5) User Exits and Triggers. Most current ISMP AIS have additional processes that are executed from within the application itself. Some of these processes are initiated by events that take place when data is entered or edited in a particular field on a form, for example, a social security number or work item code. The processes associated with verification and validation of data entry per field are called "user exits" in ORACLE terminology. Also, once data is verified, validated, and/or submitted to the database, additional processes, such as reports generation, can be "triggered" to start on the remote server. Hence, these are referred to in ORACLE terminology as "triggers". Triggers can be short, as described here, or can be lengthy. How can these processes be included properly in a GUI environment? Can these processes be converted cost effectively to a GUI environment?

6) System Configuration. What costs will be incurred by additional software and/or hardware needed to augment existing PC configurations to allow the converted AIS to execute properly? Specifically, if conversion to a GUI is feasible, what costs are associated with Corps-wide deployment of the application?

7) Other. Some ISMP AIS, namely CEFMS, contain specialized software and hardware to accomplish electronic signature. How will such features be properly implemented in a GUI environment?

One final, important issue, mouse movement verification, must be addressed in the conversion from a character-based form to a GUI form. This issue will require considerable programming effort, regardless of the methodology chosen. Fundamental differences exist between a character-based implementation of a form and a mouse-driven GUI-based implementation. In a character-based form, data entry is performed strictly through the keyboard, using the enter key, on a field-by-field basis. Appropriate verification code is associated with each field to check the validity of the data entered before allowing the user to proceed, sequentially, to the next field. This makes data entry verification a simple, straightforward process. In a mouse-driven GUI form, the user can move randomly from field to field via the mouse, entering data. This complicates the verification process significantly, requiring extensive software

modifications for each form to implement the proper business logic between field movements. Since the user can perform a mouse-click on any field, it is preferable to grey out certain fields until the user has entered valid data in a visible field. Proper code must be developed for each field determining which fields to make visible after valid data is entered in the current field. Again, this problem will require substantial new code development, regardless of which methodology is chosen for the conversion effort.

C. Method 1 - X-Windows Emulation

1. Description

This approach entails using ORACLE SQL*Forms with automated forms conversion and X-Windows emulation. An automated forms conversion program would accept the existing ORACLE SQL*Forms 3.0 forms and convert them to the most current SQL*Forms 4.x environment. The application and its forms would reside on a remote machine or host, while each individual PC would become a terminal to access the application via communication software.

2. Evaluation

Although some form of a GUI can be implemented using automated forms conversion alone, its look and feel will not adhere to user expectations, which are currently defined by a Microsoft Windows environment. Certainly, no performance gains will be experienced by the user under this approach. Network traffic will increase substantially in order to transmit the image of each form. For a large number of concurrent users, therefore, network traffic could become excessive, rendering performance unsatisfactory to the user community. In addition, each form would require software modifications to implement the proper business logic between field movements; to adhere to strict database integrity constraints, new verification code for each field would be required. This presents a significant task. Also, this approach would require new development of signature card software. Due to this combination of factors, i.e. potentially prohibitive network traffic, additional software development, and especially, failure to meet user expectations, this methodology is not recommended for achieving a true GUI environment for ISMP applications.

D. Method 2 - Oracle Developer 2000

1. Description

This approach entails using ORACLE SQL*Forms with automated forms conversion as well as additional forms redesign using the ORACLE Developer 2000 software. This would require converting existing forms in SQL*Forms 3.0 using an automated forms conversion program. The converted forms would then be redesigned

in an MS-Windows GUI environment using the ORACLE Developer 2000 software. The cost per developer copy is approximately $4000, with authorization for unlimited distribution of any application developed with the software. During actual operation, information would be transferred from the converted PC application to a host machine and vice versa by the use of ORACLE's client-server software. The software, SQL*Net, would handle the communication of all data requests to and from the server where the main ORACLE database resides. Unlike the previous approach, however, the SQL forms could be maintained on the individual desktop machine or on LAN servers at each site.

2. Evaluation

This approach could produce a GUI that would meet user expectations, in terms of its look and feel. Maintaining the forms on the individual desktop or LAN server may appear advantageous in some respects, i.e. reduced network traffic, but it would pose difficulties for version control, requiring maintenance of current forms at many locations around the Corps. More importantly, this approach will require considerably more effort, in terms of manpower and time, than the first approach: In addition to creating generic forms with automated forms conversion, developers must also build true MS-Windows forms from the generic forms, using the ORACLE Developer 2000 software. Also, the converted forms will present the same problems discussed for the previous approach: Mouse movement verification code will again have to be developed for each field to maintain database integrity and signature software must be created for the MS-Windows GUI environment. Another disadvantage presented by this approach, however, is the perpetuation of a very high level of vendor dependence. In addition to continued reliance on ORACLE SQL*Forms as the template for the interface, each developer must use the ORACLE Developer 2000 software for the redesign into a Windows environment. Accomplishing the conversion in this manner would make changing DBMS vendors extremely difficult and expensive, if not impossible, at some later stage in the system evolution. This factor raises serious questions regarding the long-term viability of this approach. Thus, while this approach is likely the most expedient to produce an acceptable GUI environment, it will involve a major effort, and the long-term effects must be carefully considered.

E. Method 3 - Generic MS-Windows Design

1. Description

This approach entails the use of one of several alternative software tool sets which can be employed to design an MS Windows-based application GUI to access any type of DBMS. These developer tools execute in an MS-Windows environment and produce MS-Windows applications with unlimited distribution rights, with a cost per developer package of less than $2000. Any of these tools could provide access to ORACLE or other databases on both local and remote platforms. Each user PC

would require a TCP/IP package properly configured to connect to the database (assuming it is remote) as well as the communication software specific to the database. For example, the ORACLE database requires ORACLE SQL*Net for remote connections. This approach is currently being used in the development of PROMIS. The tool set used, in this case, is an Ada-based Software Engineering Environment (SEE), although the discussion of Method 3 does not apply to only that environment.

2. Evaluation

This approach offers several advantages. It would reduce the level of vendor dependence associated with the previous method. It would also allow taking full advantage of the GUI features and capabilities which could be gained from a redesign of the application interface. If a database redesign should be undertaken concurrently with the GUI conversion, revisiting the design of each form would be advantageous. This approach would obviously, however, require the most extensive manpower and resources of any of the three methods described. It would entail complete redesign of each form using the appropriate software, without the aid of any automated forms conversion software or ORACLE Developer tools. However, for any major software development 'new-start', this would be our recommended method.

F. Estimated Schedule and Budget

An estimated schedule and budget are provided in the report. The schedule and budget are rough estimates of the resources needed for conversion of each ISMP AIS to a GUI environment, using the three methodologies detailed in this report. These estimates should not be used as final budgetary numbers for presentation to acquire funding for any GUI conversion effort for existing ISMP AIS. More detailed investigation of each application would be needed to provide more accurate estimates for scheduling and budget. The breakdown of costs begins on page 25. Total estimated costs as well as estimates of elapsed time for each ISMP AIS are given in the following tables:

CEFMS	Total Elapsed Time	Total Cost
Method 1	24 Months	$2,750K
Method 2	30 Months	$3,800K
Method 3	36 Months	$4,300K

REMIS	Total Elapsed Time	Total Cost
Method 1	18 Months	$553K

Method 2	24 Months	$1,039K
Method 3	30 Months	$1,289K

RMS	Total Elapsed Time	Total Cost
Method 1	N/A	N/A
Method 2	18 Months	$1,013K
Method 3	24 Months	$1,263K

No data is provided for PROMIS since it is being developed using the target technology. Therefore, no conversion will be required.

G. Recommendations

All ISMP AIS should be stable and considered fully operational before any conversion to a GUI environment can be undertaken. No "beta-test" mode software should be a candidate for conversion to a GUI. In the case of CEFMS, functionality changes, as well as problem repairs, are still occurring. Implementation of CEFMS remains in "beta-test" mode. This allows developers to repair problems and add functionality on a daily basis. Hence, CEFMS cannot currently be considered in full production. Additionally, the Corps of Engineers should strongly consider whether a permanent dependency on the ORACLE Corporation is in its best interest. This decision is critical in determining the methodology for GUI conversion. If ORACLE is to remain the sole RDBMS for ISMP applications, in perpetuity, the second approach would be the most expedient method for converting existing applications to a GUI. However, if the Corps decides, at some later stage, to use another RDBMS, using vendor-specific software for GUI design would not be wise. In conclusion, we recommend that no GUI implementation be considered until fully productional ISMP AIS are fielded within the Corps. At that time, the magnitude of the conversion effort must be weighed against the value that the change from a character-based to a GUI interface would deliver. The cost justification of conversion alone, without revisiting the fundamental functionality and design of the application, is doubtful. In other words, the best time to design and implement a GUI for the application would occur when the time arrives to use 'lessons learned' from deployment of version 1.0 of an ISMP application to design a version 2.0. When such redesign is undertaken, the effect of being tightly shackled to one software vendor for the life of the system, perhaps 20 years or more, must be seriously considered. We do not truly have 'open systems' if they are not open to changes in software vendors as well as hardware vendors. If we are bound to a software vendor as we once were to hardware vendors, we have likewise restricted the evolutionary flexibility of our systems. It is

well-documented that lack of evolutionary flexibility is a major cause of high life cycle costs for large software systems.

II. INTRODUCTION

A. Background.

The Information System Modernization Program (ISMP) is currently comprised of four Automated Information Systems (AIS). These include the Corps of Engineers Financial Management System (CEFMS), the Real Estate Management Information System (REMIS), the Resident Management System (RMS), and the Programs and Project Management Information System (PROMIS). Each of these systems was designed to satisfy the functional requirements of a certain user community. Methodologies employed to develop the first three systems, CEFMS, REMIS, and RMS, allowed them to run in the standard executing environment available. For the most part, this was a character-based environment, well-accepted at the time, in which the user would respond to a series of prompts for input from the application. These responses were entered by the user from the keyboard.

During the time of development of these ISMP AIS, the computer industry experienced dramatic changes in the information technology arena. The manner in which information could be transmitted within an individual application changed from a character-based technology to a more sophisticated mouse-driven technology. The past three years have seen substantial increases in the power of computers, both on the users' desks and in data processing centers world-wide. Additionally, dramatic changes have occurred in the development of the user interface for each application. Users, no longer satisfied with the previous standard in desktop computing (DOS-based mode), have come to expect a Windows environment.

The availability of low-cost computer power that makes the Windows environment fast and convenient has helped bring about this change in user expectations. Users are aware of the technology available as well as the ease of use and sophistication that can be achieved when the technology is fully implemented. Hence, information systems which were developed only a short time ago, using well-accepted methods and technologies, now seem inadequate.

B. Why the need for a Standard GUI.

The shift in the de-facto standard for desktop computing to a MS-Windows environment has made it desirable that the ISMP systems be revised to take advantage of this technology. Graphical User Interface (GUI) technology has progressed such that it is now technically feasible for ISMP systems, as a whole, to present a uniform, consistent user interface that can be more conveniently and quickly navigated by the end user. A GUI provides a user the technique to invoke operating system operations using a graphical screen rather than using a command line, as in DOS. The operating

system also provides an Application Programming Interface (API) that allows creation of a customized GUI to the application.

The user community demands the highest quality of user interface design, similar to the commercial software that can be purchased off-the-shelf. These software applications should appear to function as integrated systems that are powerful, user-friendly, and self-documenting. A standard GUI for ISMP systems would allow a user, for example, to simply click on an icon to move easily from one application to another. This level of integration among applications is currently provided by commercial software. User expectations of ISMP systems are being shaped by such sophisticated applications that are available at very modest prices and are in fact being used daily by the Corps.

C. Purpose of Report.

Converting existing ISMP applications from a character-based mode to a standard GUI environment under MS-Windows should not be undertaken in haste. One must first know the requirements for ISMP applications to provide users with modern, easy-to-use GUI's. In general, a number of issues must be explored to make sound decisions on converting ISMP applications to a GUI. First, what process will be needed to make the transition from the current character-based mode to the graphical mode? Secondly, what tools will be needed to accomplish this goal? Additionally, what infrastructure changes will be required? What costs will be incurred by the transition? Are the schedules feasible? How will the transition impact the user community?

Creating a GUI-based application to replace a character-based application, particularly a very large application (such as CEFMS), is not a simple 'conversion' process. It is more akin to a total new start, beginning from the software design phase and onward; in other words, it constitutes a major endeavor.

If conversion tools can be used effectively to move to a GUI environment from a character-based mode, the process will cost less and take less time. However, if these conversion tools cannot successfully generate a GUI-based application that meets user expectations and conforms to infrastructure constraints, it becomes necessary to explore other methods for converting these ISMP applications to a GUI environment.

Targeting a MS-Windows 3.1 or later environment for the GUI design is the only feasible option. Deciding whether or not to use MS-Windows 3.1, NT, or Windows 95 is not a significant issue since a properly designed GUI-based application can execute under any of these operating systems. The purpose of this report is to explore the advantages, disadvantages and readiness for ISMP AIS to move to a

standard GUI environment. Alternative methods for achieving a standard GUI environment for ISMP systems are discussed.

D. Assumptions.

All currently operational ISMP AIS were examined in their present state of operation. No modification to any existing remote or local database used currently by the ISMP AIS was proposed as part of the conversion process. In other words, the findings of this report were based on the data and functionality for each ISMP system.

E. Contents of the Report.

This report describes the current mode of operation for each ISMP AIS. The current execution environment (hardware, software, and communications required to support the execution) for each ISMP system is discussed. Common requirements for updated GUI-based systems are highlighted. These are discussed in a broad sense and do not constitute a detailed requirements specification.

Based on current technology, three general GUI implementation alternative methodologies are presented, including the advantages and disadvantages of each approach. These include an ORACLE SQL*Forms approach with emulation to an X-Windows environment, an ORACLE SQL*Forms approach in a MS-Windows environment, and a generic MS-Windows approach.

Conclusions and recommendations are provided for migrating each ISMP system to a standard GUI environment. Estimated "strawman" schedules and budgets are presented for each ISMP system to be fully migrated to a standard GUI environment.

Appendix A contains a discussion of client-server computing and non-client-server computing. This is an introductory discussion of the technology which handles remote database access as well as the advantages and disadvantages of each approach.

III. GENERAL INVESTIGATION ISSUES

The investigation of providing ISMP application systems with a MS Windows-like user interface raises multiple, complex questions, the resolution of which are essential to the objective.

In considering the issues, the following criteria were developed for comparison and evaluation of the alternative methods:

1) Characteristics of the GUI. How will the converted GUI appear to an end-user? Will the user's expectations be met by the design of the GUI? Is the GUI user-

friendly? A GUI environment, as currently perceived by Corps users, is defined as a MS-Windows environment which contains forms, list boxes, dialog boxes, and help screens comparable to modern MS-Windows applications that can be purchased off-the-shelf. Anything less will not be considered a true GUI environment by the user community.

2) Performance. Will a GUI environment improve or degrade performance for the user community? Conversion to a GUI will be futile if the application seems unacceptably slower to the users.

3) Network Traffic. What impact will a GUI environment have on network traffic if images of GUI forms have to be stored on a remote machine and transmitted to each individual user's PC for processing?

4) Version Control. How can distribution of the most current version of the application to the user community be accomplished if changes to the application occur frequently? This will be affected by the methodology used to implement a GUI.

5) User Exits and Triggers. Most current ISMP AIS have additional processes that are executed from within the application itself. Some of these processes are initiated by events that take place when data is entered or edited in a particular field on a form. For example, suppose a certain field on a form requests a user's social security number. After data has been entered in the field, an event (or process) checks the validity of the social security number and the user's name. If the data is correct, the application may allow the user to move to another field. However, if the data is incorrect, an error message may appear and the user may be required to re-enter the data before proceeding. These processes that are associated with verification and validation of data entry per field are called "user exits" in ORACLE terminology. Additional processes associated with each form may be initiated once a user enters a particular keystroke. For instance, a user may complete data entry on a particular form and select a "Save" option. When the option is selected, not only is the remote database updated, but additional processes, such as reports generation, can be "triggered" to start on the remote server. Hence, these are referred to in ORACLE terminology as "triggers". User exits and triggers can be short, as described here, or can be lengthy. Can these processes be converted cost effectively? How can these processes be included properly in a GUI environment?

6) System Configuration. What costs are incurred by additional software and/or hardware needed to augment the standard PC configuration so that the converted AIS executes properly? If conversion to a GUI is feasible, what costs are associated with Corps-wide deployment of the application?

7) Other. Some ISMP AIS (namely, CEFMS) contain specialized software and hardware to accomplish electronic signature. The requirements to allow this feature to properly execute in a GUI environment must be identified.

The issues outlined above were investigated for each of the ISMP AIS and form the basis of this report. However, the focus of the investigation centered on the proposed conversion of CEFMS to a standard GUI environment. This is the most prominent application, i.e. largest, most expensive, and most critical, of the ISMP AIS. It also uses ORACLE SQL*Forms. Therefore, the feasibility of converting CEFMS presented the most important issue to be determined.

One final, important issue, mouse movement verification, should be discussed in regard to the conversion from a character-based form to a GUI form. This issue must be addressed by each methodology and will require a significant effort, regardless of which methodology is chosen. Fundamental differences exist between a character-based implementation of a form and a mouse-driven GUI-based implementation. In a character-based form, data entry is performed strictly through the keyboard, using the enter key, on a field-by-field basis. Appropriate verification code is associated with each field to check the validity of the data entered before allowing the user to proceed, sequentially, to the next field. This makes data entry verification a sequential, straightforward process. A mouse-driven GUI form allows the user to randomly move from field to field with the mouse, entering data, with no changes submitted for update of the database until the user either performs a "Save" operation or attempts to exit the form. This allows the user more flexibility in data entry, but significantly complicates the process of verification. Since the user can move from field to field via the mouse with no restrictions, it is necessary to make certain fields "invisible" (grayed out) to the user until proper data is entered in a visible field. These restrictions on mouse movement are critical to maintaining data integrity. Proper code must be developed for each field regarding which additional fields should be made visible after leaving the current field. For example, a form to allow a new grocery store item to be entered in the database contains three fields, bar code, item description, and price. The business logic for this form in a mouse-driven GUI should be to grey out the item description and price field until an appropriate bar code is entered. Once a bar code is entered in the field, verification code associated with that field should check to determine if the bar code exists in the database. If the bar code is not a legitimate new code, an appropriate error message should be displayed. If the bar code is acceptable, the item description and price fields should be made visible to the user for data entry. Conversely, if all fields were visible, the user, after entering the erroneous bar code, could mouse-click on the price and item description fields, enter data, and press the appropriate key to save the data in the database. This would place an erroneous record in the database, compromising database integrity. The verification code necessary to address this issue will require a significant programming effort, regardless of which methodology is chosen for GUI implementation of ISMP applications.

IV. STATUS OF EXISTING ISMP AIS

A. CEFMS

CEFMS is currently operational at six different sites in the Corps of Engineers. The application executes on a remote host machine; each PC functions as a dumb terminal, using communications software to achieve the network connection to the host. CEFMS was developed using ORACLE SQL*Forms 3.0 and contains over 1200 forms. Additional processes, i.e. user exits and triggers, are spawned from within CEFMS on the host machine. These processes are written in COBOL and C and contain approximately 180,000 lines of code. Signature card hardware and software are implemented on some PCs to allow specific users to execute that role from within the application. CEFMS is scheduled for deployment to the South Atlantic Division (SAD) and the remaining districts of the Southwest Division (SWD) during FY 96. CEFMS is still running in a "beta-test" mode environment. This means that code enhancements and functionality changes continue to occur. Hence, the current version of CEFMS is not considered "final" or static.

B. REMIS

REMIS is an ORACLE-based system that is used to track information about real property owned by the Corps of Engineers. Currently, it is deployed at 33 different sites (31 districts and 2 operating divisions) and is primarily used by real-estate branch personnel. Data from REMIS is uploaded during non-working hours to the CEFMS database. REMIS is developed with ORACLE SQL*Forms 3.0, the same software tools used for CEFMS, although on a smaller scale. It contains approximately 350 forms, and 5000 lines of code associated with user exits and triggers. Similar to CEFMS, REMIS uses network communications software to connect to the host machine: it executes entirely on the remote server machine with each PC using communications software to gain the necessary connectivity to the host. Currently, REMIS is running in both ORACLE 6 and ORACLE 7 mode. A full conversion to ORACLE 7 for all districts is planned for FY 96.

C. RMS

RMS is currently developed using the Clipper and dBASE programming languages. This application runs on an individual PC and stores and retrieves data from dBASE and Clipper databases which also reside locally on each PC. Both databases employ the dBASE file structure. The stand-alone nature of the application on each PC is necessary since the program is used at construction sites Corps-wide. The application is loaded on individual laptop computers and taken directly to the construction site where data is entered. Once the user returns to his/her home station, database files are then uploaded to a central site for entry into a consolidated

ORACLE database. Presently, interfaces are available for uploading information to the Corps of Engineers Programs and Project Management Database (CEPPMDB). Interfaces to PROMIS and CEFMS are projected for late FY 96. Communication software (VistaCOM) currently handles all file transfers via a modem to the CEPPMDB. The RMS administrator handles the duties of importing each user's Clipper and dBASE files into the CEPPMDB. The CEPPMDB is then used for reporting purposes only. Specialized reports have been developed in ORACLE*ReportWriter and SQR which extract data from the CEPPMDB and create reports used by headquarter's personnel on a quarterly basis.

D. PROMIS

PROMIS is currently under development and is scheduled for deployment during summer FY 96. Its main purpose is to provide a standardized, integrated information system to support management of projects and their allocated resources within the Corps primary missions of Civil Works, Military Programs, and Hazardous, Toxic, and Radiological Waste (HTRW) Clean Up. PROMIS addresses capabilities for identifying and tracking project scopes, schedules, budgets, costs, contracts, modifications, and technical performance requirements for management and control of individual projects through planning, design, construction, and initial operations. In addition, PROMIS will derive the aggregate data from individual projects for use in the Congressional budgetary and programming processes of the Corps. The objective of PROMIS is to design, develop, and deploy a modernized information management system supporting the business practices of programs and project management throughout the Corps. PROMIS is being developed using a generic MS-Windows GUI design, adhering to the Ada mandate. It will interface with two modernized systems (CEFMS and REMIS) as well as legacy systems (PRISM, AMPRS/MCPRS, LRS, and M-CACES). It will provide standard reports using data from all interfaced systems. PROMIS is being developed using a GUI which will execute on an individual PC and use remote network connection software to transmit and receive data requests to and from an ORACLE database on a host server. The application will reside on each user's PC and execute in a MS-Windows environment. PROMIS is designed for client-server computing (see Appendix A).

This report will not address conversion of PROMIS to a standard GUI interface since PROMIS is currently under development using a generic MS-Windows design methodology. However, that methodology will be discussed later in this report as one of the alternative methodologies for converting the other ISMP applications.

V. GUI IMPLEMENTATION ALTERNATIVES

A. Introduction

Three different methodologies were investigated for conversion to a standard GUI for existing ISMP AIS. The methodologies fall into two general categories. The first category begins with an automated conversion of existing ORACLE Forms and was investigated since both CEFMS and REMIS are ORACLE SQL*Forms applications. ORACLE SQL*Forms is software that is used to design and implement forms for a particular application. Although ORACLE Corporation does not currently market its own version of a forms converter, third party forms converters by Ace corporation and Kumaran Systems, Inc. are available. These forms conversion programs accept existing ORACLE SQL*Forms (Version 3) as input and output 'raw' ORACLE SQL*Forms (Version 4.x compatible). Note that the output of the conversion program will not be ready-to-use: substantial additional work must be accomplished by the developer before conversion to a true GUI, as perceived by Corps users, is complete. Methods 1 and 2 fall into this first category of using automated forms conversion. Method 3, a generic MS-Windows GUI design (similar to the approach used in PROMIS) falls into the second category, involving a basic redesign of the application user interface, rather than beginning with the use of automated forms conversion software. The advantages as well as disadvantages of each general category are explained in the descriptions and discussions of the three methodologies.

Method 1 is an implementation of ORACLE SQL*Forms using automated forms conversion with the entire converted application residing on the host machine and user access via an X-Windows GUI emulator communication package. This is considered non client-server computing (see Appendix A) and is similar to the current mode of operation of CEFMS and REMIS.

Method 2 is another implementation of ORACLE SQL*Forms using automated forms conversion with the application residing on each individual PC. This is considered client-server computing (see Appendix A). This method requires the use of ORACLE Developer 2000 software to complete the redesign to a true GUI, making it vendor-specific.

Method 3 is a generic MS-Windows GUI implementation and does not use any proprietary ORACLE software package in the design. This method also employs client-server computing (see Appendix A).

Each methodology is discussed below, highlighting specific issues that are critical for conversion of existing ISMP AIS to a GUI environment.

B. Method 1 - X-Windows Emulation

1. Description - The application and its associated forms would reside on the remote machine (host), and the individual PC would function as a terminal to access the application via communications software. Automated forms conversion software would be used to convert the existing ORACLE SQL*Forms 3.0 forms to the most current ORACLE SQL*Forms 4.x environment.

2. Discussion of Issues

a. Characteristics of the GUI - This GUI requires each PC to be configured with an X-Windows terminal emulator for connectivity to the host machine where the converted application will reside. The X-Windows emulator, a communications package similar to VistaCOM, translates each form definition sent from the host application to an individual form on the PC with a menubar at the top. The mouse can then be used to navigate from field to field on the form. When a user terminates a session, the host-server connection is also terminated, and the user's environment for normal PC usage is restored. This GUI implementation does not have the 'look and feel' of a MS-Windows application. Therefore, users will not be satisfied with this implementation as a true GUI. Also, for reasons discussed below, the converted AIS will likely run slower than the previous non-GUI version, and thus, will not meet user expectations regarding performance.

b. Performance - This method does not take advantage of the PC's processing power. The PC functions only as a dumb terminal. No performance gains will be experienced by using this method; in fact, performance will likely degrade due to increased network traffic (see discussion below).

c. Network Traffic - The amount of information which must be transmitted over the network is substantially greater for this methodology than for the current character-based mode or for the other two methods. The bitmap imaging of each form must be transmitted from the server to each PC. This bitmap imaging contains not only information about each field and control option, but also contains color codes for each pixel (or dot) on the screen. Transmitting this volume of information from the host to each PC can cause the user to experience substantial delays.

d. Version Control - Configuration management issues will not be significantly different from the version control requirements for maintaining the current versions of CEFMS and REMIS. All of the forms required by the application as well as the application executable code reside on the host server. Hence, any changes or additions which are made will be immediately available to each user each time the application is executed.

e. **User Exits and Triggers** - The trigger applications for CEFMS are currently written in COBOL and C and reside completely on the host machine. This methodology leaves the entire converted GUI application on the server, including all trigger applications. This provides the advantage that the code for exits and triggers will not require conversion. It is important to note, however, that trigger applications are bound to particular forms, not the database. Since this methodology uses ORACLE SQL*Forms, it requires the use of ORACLE vendor-specific library functions to spawn the processes for trigger applications on the server. Hence, if this methodology is selected, ORACLE software must be used, which perpetuates vendor dependence.

f. **System Configuration** - This methodology requires the use of X-Windows terminal emulator software on each individual PC to access the converted forms in a GUI environment. Hummingbird's eXceedW package which runs in a MS-Windows environment is an example of such an X-Windows emulator. During the actual execution of the application, GUI-like forms are sent to the PC over the network. The cost per copy for this Xwindows emulator is approximately $300. Also, for specific users, a signature card board must be installed and configured properly.

g. **Other** - The signature card software for CEFMS presents another problem in a GUI environment. CEFMS currently uses the signature software via the VistaCOM software package. This feature communicates with CEFMS in a significant number of places. At the present time, the signature card software does not work in an X-Windows environment. This software will have to be developed specifically for the X-Windows emulation. This task may require a significant amount of effort and could prohibit the use of this methodology for GUI standardization. Also, mouse movement verification code must be developed for each data entry field. This will require substantial new code development.

In addition, to create and change the forms after they have been converted to a GUI environment, each ISMP AIS developer must use an ORACLE product, ORACLE Developer 2000, X-Windows version, designed specifically for this task. (Note that the third party converters cannot be used for forms maintenance after conversion.) The cost for this product is approximately $4000 per developer. Since the converted system under this methodology remains an ORACLE product, the CEAP contract handles all licensing as currently implemented.

C. Method 2 - ORACLE Developer 2000

1. **Description** - ORACLE corporation produces a developer software tool set, ORACLE Developer 2000, which runs in a MS-Windows environment. The

software provides developers the tools needed to design forms for existing ORACLE databases, whether local or remote. Each existing form in SQL*Forms 3.0 can be converted by using automated forms conversion. These converted forms must then be transferred to an individual PC or a file server for final design. Each developer must purchase and use this software package in order to effectively change any new form converted with an automated forms converter to SQL*Forms 4.x. Also, to create and change the forms after they have been converted to a GUI environment, each ISMP AIS developer must use this software package. Note that the third party converters cannot be used for forms maintenance after conversion. The cost per copy is approximately $4000. Unlimited distribution of any application developed using ORACLE Developer 2000 is authorized. Hence, CEFMS and REMIS forms would first be converted using automated forms conversion and then moved to an individual PC or file server where developers can complete the design to a MS-Windows GUI environment. Once the forms have been converted to SQL*Forms 4.x, ORACLE proprietary software must be used to effectively make any changes to these forms. Hence, this methodology perpetuates vendor dependence upon ORACLE.

2. Discussion of Issues

a. Characteristics of the GUI - Users will see a true MS-Windows GUI environment with this methodology. The GUI requires each PC to have a copy of MS-Windows 3.1 or later and a copy of ORACLE SQL*Net. As the application is initiated from a MS-Windows icon, connection to the remote ORACLE database is performed automatically, transparent to the user. The user can then navigate and query information from each form by use of the mouse, as with any MS-Windows application. Data requests are sent from the application to the main ORACLE database on the server through the connection established during initiation of the application. ORACLE's client-server software, ORACLE SQL*Net, handles the transfer of information between the converted PC application and the host server for the main ORACLE database. Each form has the appearance of a standard MS-Windows form with a menubar at the top. When the application is terminated, the connection is dropped automatically, and the user is returned to the MS-Windows environment. No additional communication software, such as VistaCOM, is needed for this methodology.

b. Performance - The processor on each PC is effectively utilized for painting screens and executing business logic; only SQL script is transmitted from the PC to the database server. Assuming the power of a modern PC is sufficient for existing MS-Windows applications, it is expected that the performance experienced by the user would largely depend on the power of the remote host to act as a database server. This means that the faster the server

can process data requests from the PC application, retrieve that data from the database, and transmit the data back to the application, the better the performance. Also, the user can start the application, minimize it to an icon, execute another MS-Windows application, and then return later to the original application by simply restoring the icon from its minimized state.

c. **Network Traffic** - Since each form can be painted by the individual PC, no form image bitmap information (see paragraph B.1.c above) need be transmitted over the network as was described for Method 1. Only data requests will be transmitted to the host. These include SQL SELECT, INSERT, UPDATE, and DELETE commands. The host then responds with the appropriate data which populates the defined form for updating or viewing purposes. This methodology will likely reduce the network traffic significantly over the present CEFMS technology. Additionally, each user initiates the application on his/her PC rather than remotely connect to a server to initiate the application, as in Method 1. System resources required on the server are also reduced since the memory needed to execute each instance of the application resides on the PC and not on the server. Currently, every instance of CEFMS or REMIS requires system resources (memory, disk space, CPU utilization, etc.) on the server. This methodology transfers some system resource burdens to the PC and frees the server to handle data requests from the application.

d. **Version Control** - Two differing approaches can be taken for implementing this methodology. One approach entails installing the converted forms and the application on each PC, meaning every user has his/her own copy of each form and the application. The other approach is to install the converted forms and the application on a file server which is easily accessible by the user on a LAN. Each approach presents separate version control considerations. If the converted forms reside on a file server, configuration management would be more cumbersome than that stated for Method 1. However, version control can become problematic if the application and all forms reside on each individual PC. For example, if the developers of the application add a new enhancement or repair a problem associated with a particular form, the revised form must be updated on each individual PC running the application. A file server can be used to contain the forms and the application rather than distributing a copy to each individual PC. Configuration management can then be accomplished by updating the revised forms on the file servers. Hence, each time a user executes the application, the current version of each form is always available. However, either approach can be problematic if not implemented properly. Formal version control processes and supporting software are needed. A specific configuration management guidance document should be written detailing the version control procedure and its implementation. Any modification to existing software should be performed in strict adherence to the guidelines of this document.

e. **User Exits and Triggers** - Triggers attached to each form must be configured to execute properly in a MS-Windows PC environment. Again, this means that the trigger applications which are now started on the server from the current ISMP AIS must be started from the PC application to execute on the remote host. Using CEFMS as an example, trigger applications reside on the server and are started by each user from within a particular form by a certain keystroke. If CEFMS (or any other ISMP AIS) is converted and moved to each PC, the issue of how the trigger applications are initiated must be resolved. ORACLE Corporation provides software to initiate these triggers on the remote server machine from within the PC application. Nonetheless, even with the use of this software, a significant amount of effort will be required to properly implement this strategy. The trigger applications, however, can remain on the remote server machine. Hence, the conversion effort for this task will be significantly less than if all trigger applications were to be redesigned and moved to each PC for execution. Finally, it should be noted that similar to Method 1, ORACLE vendor-specific software is required for this implementation, which again perpetuates vendor dependence.

f. **System Configuration** - Each individual PC executing the application does not require a communications package such as VistaCOM to successfully execute the revised application using this methodology. However, each PC must be configured with a proper TCP/IP package to connect to the remote system where the database resides. Additionally, each PC must have software that communicates directly with the remote ORACLE database. This software, ORACLE SQL*Net, controls the flow of information from the PC application to the server and vice versa (see paragraph C.2.c above). Hence, once a particular item is queried or selected for editing or viewing in a form, the appropriate data is retrieved using SQL*Net and placed in the form in a MS-Windows GUI environment. Unlimited distribution of the converted application and its forms is authorized by each purchase of the ORACLE Developer 2000 software package. Hence, if this technique is chosen for GUI development, there will be no additional charge for distribution of the application to a large number of users corps-wide. Site licenses for ORACLE SQL*Net software can probably be acquired through the CEAP contract. ORACLE'S SQL*Net costs approximately $200 per copy.

g. **Other** - Conversion of the signature software again presents a significant task. This software must be redesigned for access in a MS-Windows GUI environment. In addition, a new key mapping must be generated for the MS-Windows environment. The standard NOS100 mode will require conversion to take advantage of the MS-Windows key mapping. Also, mouse movement verification code must be newly developed for each data entry field and will require a significant programming effort. Finally, to create and change the forms after they have been converted to a GUI environment, each ISMP AIS

developer must use ORACLE Developer 2000; third party converters cannot be used for forms maintenance after conversion.

D. Method 3 - Generic MS-Windows Design

1. **Description** - Software tools exist which allow designing a MS Windows-based application GUI to access any database management system (DBMS). The developer software runs in a MS-Windows environment and produces MS-Windows applications with unlimited distribution rights. The cost per developer tool set is less than $2000. All of these packages provide database connectivity support to ORACLE and other databases on both local and remote platforms. Each user PC must again be configured with a proper TCP/IP package to connect to remote systems if the database is not local. Also, each PC must have a copy of the communication software for the specific database used on the remote server. For example, if the database is ORACLE, ORACLE SQL*Net is required. If the database is Informix, Informix Corporation markets a product similar to ORACLE SQL*Net which must be purchased for each user. This approach is currently being used in the development of PROMIS and can satisfy the Ada mandate. Ada or any of several other languages may be used to develop the client application which executes on the PC.

2. Discussion of Issues

a. **Characteristics of the GUI** - The appearance and function of this GUI is similar to that produced by Method 2. The application is started in the MS-Windows environment by clicking on an icon. Connection to the remote database is handled transparently via DBMS specific software, such as ORACLE SQL*Net. Data requests are sent to the server from the application. The data is returned over the network and placed in the forms that are designed with the application software. Again, the look and feel offered by this methodology is the same as that of Method 2. The primary difference lies in the design technique: no specific ORACLE-based software product is used in this methodology. All forms are completely redesigned. User satisfaction will be similar to that for Method 2; the converted application will have the appearance of any MS-Windows application.

b. **Performance** - This methodology takes advantage of the PC's processing power to perform screen management, execute business logic, access local files, etc., removing some of the burdens from the host. The application contains all necessary forms on each individual PC. These forms are included within the application executable itself. Network traffic is minimized (see discussion below). Performance using this method should be as good or better than current performance.

c. **Network Traffic** - Network traffic under this methodology is similar to that described for Method 2 in paragraph C.2.c above. Data requested and the resulting data flow from the remote database to the application on the PC produce the only network traffic. The speed of the application is controlled primarily by the speed with which the server software can process the data requests from the application on the PC.

d. **Version Control** - This methodology presents the same version control issues as those for Method 2. Configuration management may become problematic if each user has a separate copy of each form and the application. This methodology may have all forms contained in the program executable (meaning the .EXE file). Any addition or modification to any particular form will require the use of a new executable file by every user. Distribution of executables via file servers can reduce the number of copies needed to be distributed. As was stated in this discussion for Method 2, configuration management requires very careful and systematic planning. Formal version control practices should be developed and strictly followed. Documentation should be developed which specifically details the implementation strategy for version control.

e. **User Exits and Triggers** - This methodology may require conversion of some processes which now execute on the server to a PC environment. This could become a significant task given the volume of code that is currently associated with applications such as CEFMS. Currently, in CEFMS, trigger applications, e.g., reports, are associated with certain forms and require user selection before starting on the server. A generic MS-Windows GUI should move the control of starting these trigger applications from user selection on a form to the remote database on the server. For instance, once a user sends a "Save" request from a particular form to the remote database on the server, certain elements in a table may be updated. Additional trigger applications can then be started on the server when a particular element is updated in the database. This transfers the control of starting the trigger application from the form to the database. This would require a significant redesign of the way CEFMS and REMIS are developed. This methodology requires no specific ORACLE software and hence, does not perpetuate vendor dependence. Each form per application will require "redesign" from the current ORACLE methodology described in Method 1 and Method 2. Additionally, some reports, now spawned on the server, can be converted to take full advantage of the reporting capabilities of the GUI environment. MS-Windows commercial packages are available which can generate these reports rather than executing them on the remote server, using remote system resources. For example, it may be advantageous to use a product such as MS-Access (or other report generators) to generate reports that were previously generated remotely on the server using software developed in COBOL or C. Conversion of user

exits and triggers will require a significant effort in terms of manpower and time if this approach is selected.

f. **System Configuration** - The PC configuration required for this method is similar to that described in paragraph C.2.f for Method 2. Each individual PC executing the application will not need a communications package such as VistaCOM to successfully run the converted application. However, each PC must be configured with a proper TCP/IP package to connect to the remote system where the database resides; also, each PC must have a copy of ORACLE'S SQL*Net for data request transfers to and from the ORACLE database on the server. The recommended target configuration for good performance in a MS-Windows GUI environment should be the following or better:

1) 486/66 computer system
2) 8 megabytes of RAM
3) Super VGA monitor
4) Mouse
5) Microsoft Windows 3.1 or later
6) FTP TCP/IP 2.11 or later
7) ORACLE SQL*Net 1.1.8 or later

This method places no restrictions on distribution of the application software. However, ORACLE SQL*Net must reside on each machine to allow proper data transfer to the remote ORACLE database. As stated above in paragraph C.2.f, this could probably be purchased on the CEAP contract for Corps-wide distribution.

g. **Other** - The same concerns described above in paragraph C.2.g regarding the signature card software conversion and key mappings for a MS-Windows GUI environment are pertinent for this methodology. Also, code to handle the many events associated with mouse movement between fields on each form must be newly developed and will entail a substantial programming effort.

VI. ESTIMATED SCHEDULE AND BUDGET

A. **Caveat** - The following estimated schedule and budget are our judgments after considering some basic magnitude measures of each application, e.g., number of forms, number of lines of source code, etc., and the effect of the issues discussed previously. They are rough estimates of the resources needed for conversion of each ISMP AIS to a GUI environment using the three methodologies detailed in this report. These estimates should not be used as budgetary numbers for presentation to acquire funding for any GUI conversion effort for existing ISMP AIS. More detailed

investigation of each application should be undertaken to give more accurate estimates for scheduling and budget. The scope of this study did not allow for that level of detailed examination of each of the applications.

The estimates given below for each system are IN ADDITION TO any ongoing work to debug, field, maintain and support the current versions of each ISMP AIS. Fielding costs shown below reflect only the costs of commercial off-the-shelf software (COTS) that must be installed on each user's PC. Additional fielding costs will likely be incurred with each method and will depend on fielding approaches chosen.

Please note that for RMS conversion, method 1 is not applicable since the current application is not written in SQL*Forms.

B. Projected Time to Complete and Costs

CEFMS	Begin Fielding	Conversion Cost	Fielding Cost	Total Cost
Method 1	24 Months	1000K	1750K	2750K
Method 2	30 Months	2500K	1300K	3800K
Method 3	36 Months	3000K	1300K	4300K

REMIS	Begin Fielding	Conversion Cost	Fielding Cost	Total Cost
Method 1	18 Months	500K	53K	553K
Method 2	24 Months	1000K	39K	1039K
Method 3	30 Months	1250K	39K	1289K

RMS	Begin Fielding	Conversion Cost	Fielding Cost	Total Cost
Method 1	N/A	N/A	N/A	N/A
Method 2	18 Months	1000K	13K	1013K
Method 3	24 Months	1250K	13K	1263K

VII. SUMMARY

A. Conclusions

The following paragraphs summarize conclusions regarding the main issues, discussed in the body of the report, for each methodology. Before summarizing the issues, however, it should be reiterated that regardless of the methodology chosen, the conversion from a character-based form for data entry to a mouse-driven GUI form will require extensive new code development to accomplish the necessary mouse movement verification.

Method 1 would be the most expedient method to use if the GUI environment were acceptable. Existing forms could be converted quickly with automated forms conversion. However, the GUI environment that is produced by this methodology is not desirable. It will not meet the performance expectations of the users. Substantial programming will be required to incorporate all business logic between field movements and maintain database integrity. The network traffic needed to define and display each individual form from the remote server to the PC will be increased substantially. The Ada mandate will not be addressed by use of this methodology. Signature card software must be developed to properly work in an X-Windows emulator GUI environment. Vendor dependence will be perpetuated: purchase of ORACLE's Developer 2000 package will be required each developer to make changes or design any new forms. The cost and time required are the least of the three methods, yet still substantial.

Method 2 could be considered the second most expedient method to use for conversion to a GUI environment. This methodology will produce a modern, MS-Windows-based GUI. User expectations can likely be satisfied by this implementation methodology. Network traffic will be significantly reduced over Method 1 since only data will be transmitted. No form images will be transmitted over the network. This methodology will also require substantial additional programming to adhere to strict standards of business logic and database integrity. The user exits and triggers associated with each form must be examined carefully, and code to "spawn" or start these applications on the server must be developed, a non-trivial task. Also, the Ada mandate again will not be addressed and vendor dependence will be perpetuated. Conversion using this methodology will require a substantial investment in time and money.

Method 3 can also produce a modern, MS-Windows-based GUI. In addition, network traffic will be minimal: data requests only, with no form bitmap image information, are transmitted over the network. This means that the appearance as well as the performance of the GUI produced by this method will meet user expectations. Therefore, in regard to these two aspects, performance and appearance, Method 2 and Method 3 rate comparably. Method 3 differs from Method

2 as well as Method 1, however, in two additional aspects. First, Method 3 can be implemented in compliance with the Ada mandate, as is currently being accomplished with the development of PROMIS. Secondly, this method does not perpetuate vendor dependence. While eliminating vendor dependence presents many advantages, it also contributes to making the conversion time and cost for this method the highest of all three. Nonetheless, this initial investment in time and expense should be weighed against the long-term consequences of being bound to a single software vendor for the life of the system.

B. Recommendations

This report has examined some of the issues related to conversion of ISMP applications to provide users with modern graphical user interfaces. We believe the issues examined are the most important to determine conversion viability and make reasonable recommendations. However, this study was not an exhaustive examination of the applications or the technologies available. It was necessarily limited in duration and scope. It is also important to recognize that software technologies are currently changing rapidly, and conclusions and recommendations regarding the most viable methodologies today may not apply in the future. The following are our recommendations, based on the apparent state of the applications and the technologies available today.

1. No ISMP application should be considered a candidate for conversion until the application is fully debugged and stable, and considerable operational experience has been gained. At the present time, CEFMS is considered to be in "beta-test" mode. This means that functionality changes, program fixes, and enhancements occur frequently. Hence, no conversion to a GUI environment should begin until a version of CEFMS is released, in production, and stable.

2. The recommended time for conversion for any application will occur when a major update of that application's functional content, data content and fundamental design is being undertaken. In other words, the time to convert an application to employ a GUI will occur when operational experience indicates the necessity of initiating a major project to move from version 1 to version 2.

3. Conversion using Method 1 described herein is not recommended.

4. Conversion using Method 2 should be undertaken only if a thorough analysis of total life-cycle costs supports the cost-effectiveness of committing to dependence on a single software vendor (ORACLE) for the life of the system.

5. Conversion using Method 3 is the recommended approach when the magnitude of the system changes being implemented make the effort the equivalent of a major application redesign. This method can be used to adhere to the principles

of software engineering that the Corps has adopted, as well as the Ada mandate. This method, however, is not 'quick and dirty' and requires a substantial investment in time and money.

APPENDIX A

A. Client-Server Computing.

1. Description.

Client-server computing is a new networking computing paradigm that optimizes network usage, and greatly facilitates data analysis and decision making by end users. Client-server networks are comprised of back-end servers that are responsible for data storage, organization and retrieval, and front-end systems running applications that manipulate, analyze and present data to the user in an intuitive way.

An application developed exclusively for client-server computing means that the program executable resides on an individual machine (PC). This machine is what is referred to as the client or front-end system. The server functions only as a database retrieval system. The server receives data requests from the application on the client, processes the request, and sends the data requested back to the client machine for processing. The server is referred to as the back-end system. The application on the client machine must then interpret the information received from the server and handle it appropriately.

The client or front-end system contains the application. It also contains the necessary networking software to link to the database on the server or back-end system. The server contains the database itself and the software which responds to the database requests from the application on the client. This software is specific for each type of remote database that is used. For example, ORACLE has its own version of remote database software. Also, Informix contains its own server database software. Figure A1 depicts the client-server architecture.

In the example below (see Figure A2), the SQL database server (ORACLE) is responsible for data storage, record locking and processing other data requests. The client computer runs an application that is compliant with the Open Database Connectivity (ODBC) standard and simply queries for specific information from the server. The server retrieves data and sends it back to the client. The client application then manipulates and displays the information to the user via various functions of productivity tools like ORACLE SQL*Forms, Microsoft Visual Basic, Microsoft Excel, etc.

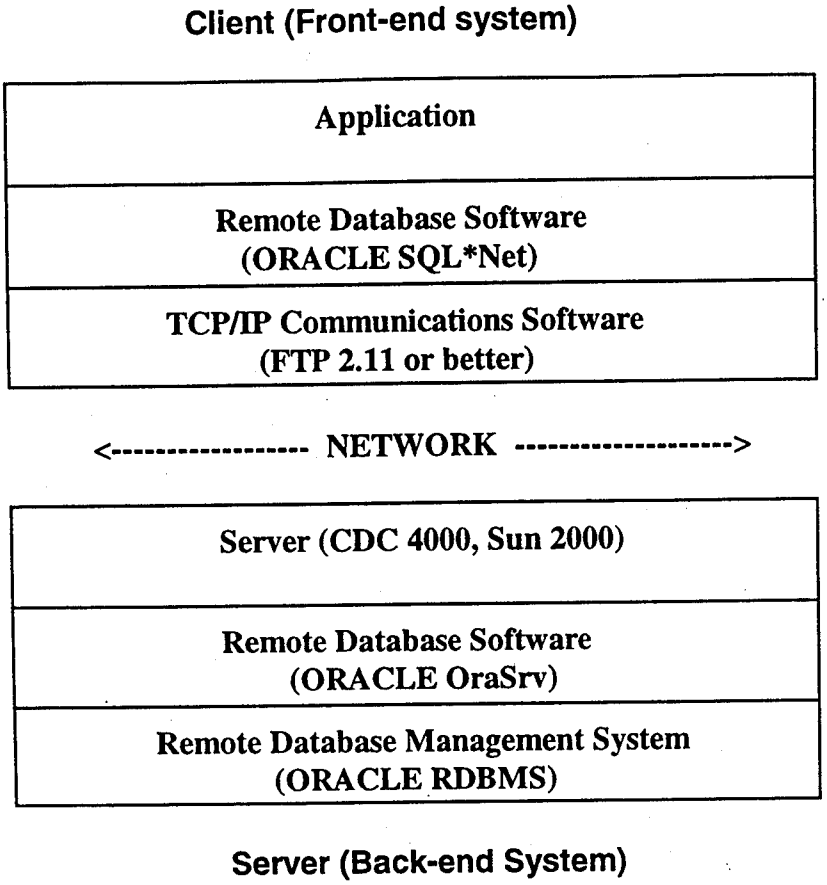


Figure A1. Client-Server Architecture.

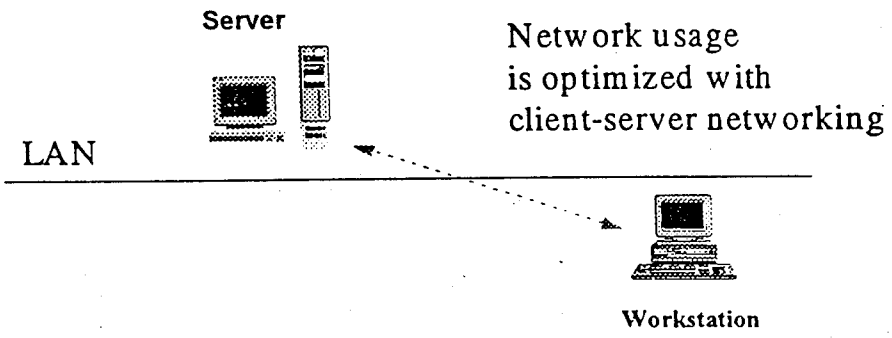


Figure A2. Client-Server Computing.

2. General Advantages.

i. The majority of the computing power is handled by the individual client machine or front-end system (PC). Hence, performance can be substantially improved for some applications since the user can invoke the application very quickly from his/her PC.

ii. The GUI software resides on each client machine which does not place a burden on the communications network. If all forms reside on the server or back-end system, each time the application on the client requests data to be retrieved and displayed for viewing or editing purposes, not only is the data retrieved but the entire form image must also be transmitted over the network. This places a large burden on the communications network and can degrade performance substantially. However, network traffic in this model is limited to:

a) a query, or data request, from the client workstation or front-end system

b) specific data requested by the client workstation or front-end system

iii. The application can connect to multiple servers or back-end systems for data requests. The entire process is transparent to the user. Each time a user requests data from the application on the client, the data request may be sent to a number of servers to retrieve the data without the user's awareness of which server has been accessed. This allows the database to be spread across multiple servers which can benefit performance.

3. General Disadvantages.

Application maintenance can present a problem. Each time a change is made in the GUI or problems have been repaired in the application, corrected software must be sent to each client (executable, form definitions, etc.) This can present a serious configuration management problem. The developers of the application must clearly state how this will be accomplished in order to avoid significant version control problems.

B. Non-Client-Server Computing.

1. Description.

In the past, data access and retrieval across networks has not usually followed the client-server networking paradigm. In the non-client-server model (see Figure A4), a server stores data in a database, and client workstations interact with applications that are responsible for directly manipulating the database. The application actually runs on the server and handles such tasks as opening the database, locking records, and requesting and retrieving data. An application developed using the non-client-server computing technology is typical when the server is a mainframe. The entire application resides on the server or back-end machine (forms, executable, database, additional programs, etc.) The client or front-end system acts only as a dumb terminal once the connection to the server is established via communication software. Figure A3 depicts the non-client-server architecture. Notice that the burden of the work is handled almost entirely by the server and not the client.

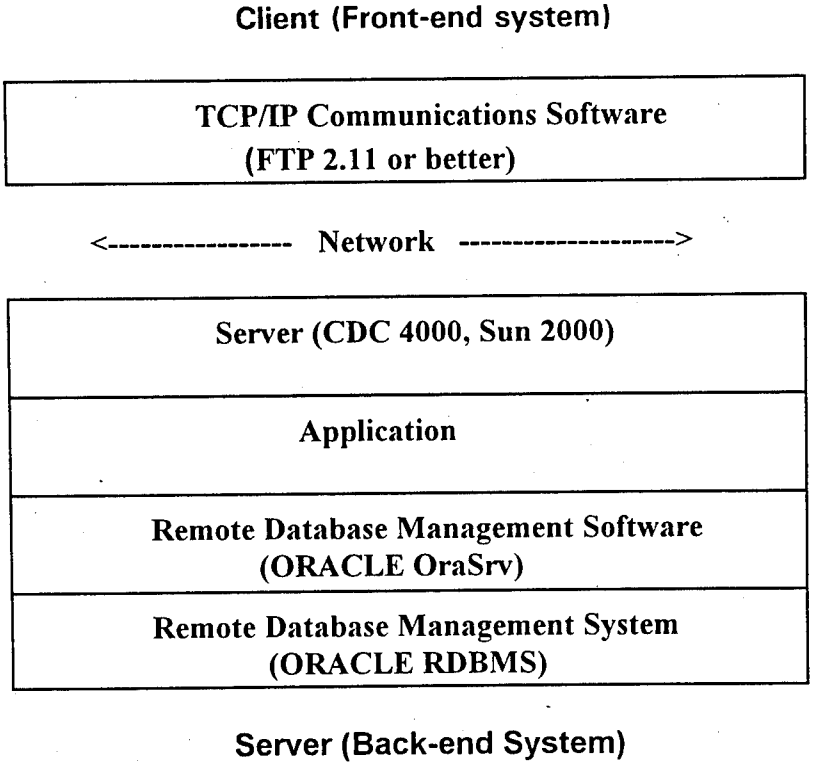


Figure A3. Non-Client-Server Architecture.

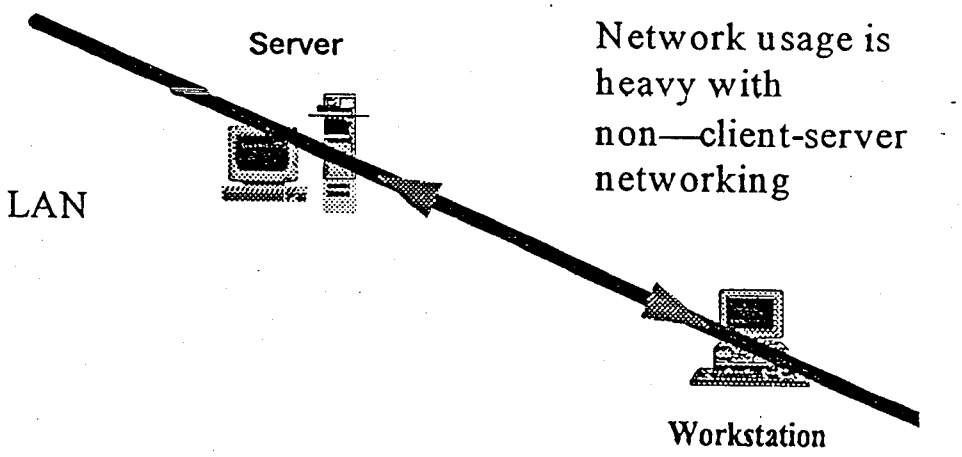


Figure A4. Non-Client-Server Computing.

2. General Advantages.

i. Configuration management is simplified because the entire application resides in only one place, the server or back-end system. If a bug is repaired or a new feature added, the server or back-end system is the only place where changes have to be incorporated. The client or front-end system receives no impact. The current version of the software will always be available on the server without any user intervention.

ii. Client machine configurations can be rather simple and inexpensive since the machine acts only as a terminal.

3. General Disadvantages.

i. The server can become overloaded with data requests spawned by a large number of users simultaneously running the application. The server's memory is finite. Once a large number of users begin the same application, memory is reserved for each instance of the application by the server. This degrades performance and results in dissatisfaction with the application design.

ii. The client machine's computing power is not being effectively used. Since the only need for the client machine is connectivity to the server, the client's processor is not being utilized to distribute the workload.

iii. Network traffic can become problematic. In the non-client-server model, because applications open entire databases and reside totally on the server, network traffic is not optimized. This puts a heavy load on the corporate network and impacts performance of attached workstations (clients). Performance degradation is caused by the requirement that every form, menu, or screen, along with the data that is requested by the application on the server, be transmitted over the network in its entirety to each client for the application to run successfully.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1996	Final report

4. TITLE AND SUBTITLE
Investigation of Issues for Conversion of the Information System Modernization Program (ISMP) Automated Information Systems (AIS) to Use a Graphical User Interface

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael E. George, Julia A. Baca

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER
Technical Report ITL-96-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

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

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report examines important issues that must be addressed if the U.S. Army Corps of Engineers (USACE) Information System Modernization Program (ISMP) Automated Information Systems (AIS) are to be converted to a graphical user interface (GUI) environment. Three methods for achieving the conversion are presented. Two of the methods begin with the use of automated forms conversion software. The third method is based on a redesign of the interface. Criteria for evaluating the three methods are defined. Each method is then evaluated according to the criteria. The report concludes with a summary critique and recommendations for the conversion effort.

14. SUBJECT TERMS

Automated forms conversion	Graphical User Interface (GUI)
Client-server computing	Terminal emulation
Configuration management	User exits and triggers

15. NUMBER OF PAGES
42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED		