US Army Corps
of Engineers
Construction Engineering
Research Laboratories

USACERL Technical Report FF-95/10
April 1995

Development of a Transparent Computer Application Distribution System for the Directorate of Civil Works

by
Laura L. Harmet, Edward J. Japel, Scott Maxwell, and Wayne J. Schmidt

The number of computer programs and updates developed and distributed by the U.S. Army Corps of Engineers continues to grow. Processing, archiving onto diskette, and mailing this software can cost the Corps thousands of dollars and dozens of man-hours each year. Furthermore, diskettes are subject to damage or delay, which can create extra unnecessary costs for the Corps and its customers.

This report documents the development of GAPPL (Get Application), a computer application distribution system designed to provide users in the Directorate of Civil Works quick, cost-effective access to group software and updates. GAPPL was designed to work transparently to the computer novice while offering the experienced user capabilities for customizing certain settings to better meet individual needs.

GAPPL has been in use by the Civil Works Operations and Maintenance Branch since 1992. A return on investment for GAPPL has not been calculated, but experience with another electronically distributed Corps application suggests that distribution savings for a single new application to 40 districts may amount to almost $7200 annually. GAPPL is designed to distribute up to 30 applications.

USER EVALUATION OF REPORT

REFERENCE: USACERL Technical Report FF-95/10, *Development of a Transparent Computer Application Distribution System for the Directorate of Civil Works*

Please take a few minutes to answer the questions below, tear out this sheet, and return it to USACERL. As user of this report, your customer comments will provide USACERL with information essential for improving future reports.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

_____

_____

_____

2. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.)

_____

_____

3. Has the information in this report led to any quantitative savings as far as manhours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

_____

_____

4. What is your evaluation of this report in the following areas?

   a. Presentation: _____

   b. Completeness: _____

   c. Easy to Understand: _____

   d. Easy to Implement: _____

   e. Adequate Reference Material: _____

   f. Relates to Area of Interest: _____

   g. Did the report meet your expectations? _____

   h. Does the report raise unanswered questions? _____

i. General Comments. (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.)

_____

_____

_____

_____

_____

_____

5. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____

Telephone Number: _____

Organization Address: _____

_____

_____

6. Please mail the completed form to:

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORIES
ATTN: CECER-IMT
P.O. Box 9005
Champaign, IL 61826-9005

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1995	Final

4. TITLE AND SUBTITLE
Development of a Transparent Computer Application Distribution System for the Directorate of Civil Works

5. FUNDING NUMBERS
CWIS
32719

6. AUTHOR(S)
Laura L. Harmet, Edward J. Japel, Scott Maxwell, and Wayne J. Schmidt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratories (USACERL)
P.O. Box 9005
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER
FF-95/10

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, U.S. Army Corps of Engineers (HQUSACE)
ATTN: CECW-OM-B
20 Massachusetts Ave. NW
Washington, DC 20314-1000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Copies are available from the 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)

The number of computer programs and updates developed and distributed by the U.S. Army Corps of Engineers continues to grow. Processing, archiving onto diskette, and mailing this software can cost the Corps thousands of dollars and dozens of man-hours each year. Furthermore, diskettes are subject to damage or delay, which can create extra unnecessary costs for the Corps and its customers.

This report documents the development of GAPPL (Get Application), a computer application distribution system designed to provide users in the Directorate of Civil Works quick, cost-effective access to group software and updates. GAPPL was designed to work transparently to the computer novice while offering the experienced user capabilities for customizing certain settings to better meet individual needs.

GAPPL has been in use by the Civil Works Operations and Maintenance Branch since 1992. A return on investment for GAPPL has not been calculated, but experience with another electronically distributed Corps application suggests that distribution savings for a single new application to 40 districts may amount to almost $7200 annually. GAPPL is designed to distribute up to 30 applications.

14. SUBJECT TERMS
Get Application (GAPPL)
Computer Application Distribution System
Communication/Computer Systems
Civil Works

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	Unclassified	SAR

Foreword

This study was conducted for the Directorate of Civil Works, Headquarters, U.S. Army Corps of Engineers (HQUSACE) under "Civil Works Investigations and Studies"; Work Unit 32719, "Management Tools for Civil Works Operations and Maintenance." The technical monitors were John Perez and David Harmon, CECW-OM-B.

The work was performed by the Facility Management Division (FF) of the Infrastructure Laboratory (FL), U.S. Army Construction Engineering Research Laboratories (USACERL). Michael Golish is Acting Chief, CECER-FF, and Alan W. Moore is Acting Chief, CECER-FL. The USACERL technical editor was Gordon L. Cohen, Information Management Office.

LTC David J. Rehbein is Commander and Acting Director of USACERL, and Dr. Michael J. O'Connor is Technical Director.

Contents

1 Introduction

Background

The number of computer programs being developed, distributed, and updated by the U.S. Army Corps of Engineers (USACE) grows every year. This increase has created a distribution challenge for Corps software maintainers and developers. New applications and updates typically have been distributed on diskette through the mail. This distribution method can take many administrative hours to process—and literally days for delivery to the user. Many users cannot do their jobs without these software programs. Any disk-handling damage or delivery delays can seriously interfere with productivity and the ability to meet deadlines.

Because mainframe computers offer many users access to shared data at the same time, a software developer may give many users simultaneous access to a new or updated program by loading it onto a mainframe. Electronic connectivity between mainframes and personal computers (PCs) allows users to download the application over telephone lines (by modem) immediately upon release. Unfortunately, electronic distribution can seem almost impossible to software users not familiar with the operating systems for each piece of hardware involved in the transfer process. Before downloading files from a mainframe, modem settings, network settings, and mainframe settings must be initialized. This task can unnerve novices and hinder their productivity.

In previous work, for the Corps Directorate of Military Programs, the U.S. Army Construction Engineering Research Laboratories (USACERL) developed and implemented a user-friendly mainframe-based application distribution program—PC Dugout—in support of the Military Construction, Army program (Japel et al, May 1991). PC Dugout was designed specifically to distribute software transparently within the Programming, Administration, and Execution (PAX) environment. However, Dugout was not required to be portable for application outside of PAX.

The Directorate of Civil Works subsequently identified the need for a similar distribution system for Civil Works applications. The system not only would be required to operate effectively, economically, and transparently, it would have to be portable for application beyond the then-current Civil Works computing environment.

The distribution system also would have to be compliant with the Corps of Engineers Automation Plan (CEAP), which was still under development at the time. Finally, unlike Dugout, the new application distribution system would be PC-driven rather than mainframe-driven. USACERL was tasked to develop the system.

Objective

The objective of this research was to develop a PC-based system for reliably and transparently distributing Civil Works software applications to remote client computers.

Approach

The fundamental requirement for the system was that it reliably and transparently distribute any computer application to any user group, even one comprising many novice users. The system was conceived as a broad application, that is, GAPPL would operate the same way in any number of different instances, able to manage different application libraries for different unique work groups. In creating the prototype, the GAPPL software developers selected a representative Corps software developer and set of users: the developer was the Civil Works Operations and Maintenance (CW O&M) branch, and the targeted users were Engineer District personnel involved in the CW O&M budgeting process. These offices were chosen because of the researchers' previous experience with their requirements and the software applications used to meet these requirements.

Core design considerations were specified and criteria for system development tools were identified. The prototype system was developed for use within MS–DOS.*

Mode of Technology Transfer

Documentation and training materials have been developed for GAPPL and are now being used in the field (Harmet and Japel, August 1992). The Directorate of Civil Works, O&M Branch, administers the update and release of Corps applications to District and Division personnel.

---

* MS–DOS: Microsoft Disk Operating System. Also referred to as "DOS" in this report.

Client system requirements are:

1. a PC using MS-DOS version 3.0 or later, at least 4 Mb of free disk space, and 580 Kb random-access memory

2. a login ID for the mainframe on which the application library resides, and read-only access to the application library (available from the work group's GAPPL administrator)

3. a current copy of the GAPPL executable program file.

Administrator system requirements are:

1. a PC using MS-DOS version 3.0 or later, at least 4 Mb of free disk space, and 580 Kb random-access memory

2. a current copy of the GAPPL administrator PC executable program

3. administrator access to all end-user mainframe files.

The GAPPL executable file and documentation are available from the USACERL Workforce Improvement Team (CECER-FFK), commercial telephone 217-373-6718.

Application developer requirements are:

1. use of standard Windows-compliant programming tools

2. support of object-oriented graphical user interface

3. access to TCP/IP connectivity or modem.

2 Design Considerations

The five main design considerations governing the development of GAPPL were:

1. ease of use
2. reliability
3. cost-effectiveness
4. version tracking
5. openness.

Ease of Use

Some computer users feel comfortable with applications that display technical information on-screen while a program is running. However, many others find technical information less helpful than visual symbols, pictures, menus, and choice lists. When deciding on which method to implement, the developer needs to look at the targeted user. Normally, if a typical user has substantial experience with computers, then providing technical information within the application is appropriate. In a communications program, such technical information includes display of the login process and the names of all files being transferred. On the other hand, if the typical user is a computer novice, then hiding such processes or making them transparent is more appropriate. Any decisionmaking required would be via menus or choice lists. Most users targeted for GAPPL had limited experience with computers. For this reason, hiding the technical information was the best approach to user interface design.

Reliability

Data Transfer

One aspect of reliability is that the product must dependably transfer data. Line noise and transmission errors due to mainframe glitches are the main factors that interfere with data-transfer reliability. There are several ways to combat data transfer irregularities. One lies in the developer's choice of transmission protocol—good error-

correction features are necessary. Another way to combat transmission errors is to go through a nationwide network (or panoramic network), also using error-correction ability. A third way is to use an error-correcting modem, but this is typically a user choice. The application developer must be careful about choosing more than one error-correcting device because too many can slow the data transmission. Data transfer failures undermine a user's trust in a product, especially if the user is a novice, but a slow transmission rate can greatly discourage use as well.

Connection

Another aspect of reliability is that the product must be able to connect through any path to the mainframe. As shown in Figure 1, GAPPL must support four connectivity paths. The user can use an individual PC through a modem, or a local-area network (LAN) through network modems. Using either method, the user can then dial directly to the destination mainframe or dial into a nationwide panoramic network that has access to the destination mainframe—CDCNET, for example. Within these four connectivity paths is a variety of local and panoramic networks, each with its own nuances. By design, GAPPL must maintain reliability regardless of which connectivity path the user takes.

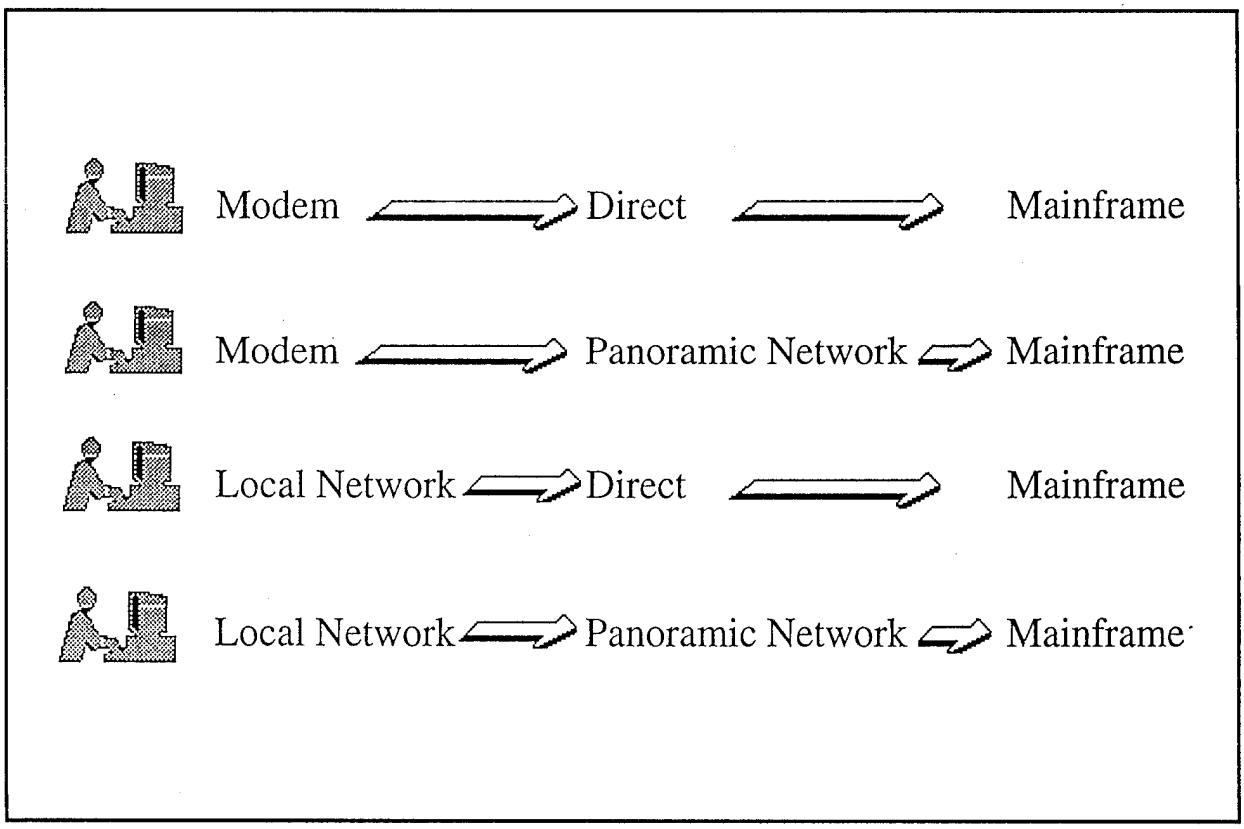


Figure 1. Connectivity paths.

LANs presented a unique problem for GAPPL developers. With so many different brands of LAN technology and data switches available, it is impractical to allow for dedicated access by every possible type. The developers considered enabling GAPPL to watch the user connect to the mainframe and remember the process. Unfortunately, a "watch me" program can only learn one rigid procedure; it cannot deal with exceptional situations such as a simple busy signal. A more practical design was to allow the user to navigate his or her own path up to a common point of connection. At this common point, where all connection paths converge, the user returns control to the GAPPL system to continue the connection process.

Cost-Effectiveness

The conventional method of application distribution required the formatting of many disks, loading the disks with the application and an installation file, and mailing the disks to all users. The developer has had to follow this process every time the program is updated. To further complicate matters, the distribution diskette must make it through the mail system without being damaged or corrupted, and the user must then install the application correctly on his or her PC before software distribution can be considered successful. If any of these steps cause physical or electronic disk corruption, the developer must repeat all distribution steps. This cumbersome process wastes both money and labor.

Time

When distributing applications from a mainframe via telephone lines, each user can get the data whenever it is convenient. The developer—or in the case of GAPPL, the administrator—need only make one distribution version of an application and transfer it to the mainframe. This results in a substantial decrease in time required of the distributor. On the other hand, the time spent by the user to retrieve a new or updated application increases, which may discourage use of the product. The design objective, then, was to shorten data transfer time as much as possible. With asynchronous communications, short transfer times can be a challenge. Transfer rates are affected by the communications protocol, the mainframe, the telephone lines, and the modems.

Another factor that affects transfer time is the size of the file being transferred. The larger the file, the longer the transfer time. An archiving (data compression) utility can be used to reduce each file's size. However, this would require GAPPL to be able to create self-extracting archives because users may not have access to a compatible

unarchiving utility on their PCs. PKware archiving utilities were considered because the Corps site license from PKware* made this the most cost-effective alternative.

The archiving utilities are used to distribute the original application files. As changes are made to the system, the archived application does not have to be distributed in its entirety. A package that builds patches for the changes can be used to decrease the sizes of update files. RTPatch, by Pocket Soft, Inc.,** was chosen because of the timeliness of its release. (Other similar packages have since been released.)

The organization of archive files also affects transfer time. To archive an application, all the files and executables belonging to the application are usually compressed into one archive file. If an application were separated into smaller archive files, then transferring one of those smaller files as an update would take less time than transferring the entire application. Each smaller archive file would consist of files likely to change in an update. An application update would require transferring only the information that has changed instead of the entire application each time. As a result, the user connection time to the mainframe would be much shorter. Unfortunately, this method does not improve the transfer time for an entire application—in fact, it increases the transfer time by a small amount because the transmission protocol must spend a little time setting up each file in the set. However, since GAPPL users will be getting updates more frequently than whole applications, and since the additional overhead is negligible in any case, the GAPPL developers chose to split applications into multiple files.

When the application has been downloaded, it must be successfully installed to consider distribution complete. An easy-to-understand installation method is also required to save manpower time. GAPPL design demands an installation procedure to be consistent over each application so users will know how to successfully install any application. At the same time, each installation procedure is specific to an application. One application may require an installation program that prompts for many directories, while another may only require an *install* batch file that requires no parameters. Consequently, no common install program could be developed for all applications. To solve this problem, GAPPL designers required that all installers use the same command—INSTALL—to execute software installation. The Corps application developer customizes an INSTALL.BAT or INSTALL.EXE file to do whatever is needed; GAPPL merely looks for either type of file and executes it.

---

* PKware, Inc., 7545 North Port Washington Rd Suite 205, Glendale, WI 53217-3422

** Pocket Soft Inc., P.O. Box 821049, Houston, Texas 77282

Cost

Short transfer times allow shorter connection time with the mainframe over phone lines, which means smaller charges for processing time. If the user is going through a direct path, (e.g., stand-alone computer via modem), long-distance charges are an added expense. Connection time cost is not the only expense—initial distribution cost is also a factor. Depending on which development tools are used and which protocol is employed in developing an application distribution system, royalty fees may be required if it is issued to more than one or two users. To compare the expenses of distribution on diskette to distribution via GAPPL, the following lists of expenses must be considered:

Distribution expenses via GAPPL

- Administrator's labor to create one set of distribution files.
- Long-distance telephone call
- Processing time
- Royalty fees for distribution tools

Distribution expenses via diskette

- Administrator's labor for one set of distribution diskettes (times number of users).
- Postage
- Cost of diskettes

In a study of the Civil Works Automated Budgeting System, the cost of distributing that application by mail ($10 per user) to approximately 40 users was calculated at $7200 per year (*Return on Investment Case Study*, May 1992). An electronic application distribution system—the forerunner of GAPPL— developed specifically for ABS avoided most of that cost. It is estimated that a mature Corps application may need to be updated two to three times a year. (A new application may require 10 or more updates in its first year). GAPPL was designed to administer up to 30 applications for a single workgroup. It can be seen that a GAPPL instance used up to its full potential could save $36,000 annually in diskette distribution costs. (Logic: $10 x 40 [Districts] x 30 [applications] x 3 [updates] = $36,000.) GAPPL also cuts the administrator's labor costs because the distribution files need only be prepared once.

To make GAPPL cost-efficient, the expenses it incurs should not be more than the cost of diskettes, postage, and labor of distribution by mail. By design, two expenses for GAPPL were avoided. All external software used by GAPPL is already site-licensed by the Corps or are free for nonprofit distribution, so no royalty fees apply. GAPPL can also leverage the toll-free or local telephone numbers often used by panoramic networks.

Version Tracking

In order to implement the multifile-update design (see Cost Effectiveness—Time), GAPPL needs to know which clients have which versions of which applications. The ideal place to maintain this information, requiring no transfer time and thus the least cost, would seem to be the user's PC. However, if a user ever needed to access the same version of an application twice—which may be necessary if an error occurs during installation or an update is accidentally deleted—he or she would have to adjust the version number. This is not the best method because the majority of GAPPL users are assumed to have limited computer experience. Therefore, the version information should be maintained on the mainframe, where the administrator has access to it.

Another design consideration is the method by which version information is stored. There could be one file containing all version numbers of every application, or there could be one version file per application. Because a GAPPL objective is to lower cost by minimizing connection time, and because all version numbers are not required at once, using one version file per application is the better approach.

Openness

While the GAPPL executable was designed to reside on the user's PCs, some additional files also must be accessable by all users. These system files, such as the list of available applications, must reside on the mainframe. GAPPL had to be able to issue mainframe commands to access these files.

System openness encompasses two characteristics: portability and versatility.

Portability

In an open system, mainframe-specific commands cannot be interwoven with a PC executable. The mainframe-specific problems that need to be considered are accessing the mainframe, finding the GAPPL support files, and manipulating those files.

One way to keep a communications system portable is to choose a protocol with scripting capability. This capability allows the login script to be changed without changing the executable that drives it, thus allowing different mainframes to be accessed. Another way to keep a system portable is to use files to store mainframe-dependent variables, such as mainframe file references or macro definitions. It was

decided to implement both of these methods in GAPPL to keep the system highly portable.

Versatility

GAPPL had to be designed so it could transfer anything from a very small program like a .COM file for a PC to a very large program like the Civil Works Automated Budget System (ABS). To do this, GAPPL would have to work with some kind of format common to all computer programs.

The most obvious feature common to all programs is that they are binary. Many programs also include procedures to move or install an application after it has been transferred to the user's computer. Many applications have their own specific installation requirements, so an application's install procedure should be transferred along with the program software from the mainframe. Because install procedures may be written in text, GAPPL would have to be capable of transferring text files as well as binary. Therefore, to build in versatility, GAPPL defines a standard application as one or more binary files along with zero or more text files.

In addition to file types, file organization had to be considered. As discussed previously (see Cost Effectiveness—Time), a standard version of an application should consist of several archived files, each archive containing files that would be likely to change concurrently in an update. This standard facilitates quick update transfers. Another GAPPL standard promoting versatility is the use of a self-extracting archiving utility so the user need not acquire special decompression software. (As noted previously, this standard also reduces application transfer times.) Using these three standards, GAPPL can distribute any kind of application.

3 Criteria for Development Tools

Communication Protocol Requirements

A communication protocol is a set of rules for transferring information between computers. Transferring data between a PC and a mainframe is difficult because the hardware and operating systems are different. A communication protocol can navigate such differences so two dissimilar computers can communicate as if they were using the same hardware and operating system. Vistacom* and Kermit were the two protocols considered for use with GAPPL.

Wide Usage

To maintain portability (see Chapter 2: Design Considerations—Openness—Portability) the protocol must be operable on all kinds of panoramic networks, mainframes, and PCs. The mainframe that GAPPL was originally to interact with was the Washington Computing Center (WCC) machine. When designing GAPPL, the researchers knew the Corps would be standardizing on one computing environment, and that the WCC programs would be moved to that environment. But the destination mainframes were still unknown. The protocol that GAPPL implemented had to be guaranteed to work on both the WCC and the future mainframes. Vistacom, a Corps of Engineers standard, would almost certainly be accessible on the new machine. But Vistacom was not accessible from the WCC machine. Kermit, on the other hand, had a worldwide network of support and had been implemented on almost every type of mainframe. This protocol was already available on the WCC machine.

Error Correction

To promote reliability and user confidence, GAPPL must consistently transfer data without errors. Transferring data without errors is mainly the responsibility of the communication protocol. Most protocols transfer files in packets—chunks of data wrapped in protocol-specific information. Error-detecting protocols examine the wrapper to determine whether the enclosed information was corrupted during

* Control Data Corporation (CDC), 8100-T 34th Ave. South, Minneapolis, MN 55425.

transfer. Error-correcting protocols re-send corrupted packets until they get it right. Both Kermit and Vistacom detect and correct errors.

File Transfers

The protocol for GAPPL must also be able to handle two different file types (see Chapter 2: Cost-Effectiveness—Openness—Versatility): text and binary. Most protocols, including Kermit and Vistacom, support these file types with ease.

Expense

The main objective of GAPPL is to inexpensively provide wide timely access to applications. A protocol can affect cost in two ways: the speed with which it transfers a file, which affects how much the user has to pay for mainframe processing time; and the royalty fees for using it, which may cost the user or the distributor (depending on who buys it).

Speed. A protocol's transfer speed depends on its algorithm. The Vistacom algorithm is different from Kermit's, and they transfer files at a different rate. Transfer speed can also be affected by user-changeable protocol settings, such as packet size. In general, the larger the packet size, the faster a file will be transferred. However, large packets slow transmission on noisy telephone lines, since more data must be re-sent when there is an error. If the phone line is very clear, it is best to set the packet size as large as possible. Kermit allows the user to change the packet size; Vistacom does not.

Another feature that only Kermit supports is *sliding windows*. Kermit normally sends each packet (window) of information to the receiving Kermit, then waits for the receiving Kermit to check the packet and confirm succesful transfer. With sliding windows, the receiving Kermit will wait for the specified number of packets before responding. This allows the transfer to proceed much faster. As with the packet size, the user can set the number of windows. The limitation to the number of sliding windows is reliability: using a large number of windows may cause the protocol to spend more time re-sending information. A drawback of this feature is that it is not available on all versions of Kermit, so the GAPPL user may not be able to take advantage of sliding windows.

Direct costs. The direct cost of a protocol is that which the user has to pay. Kermit, a university-written and university-supported package, may be distributed free of charge as long as it is used in nonprofit packages. Vistacom is copyrighted, and the user must pay a royalty fee to use it.

Scripting

In order to make GAPPL's operation transparent, the data communications dialogue must be hidden from the user. The protocol should be able to execute commands without needing input through an elaborate user interface. Putting several protocol commands in a file is called *scripting*. Having a protocol execute a script is much like having DOS execute a batch file. Each command is executed sequentially, and there are control structures that the programmer can insert to give the scripts looping and checking features.

Kermit allows scripting and can even be executed from the command line without having to employ the user interface. Vistacom also allows scripts, but the scripts cannot be executed from the command line. The user must be within Vistacom, using its menu-driven interface. A package called VistaKit allows the programmer to micro-manage the communication processes. While it provides a great deal of control, the time it takes the user to become familiar with its functions is prohibitive.

Development Language Requirements

Portability

An application is portable if it can be moved easily from one operating system to another. To successfully make an application portable, the development language itself must be portable.

Size

Size is an important consideration when deciding which programming language to use for the interface/driver portion. The front end of GAPPL must be able to spawn communication processes, so it must not be so big that it consumes all random-acess memory (RAM) with overhead functions. RAM is not the only resource taken into consideration. Another consideration is that many of the targeted users' personal computers have a minimal amount of memory and little available disk space. Consequently, the interface cannot require a substantial amount of disk space.

The development language best suited to these criteria was the programming language C. The C language is easy to code and maintain. Programs written in C can be ported to other microcomputer architectures, and the language can be written at a very low

level to reduce memory requirements for computing overhead. Microsoft* QuickC was used in the development of GAPPL—the package's QuickC compiler generates small compiled modules that can efficiently be distributed from a mainframe. Also, QuickC requires no runtime module, so software written in it can be distributed and used without paying a royalty fee to the language developer.

* Microsoft Corp., 1-T Microsoft Way, Redmond, WA 98052-6399.

4 General GAPPL Features

The GAPPL user interface consists of menus, choice lists, and forms. These constructs make the application easy to use for the novice. Upon entering the GAPPL system, the main menu is displayed. The main menu includes three options:

1. set up and edit communication parameters
2. get an application
3. quit.

Within the communication setup form, the user sets common parameters such as communication port, telephone number, login ID, and password. These parameters provide features intended to give the user control over the unpredictable connection process and communication line. The control parameters include the variety of paths available, baud rate and parity choices, modem settings, and control of speed versus accuracy.

When the user decides to get an application, other features perform functions that are informative or increase reliability. These features include out-of-date notices, disk space checks, and error recovery procedures.

Communication Parameters

Paths

As discussed in Chapter 2, four basic connectivity paths are supported. The user can connect through an individual modem or through a bank of networked modems. Either way, he or she can then connect with the destination mainframe by calling it direct or by calling through a panoramic network that has access to the mainframe. When GAPPL was first implemented, the specific paths it supported followed the general paths set up by design. The user could go through a modem or through a network to a bank of modems. From there the user could (1) call the panoramic network TELENET, which offered access to the Washington Computing Center (WCC), or (2) call WCC directly. Later, when TELENET was deleted from the choice list because of its high cost, the user could only call WCC directly. Upon porting GAPPL

to the Corps of Engineers Automation Plan (CEAP) environment, the specific paths changed again. This time the user could not call directly to CEAP. The CEAP environment serves mainframes spread across the nation, linked through a nationwide network called CDCNET. Those mainframes could be accessed only through CDCNET.

In the communications setup form, two variables—route and mode—generate the four general paths.

Route describes the way the user is accessing the mainframe. Currently, a choice list allows the user to pick either CDCNET or manual connection. As stated above, CEAP does not allow direct access, only access via CDCNET. The manual route actually accesses CDCNET but gives the user greater control over the connection process. The manual route was provided for more experienced users who can use the extra control to their own advantage. It also provides a fail-safe path to the mainframe in case something changes in the general login procedure. Choosing CDCNET for the route automates the connection as much as possible.

Mode represents the device used to connect to the mainframe. The user can choose between modem or network. The reason for the difference is that a variety of networks and dataswitches are available. Providing specific commands for every network would not have been feasible. GAPPL initializes the port according to the baud rates and parities entered, then releases control for the user to access the modem from his network. Once he has reached a point that is common to all networks, the user returns control to GAPPL. The place where the user returns control—the *escape point*—depends on the path being taken. For instance, when connecting to WCC directly, the escape point was when the prompt "TSO" appeared, but when accessing WCC from TELENET, the escape point was when the prompt "CONNECT" appeared. When using a modem directly connected to the PC, the connection process is fully automated. There is no need for an escape point because control is never relinquished to the user.

Baud Rates and Parity

As in most communications packages, GAPPL includes a way to change parity and baud rate. The parities supported by GAPPL are even, odd, mark, and none. GAPPLs way of handling baud rates is slightly unusual. Instead of using just one baud rate, GAPPL requires two. This is because when connecting through a network, the user may communicate with the network at a different baud rate than the network communicates with the destination mainframe. The rate at which the user communicates with the network is called the *local baud rate*. The rate at which the network communicates with the destination mainframe is the *mainframe baud rate*. For users

connecting directly through a modem, the local rate and the mainframe rate are the same.

Modem Type

The type of modem affects the reliability of the connection. After the initial implementation of GAPPL, the researchers discovered that some users were having trouble getting their modems to respond correctly. Different brands of modems often have their own set of commands with which to set modem soft switches. Hayes* brand modems have a widely used set of commands, and many other modem manufacturers incorporate these Hayes commands to make their own products Hayes-compatible. GAPPL requires a modem to work in a standard way. To consistently get the user's modems to respond correctly, it uses modem commands to control modem return codes, dialing type (tone or pulse), and other settings (depending on the brand of modem). The modem types GAPPL recognizes, and requires specific settings on, are Hayes and Multimodem V.32 modems. Other types can be added easily.

Speed and Accuracy

Speed and reliability are sometimes mutually exclusive in data communications. The speed is related not only to the baud rate but also to the packet size (see Chapter 3: Protocol Requirements—Expense). Each packet has additional data associated with it, like beginning and ending data and cyclic redundancy checksums. Bigger packets generally reduce transfer times since this protocol overhead is a smaller fraction of the whole. However, longer packets of data are more likely to be interrupted or corrupted by static on the phone line. A packet must be re-sent if it is lost or corrupted during transmission, so larger packet sizes may actually increase the amount of data transferred. Increasing the number of retries also decreases the chance that the file will be successfully transferred because Kermit quits if it is forced to retry too often.

With smaller packet sizes, the span between beginning and ending packet data is shorter, but there is more data being sent because the file is divided into many more packets. Low baud rates and small packets somewhat decrease the likelihood of file transfer failure. GAPPL provides a speed/accuracy setting that gives the user three choices: quickest/reliable, quick/more reliable, and slow/most reliable. The quickest/reliable setting divides the file into the largest packets available, whereas the slow/most reliable setting divides it into the smallest packets possible. This capability was added after the initial implementation of GAPPL as an attempt to decrease failed

* Hayes Microcomputer Products, Inc., 5835 Peachtree Corners East, Atlanta, GA 30348.

transfers. Reducing packet sizes in this way gives the inexperienced computer user some control over technical communication variables.

Application Updating Features

Out-of-Date Notices

Because GAPPL keeps track of version numbers on the mainframe, it can notify the user when he or she does not have the current version of an application. Before the user is shown the list of applications to choose from, all version files are downloaded. An asterisk appears beside those that need to be updated. This feature was also added after the initial design of GAPPL because users were getting confused about which applications needed to be updated. The time spent transferring additional version files makes up for the time a user would spend checking on whether the application was up-to-date.

Disk Space Check

Transfer failure due to insufficient disk space on the client machine is a potential cause of lower productivity and user frustration. GAPPL checks the user's hard disk for adequate space before proceeding with the download. This feature also potentially saves much wasted connection time.

Error Recovery Capabilities

Besides using a protocol that detects and corrects errors, GAPPL provides a few other checks to maintain reliability. One way GAPPL does this is to log the interaction between the personal computer and the mainframe. GAPPL keeps two separate log files: one keeps track of the PC commands to the mainframe and the mainframe's response; the other records the status of every file transferred. This second log file tells whether a file transfer was interrupted, failed, or succeeded. It also tells how many bytes were transferred. When there is a problem connecting to the mainframe via GAPPL, or when the update was unsuccessful, the user can look at the log files to more fully understand the problem.

When downloading the update files, GAPPL writes the scripts so transfer of each update file will be tried twice before the transfer is considered unsuccessful. By trying twice, GAPPL may recover from line noise interfering with the transfer.

On the next attempt after a failed transfer, GAPPL will continue transferring the update from the point of failure. If the transfer fails in the middle of an update, the next time the user logs in and attempts to get the update, GAPPL will download only the files that did not make it through the first transfer. The user does not have to get all the files associated with an update or complete system during one session. Resuming from the point of failure avoids wasting time and money getting files the user already successfully received. It also provides the user a way to avoid noisy telephone lines. Many times, retrying a noisy connection a couple of hours or a day after failure can make a large difference in the quality of the connection.

5 Functional System Description

There are three main functional components to GAPPL: (1) the set of files it references on the mainframe, (2) the set of files it uses on the PC, and (3) the process by which it transfers data.

Files on Mainframe

The files on the mainframe are the ones that must be maintained by the administrator. Three categories of files reside on the mainframe: the *administration* files, the *application* files, and the *status* files.

The administration files consist of a list of *users*, a list of *groups*, and a list of *applications* available to each group. The list of applications is commonly called a *directory* because it gives other information pertaining to each available application, including where it resides on the mainframe and who is responsible for maintaining it. The list of users consists of all login IDs that have access to the GAPPL files and also lists which group the users belong to. A group is a set of users with access to a predefined set of applications. Each group has access to a different set of applications, which is reflected in the list of applications that appears when using GAPPL. A group's list of applications can be thought of as a *view*—a subset of the overall set on the machine. There is also an *admin* group, which has read/write access to the mainframe files; it defines the groups and their view of the applications. The groups maintained in the system are listed in the groups file.

The application files are those that comprise an application. Each application can be divided into member files. These member files are the set of organized archive files discussed in Chapter 2 (Design Considerations—Cost Effectiveness—Time). Dividing the application into modules yields a better way of updating applications. A user retrieves only the files needed for an update. Each complete application includes all of its modules (member files), an installation procedure file, and a GAPPL member file that lists the names, sizes, and other information about each module of the application. The GAPPL member file provides the information needed to build scripts that accomplish the file transfers.

A status file is maintained for each application/user combination. So, for example, if the administrator is distributing three applications, each user will have three different status files—one for each application. The status file is created when a user downloads or updates an application. If the user has never attempted to retrieve a particular application, the status file for that application will not be created. Status files track which user has which version of which applications. They enable GAPPL to know which files a user needs for an update, and document for the administrator who has what version and how they got it. This latter information is required for accurate technical support when a user has problems (see Appendix A).

Files on the Personal Computer

The major sections of GAPPL reside on each user's PC. To access the mainframe files, a user must have the GAPPL executable file and the Kermit executable file. Two groups of text files must also reside on the PC, in the same directory as the executables. One group consists of script files for Kermit. These scripts are used to set up the local copy of Kermit, log into the mainframe, set up the remote copy of Kermit, and transfer files. Some of the scripts contain variables that are set according to site specifications. The other group of text files contain wholly site-specific information used in the scripts. This information includes baud rate, parity, and the telephone number used to reach the remote mainframe. The user configures the system initially, and the settings remain the same until the user decides to change them (see Appendix A).

Transfer Process

When GAPPL is told to get an application, it first constructs scripts according to the settings in the communications parameters form. The scripts control which communication port, baud rate, and parity are used. The user has a choice of communicating via a modem or a LAN. This choice affects what kind of script is built. GAPPL must also build a script to get the user file from the mainframe. GAPPL needs this file to check whether the user is a registered GAPPL user and to determine which group he or she belongs in. The group determines the user's view of available applications.

Once the scripts are built GAPPL executes Kermit, which in turn runs the scripts. Kermit logs into the mainframe and transfers the users file. If the script fails at any point, Kermit logs the error and GAPPL informs the user of the problem. When the users file is transferred, GAPPL then checks the users file for the logon identification

(ID) used. If the ID is not in the list, GAPPL quits the process. If the ID is on the list, GAPPL retrieves the user's group directory (list of applications). GAPPL then constructs another set of scripts to transfer the list of applications based on the group, and again calls Kermit to execute the scripts.

Next, GAPPL shows the user the list of applications. Asterisks mark the applications that need to be updated. The user picks the applications to be updated in the current session. GAPPL then retrieves the user's status file and the member file for that application. The user must decide whether to download only the updated files or the entire application. In either case a script is built. If the user wants an entire application, every module of the application must be transferred. If just an update is desired, only the updated modules are transferred. In the latter case, only modules listed in the GAPPL member file with version numbers greater than the user status file version are transferred.

If the user has enough disk space for the transfer, GAPPL indicates how long the transfer will take and gives the user a choice to quit or proceed. If the user chooses to proceed, GAPPL calls Kermit to run the constructed script. A transfer screen for each module indicates how much of the file has been downloaded. When all files have been successfully downloaded, GAPPL asks the user if he or she would like to install the application. If so, it looks for an install file and executes it.

The application transfer process used by GAPPL is further illustrated in the data flow diagram shown in Appendix B. To learn more about using GAPPL, read the GAPPL user's manual. (Harmet and Japel, August 1992).

6 Summary and Recommendations

Summary

The computer application distribution system GAPPL, developed for the USACE Directorate of Civil Works, is an effective, user-friendly tool designed to give software users access to new applications and updates more quickly and cost-effectively than distribution on diskette by mail. While system operation is virtually transparent for the benefit of novice users, GAPPL provides the more experienced user with capabilities for modifying certain settings and defaults to tailor the transfer process more closely to the individual's needs. A return-on-investment for GAPPL has not been calculated, but a previous return-on-investment case study of the electronically distributed Automated Budgeting System (ABS) indicates that the costs avoided by electronically distributing a single application to 40 Districts may amount to almost $7200 per year. Therefore, cost avoidance by using GAPPL could save the Corps tens of thousands of dollars per year depending on how many applications were distributed and how many users each application must be distributed to.

As noted in Chapter 2, one of the main goals of GAPPL was to establish an open system that could easily be ported from one computing environment to another. An important element in GAPPL's portability was the selection of a communication protocol. Although Vistacom was the Corps standard communications package and was considered carefully as a development tool, the Kermit protocol ultimately was chosen because it is more universally available, more flexible, and may be used in not-for-profit software packages obligation to pay royalty fees. Kermit transfers data slower than some other protocols, but since it has a consistent communication interface across three different operating systems, the advantage of portability outweighed the relative transmission speed deficit.

Improvement of data transfer rates has been a continuous goal throughout GAPPL development because faster file transfer means greater distribution cost avoidance. Transfer rates can be affected by the size of the file and the baud rate. The integration of archiving utilities decreases file size by as much as 50 percent, but even a compressed application module may measure in the megabytes and take a considerable amount of time to transfer.

Recommendations

Two general aspects of GAPPL could be improved with additional work: transfer rates and user interface.

Transfer rates could be improved by integrating patch utilities into GAPPL. A patch is a file that encodes precisely that information that changes between two versions of an application. A patch is built using a special build executable and is applied using a special patching executable. The size of a patch file can be as little as a few hundred bytes to several thousand bytes, depending on the number of changes made in the newer version. Such a decrease in update module size would represent a great boost in distribution efficiency. Another way of improving transfer rates is to increase baud rates. Over the lifetime of this work modem speeds have increased dramatically. At the beginning of this research the transfer rate was 1200 bits per second, and reliability was low. At the time of this writing, some sites have trunk lines running at 19,200 bits per second. These are all transfer rates for asynchronous communication. If synchronous communication can be implemented, transfer rates would jump to 57,600 bits per second. Synchronous communications will be an option when the CEAP environment is completed.

The user interface also could be improved. GAPPL currently uses simple menus and forms that are easy to use but not consistent with other Corps applications. The Automated Budget System (ABS) uses menus and forms that operate slightly differently from those in GAPPL. The Microsoft Windows™ user interface is designed to promote similarity between applications. Although ABS is a database application and GAPPL is a communication application, user efficiency could be further enhanced if the two systems used the screens and pulldown menus common to all Windows™ applications.

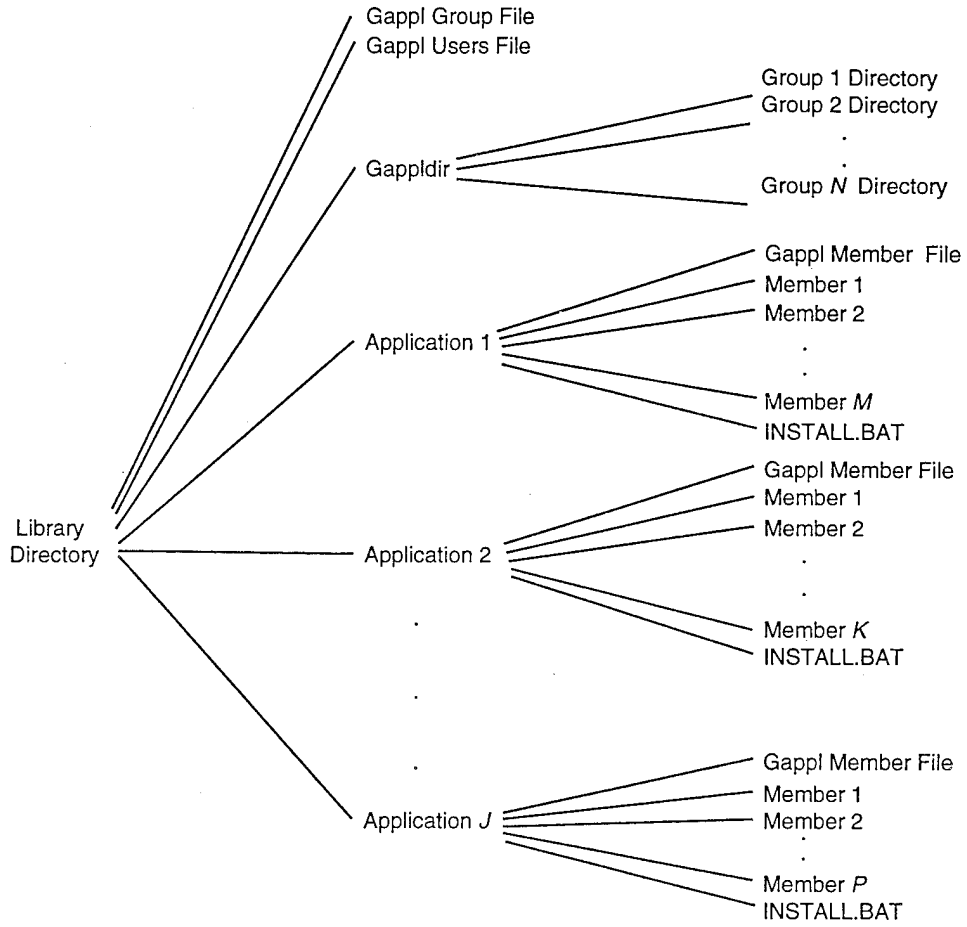
References

"Automated Budget System for Divil Works Districts," *Return on Investment Case Study*, vol R15, no. 1 (U.S. Army Construction Engineering Research Laboratories [USACERL], May 1992).

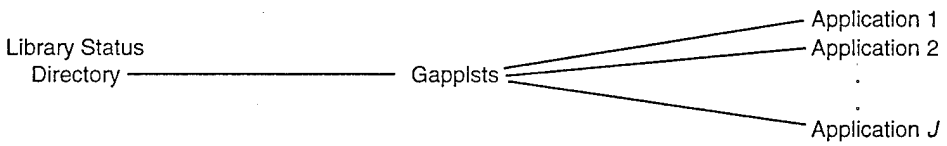
Harmet, Laura L. and Edward J. Japel, *GAPPL Application Distribution Facility User's Guide: Version 2.0 for the Corps of Engineers Automation Plan*, TR FF-92/03/ADA256860 (USACERL, August 1992).

Appendix A: GAPPL Mainframe and PC File Structures

Path Structure Required of Remote Computer



Status Files on Remote Computer



Note:

$J <= 30$, $N > 0$, $M > 0$, $K > 0$, $P > 0$

GAPPL Users File Structure

Record Format			
Field	Type	Description	Length
Login ID	string	Login ID to identify the user	limited to 25 char.
Gappl group	string	The group that the user belongs in	8 characters

Use: * Identifies the user's group, which specifies the directory he will see.

GAPPL Directory File Structure

Record Format			
Field	Type	Description	Length
Description	string	Description of the application	40 characters
Version	"	Latest version of the application	5 "
Blank Space	-	This is a blank field needed to separate version -date	1 "
Date	"	Release date	10 "
Name	"	Name of the application. Used for PC directory	8 "
Directory	"	Directory on the mainframe where the application is.	32 "
Initial Version	"	Last distributed version on disk	5 "
POC Name	"	Point of Contact name	20 "
POC Phone	"	Point of Contact phone number	20 "

Use: *maintains information for the various applications
*source of the menu displayed on the user's PC

GAPPL Member File Structure

Record Format			
Field	Type	Description	Length
Member Name	string	Mainframe filename of the member file.	8 characters
Version	"	Latest version of the member	5 "
Zipped Size	"	Size of the zipped member in bytes	8 "
Unzipped Size	"	Size of unzipped member in bytes	8 "
DOS Filename	"	PC filename without the extension	8 "
DOS Extension	"	PC extension with or without the '.'	4 "

Use: *maintains information for the members that comprise an application

Note: The text files, such as INSTALL.BAT, do not have a version number in the GAPPLMBR file. This is so they are brought down every time the user gets an update.

GAPPL Status File Structure

Record Format			
Field	Type	Description	Length
Version	string	Latest version the user has	5 characters
Date	string	Date the user downloaded the application	10 "
User Name	string	The person who downloaded the application	20 "
Route	string	Path	15 "
Mode	string	Network or modem	15 "
Local_baud	string	Network/modem baud rate	6 "
Remote_baud	string	Mainframe baud rate	6 "
Login id	string	Login id	20 "
Appname	string	Name of application	8 "
Group	string	Group	8 "

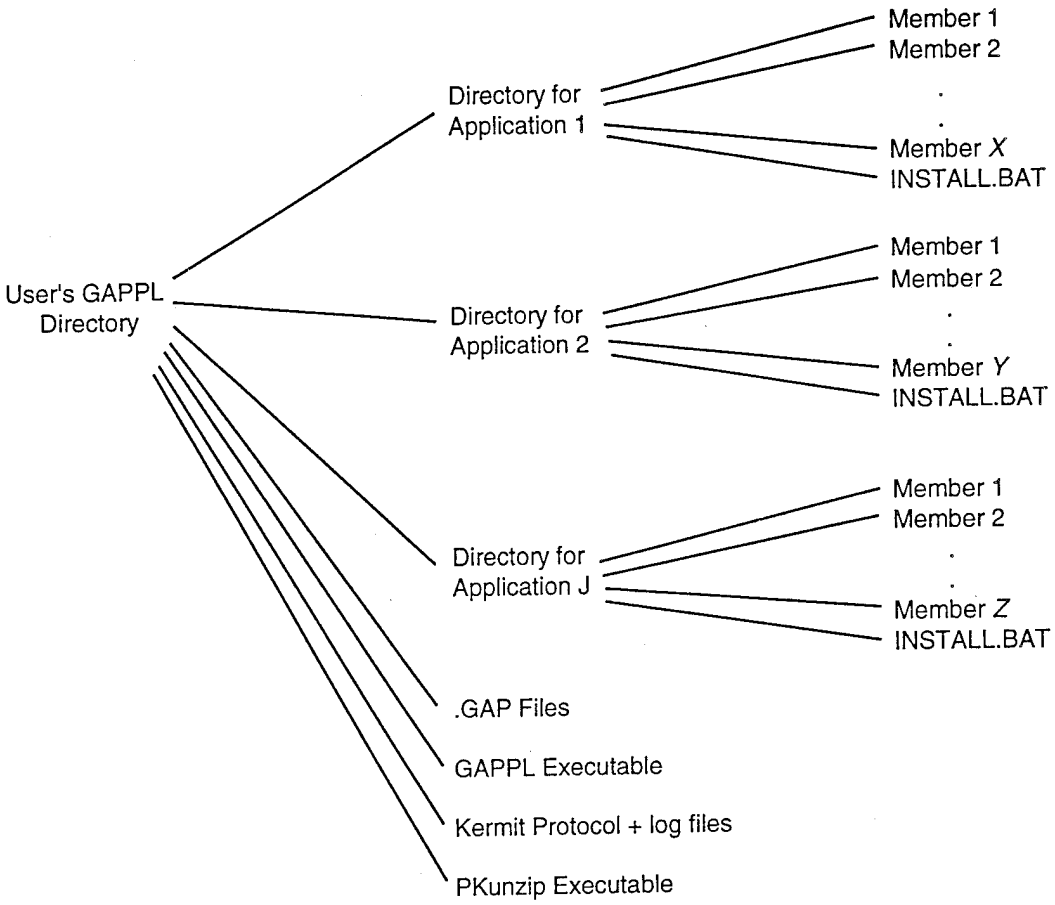
Use: *records the latest version of each application a user has.
Note: There is a GAPPLSTS file under each user's logon for each application that they have downloaded. Each file consists of one record described in the above format.

GAPPL Group File Structure

Record Format			
Field	Type	Description	Length
GAPPL group	string	Valid group	8 characters

Use: *maintains list of valid GAPPL groups

Path Structure on User's PC



Note:

J <= 30

The number of application directories depends on how many applications the user has downloaded through GAPPL. The number of members for any application depends on whether the user has downloaded just the updates for an application or the entire application. In any case, the INSTALL.BAT file is brought down for every download.

Appendix B: GAPPL Data Flow Diagram

This appendix shows a series of diagrams that represent how data flow through the GAPPL process. Figure B1 charts the overall data flow. The circle, or "bubble" in Figure B1 is "exploded" in Figures B2–B5. Figure B2 shows processes within bubble no. 1 in Figure B1, Figure B3 shows processes within bubble no. 1.1 in Figure B2, and so on.

Project Name:	Application Distribution (GAPPL)
Project Path:	d:\gapdsgn\
Chart File:	dfd00001.dfd
Chart Name:	GAPPL data flow
Created On:	Sep-16-1993
Created By:	Laura Harmet
Modified On:	Oct-22-1993
Modified By:	Laura Harmet

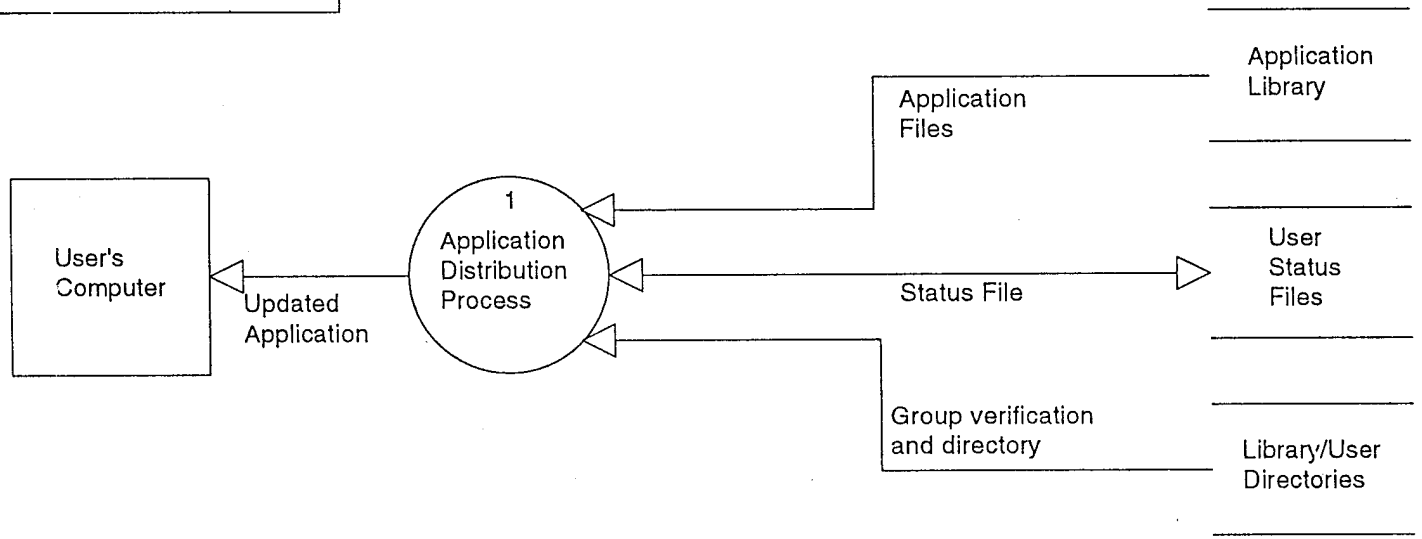


Figure B1. Overall process showing bubble no. 1.

Project Name:	Application Distribution (GAPPL)
Project Path:	d:\ecwin\gapdsgn\
Chart File:	dfd00002.dfd
Chart Name:	Application Distribution process
Created On:	Sep-20-1993
Created By:	Laura Harmet
Modified On:	Sep-20-1993
Modified By:	Laura Harmet

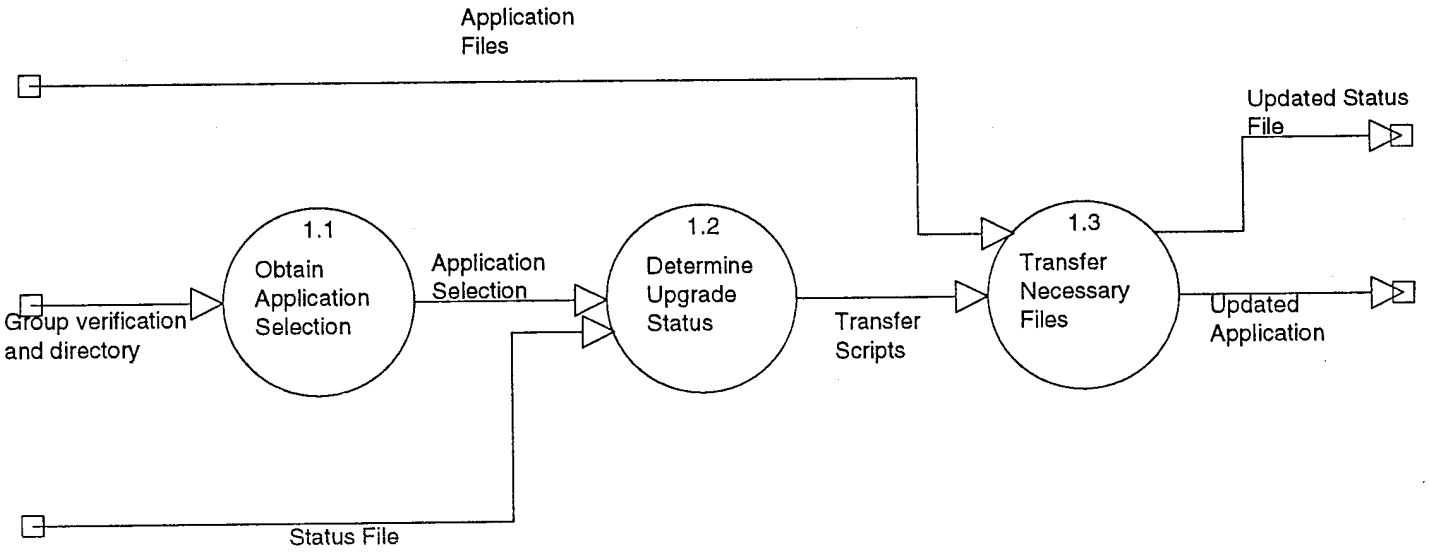


Figure B2. Process exploded from within bubble no. 1, showing bubbles no. 1.1, 1.2, and 1.3.

Project Name:	Application Distribution (GAPPL)
Project Path:	d:\gapdsgn\
Chart File:	dfd00004.dfd
Chart Name:	Obtain Application Selection
Created On:	Sep-20-1993
Created By:	Laura Harmet
Modified On:	Sep-23-1993
Modified By:	Laura Harmet

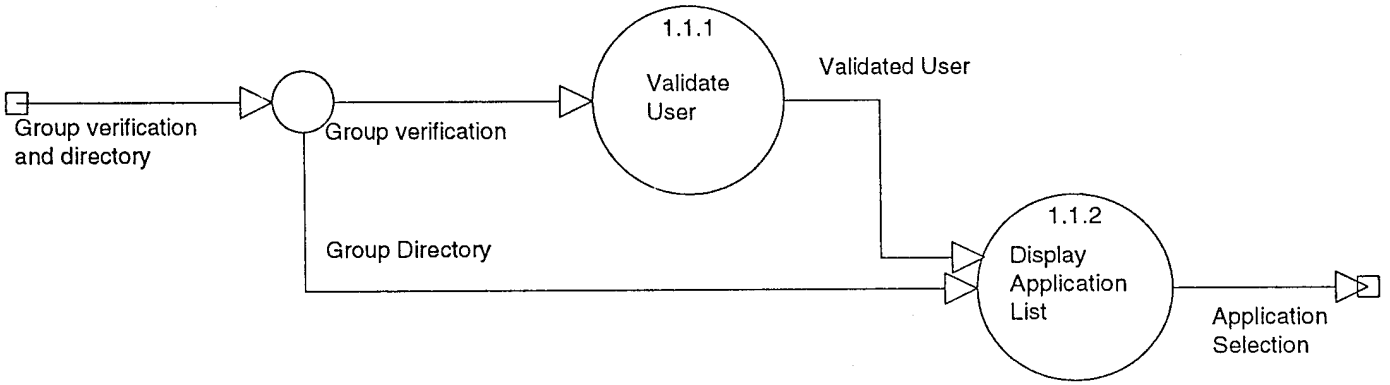


Figure B3. Processes exploded from within bubble no. 1.1.

Project Name:	Application Distribution (GAPPL)
Project Path:	d:\gapdsgn\
Chart File:	dfd00005.dfd
Chart Name:	Determine Upgrade Status
Created On:	Sep-20-1993
Created By:	Laura Harmet
Modified On:	Sep-21-1993
Modified By:	Laura Harmet

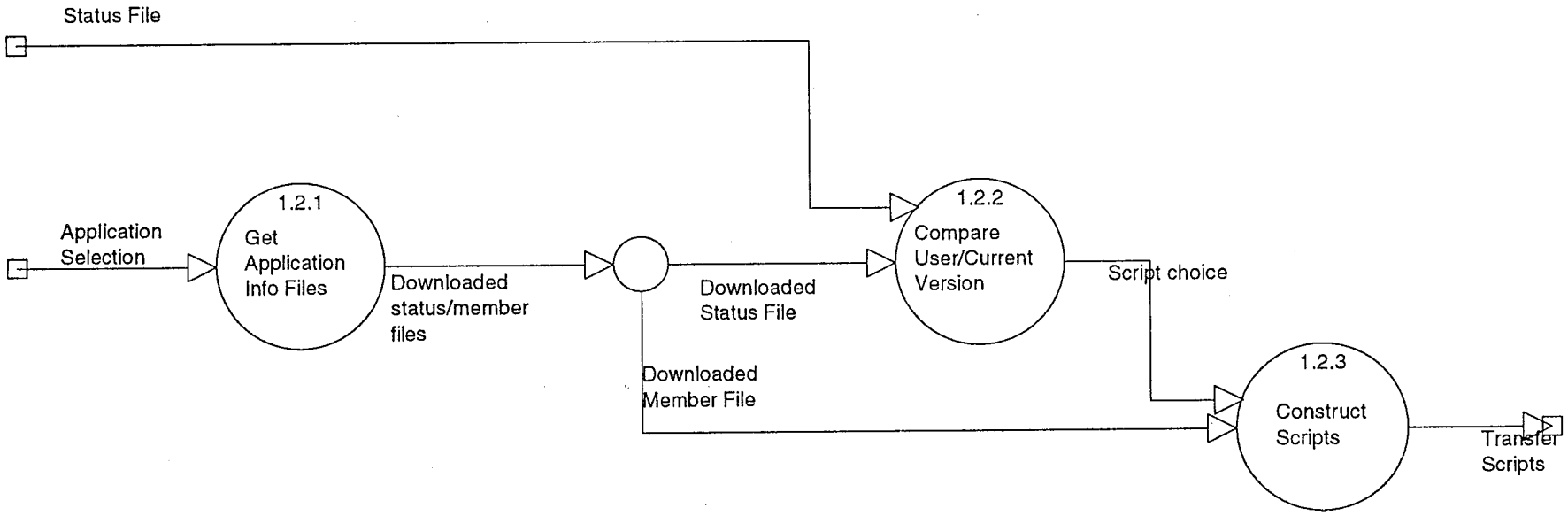


Figure B4. Processes exploded from within bubble no. 1.2.

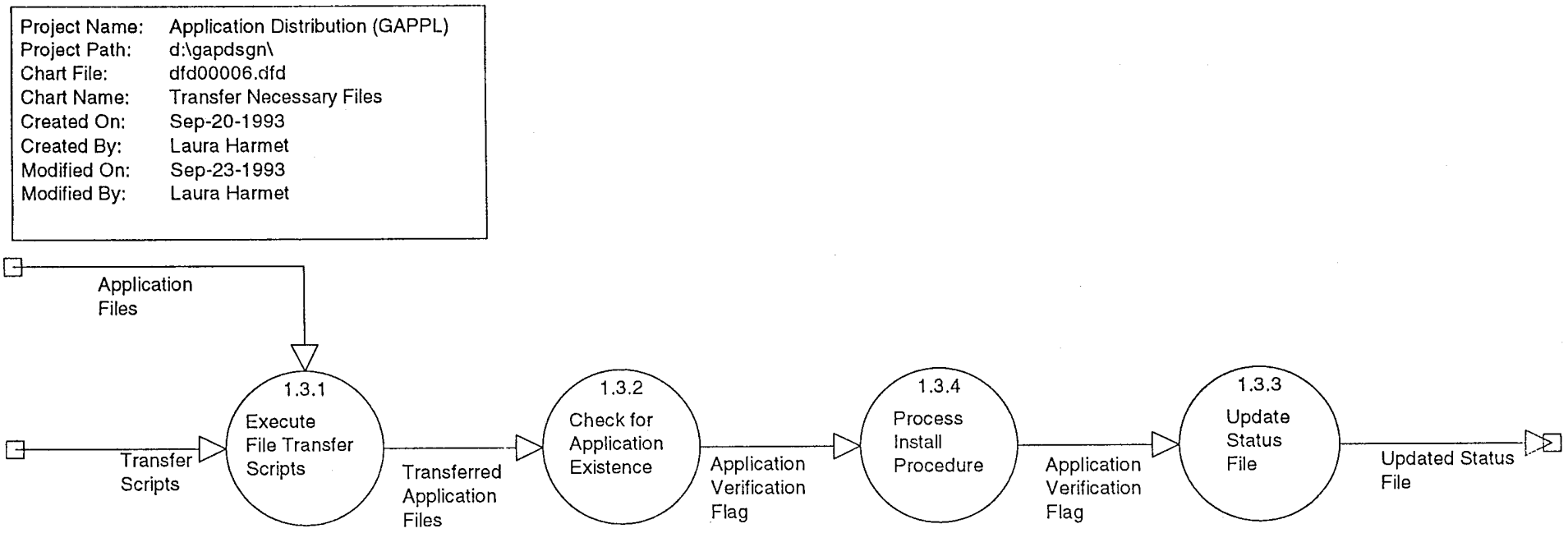


Figure B5. Processes exploded from within bubble no. 1.3.

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CECC-R
ATTN: CERD-L
ATTN: CERD-OM-B (2)

US Army Engr District
ATTN: Library (40)

US Army Engr Division
ATTN: Library (12)

CEWES 39180
ATTN: CEWESHE-E

Defense Tech Info Center 22304
ATTN: DTIC-FAB (2)

63
4/95