Journal of Infonnation Systems Education ***WINTER 1998***

Software Submission using Client/Server Architecture in a Windows Network Environment

Luann Stemler David Doss

Bill Swafford

Applied Computer Science Department Illinois State University

Normal, Illinois 61790-5150 (309) 438-3228

Istem le@rs6000.cmp.ilstu.edu dldoss. @rs6000.cmp. ilstu.edu beswaff@rs6000. cmp. i Istu .edu

ABSTRACT

This paper describes a system for program submission using the clienUserver model. A Visual Basic front-end is combined with a C client to provide access to a C serve!:

INTRODUCTION In the fall of 1994, the curriculum of our depanment changed from a *PU1* mainframe based programming environment to one using C on miCroComputer development systems. In addition to the changes in language and platform, a weekly tWo hour closed laboratory component was added to the introductory program- ming course. Previously in programming courses, faculty had been able to easily control mainframe test data sets so that grading student programs could be carried oUt by carefully checking pro- gram listings together with program output. With the implemen- tation of the new curriculum, student programming assignments had to be submitted to faculty in the form of both a program list- ing and a floppy diskette containing the program source code.

The introductory programming CoUrse is organized into forty stu- dents per lecture section with tWo twenty-student laboratories per lec- ture. There are typically six programming assignments per lecture sec- tion. After the first two weeks of each semester, students are required to write from one to three small programs per week as a part of the closed laboratory experience. These activities produce approximately 30 programs per student, per semester, for a grand total of 1200 pro- grams. Of COUrse, most of these programs are small and easy to grade, but the overhead of grading either twenty laboratory or forty lecture programs that reside on separate diskettes is considerable. A non-triv- ial amount of time is consumed in inserting and removing diskettes, as well as in accessing the program files that reside on the diskettes.

In addition to the time required in manipulating so many diskettes over the course of the semester, the instructor must invest additional time in order to maintain the connection between a stu- dents program listing and the proper diskette containing the pro- gram source code. AlthoUgh it is true that some of the laboratory programming activities are Short enough that they could be ade- quately graded withoUt a program listing, none of the lecture pro-

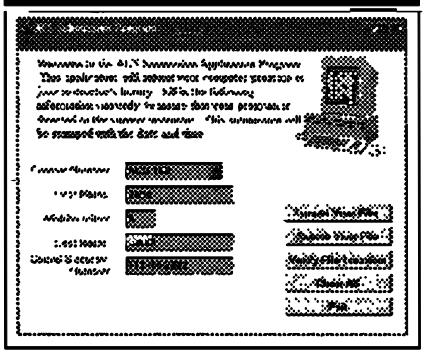
gramming assignments fall into this category **In** order to keep their program listing and diskette together, the students were required to submit their assignment materials in a folder or envelope. While this solution maintains the required connection between listing and source, it does add even more overhead to the grading process.

The process described above proved to be workable, but was so tedious that a better solution was sought. Since the faculty had little experience with PC based networks, the obvious solution did not present itself immediately: However, by the second year of the new curriculum, a program had been developed and placed on all departmental computers that would allow students to submit their source code directly from their diskette in the a: drive to a network server. The server was organized by course and section, so that programs for a given class were all collected together.

Visual Basic was chosen as the tool with which to develop the Submit program because of the **GUI** front end it provides. The facul- ty member who actually wrote the program already had expenise in using this development environment, so there was no initial learning curve. However, new facilities incorporated into the tool by Microsoft have led to improved usability characteristics of the Submit program.

DESIGN OF GUIINTERFACE The first problem in the program design was to determine a method- ology for saving the stu- dent's files on the server. It was decided that each course would have a directory on the file server with each section

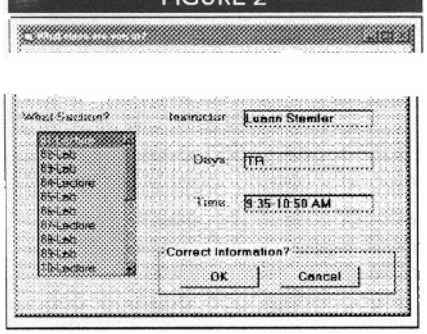
FIGURE 1



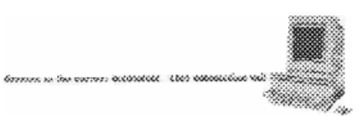
of the course having a subdirectory within the course number's directory, for example ACS 168, Section 01 would have its own directory path of T:\ACS168\01. The student must click on the arrow in the combo box adjacent to the course number and select the course title. This action brings up a new visual display screen. The program reads a database stored on the server for the course section information. The section numbers are displayed in a list box on the left side of the form, as shown in Figure 2. The stu- dent must click on their section number to display the correct information regarding instructor and time of the course. This information needed to be displayed because many students could not remember what section they were in, but could usually remember the instructor's name and time of the course. A data- base was chosen to store the course information due to the quan- tity of maintenance needed at the beginning of each semester and the ease of manipulating the data. Visual Basic Professional edi- tion has a built-in database that was used in this project for con- venience. When the selection of the course number and section number is complete, the path to the correct server subdirectory is determined. However, the path is not displayed for the student. Each instructor has access to the subdirectory of their corre- sponding section. The original design was concerned with ser- vicing only the single multi-section course, but the solution has proven to be flexible enough to be used by any number of cours- es. Currently there are 10 different courses using the program with each course having multiple sections.

The next methodology to be determined was
, ',.'"!"'~"'"'::",,,~:r~,=;':c""=t;.:1i\v~,;;!1";"!:;". I how to store all student files enrolled in the same section in a subdi- rectory with unique names. The convention that was previously used
, on the mainframe for student program names was adopted. Each stu-
dent's file is named using the first letter of their first, middle, and last name along with the last four digits of their social security number, plus a single digit project number. The project number was added to give students the ability to save multiple programs in an instructor's directory. For example the file name for the information collected in Figure 3 would be]KS22222.C.
The student must use the ***Locate Your File*** button (Figure 3) ~ to select the file they
.wish to submit. A com- mon dialog box will appear in which student identifies the file to be submitted. The student then enters the project number and clicks on the ***Submit Your File*** button (Figure 3). The date and time of the server is used for the

FIGURE 2



't:':..:~;~~::-:~ ':':';=~.~~~ ~;~~"::~:';.~;7. ~~~~"' "



,,.,.,.,.,.;(0

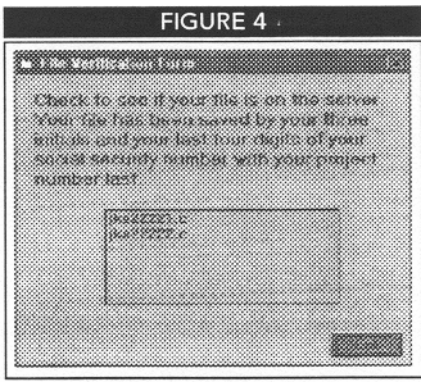
"'"",,".:". ,..',,"", ",..",..,. *:w."y~. ,.v~* Ooc.""
'''''''' "".''-
,.,":-, :~,

0$0:, '.' ~*~~*i:ji:iii'* ~fiiiliT$'j;-',;r~ ~
~ :--"';_.-,-,:,'' -" -,.~~:::::::::::: wm

program when a student file is copied to their instructor's directo- ry. A program can be submitted any number of times, and only one copy is saved, that being the most recent submission.

Most students were uncomfonable about relying on blind faith that the file was stored in the correct instructor's directory An addition button was included, *Verify File Location,* that allowed students to veri- fy their file was copied to the correct location. The directory location is again invisible to the stu- dent. The student will only see the filenames that have their own initials and last four digits of their social security number, as shown in Figure 4. Files are copied with the original file eXtension because of the various courses that are using this Submit program.

FIGURE 4



IMPROVED VERSION The first version of the Visual Basic Submit program was used successfully for a full academic year and it greatly improved the pro- gram handling process for the instructors. However, two observed potential problems led to the consideration of further changes. The first possible problem is related to the fact that student programs are collected to hidden subdirectories on a network server to which the students have write and create only access. Using the *dir* command with the *ah* switches, hidden file and directory names can be listed. Once the name of a hidden directory is known, the *dir* command can be used to list the files in the hidden directory Then it would be a simple matter for someone with write and create access to sys- tematically destroy all the student files. The existence of virus pro- grams strongly suggests that someone might do just that.

The second possible problem is related to the use of the date/time stamp associated with submitted programs. This stamp is typically used to determine if a program was submitted by the assigned deadline, and if not, how many penalty points to assess. In the initial system, this date/time stamp is generated by using the date/time values of the computer from which the student submits his/her program. It is a very simple matter to reset these values backward so that a late program appears to have met the deadline requirements. This is not as malignant an activity as destroying other student programs, but is much more likely to happen.

The solution to both of these problems is to conven the existing Submit program into a client/server application. The server will save programs on a network drive to which the students have only the access provided by the server. This access will consist of saving a program submitted by a student to the appropriate subdirectory, and providing a student with a directory listing of programs that he/she has previously submitted. The first problem is defeated because stu- dents cannot use the *dir* command on the server drive since they have no direct access to it. The date/time problem no longer exists because the server uses the date/time values of the computer on which it is running, and not those of the client computer.

Converting to the client/server model allows the use of the existing Visual Basic Submit program to continue to provide the GUl interface to the students. The necessary network protocol sequence to access the services of the server is added by using a Dynamic Link Library (DLL) that is called by the Visual Basic module once all student information is collected. The **DLL** actu- ally read the file from the student's diskette using information col- lected by the Visual Basic program, and sends it to the server.

An added benefit of the client/server version of the Submit pro- gram is that students are now able to submit their programs remote- ly using their own computers through a modem to access the uni- versity network. This is seen as a great convenience by students.

CONCLUSION The Visual Basic module is written in version 4.0, so it can be run on both Win NT/Win 95 (32-bit) and Win 3.x (16-bit) sys- tems. The server is written in Visual C++, version 4.0, and can run on Win NT and Win 95 systems. Because there are still a large number of Win 3.x systems in use, there are actually two versions of the client DLL module, one in Visual C++, version 4.0, for 32- bit systems, and one in Visual C++, version 1.52, for 16-bit sys- terns. This software will be made available to interested parties from our depanmental Web site.

It is too early to determine how gracefully the current version of the Submit software will handle the necessary volume of stu- dent program submissions. But the multitasking capabilities of the 32-bit operating systems vinually guarantee that a version can be developed that will handle the required load.

BIBLIOGRAPHY Cooper, Alan (1995). ***About Face The Essentials*** **of** ***User Interface Design.*** **Foster City: IDG Books Worldwide, Inc.**

Hall, F., Towfiq, M., Arnold, G., Treadwell, D., & Sanders, H. (1993, January 20). *Windows Sockets: An Open Interface for Network Programming under Microsoft Windows, Version 1.1. Available: ftp:llftp.microdyne.com/pub/winsock.*

Microsoft (1995). *Microsoft Visual Basic Programmer's Guide* Redmond: Microsoft Press.

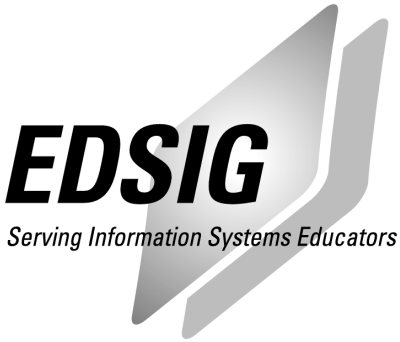
Microsoft (1995). *Notes for Developing DLLs for use with Microsoft Visual Basic Version* 4.00 (VB4DLL. TXT). Available: *ftp:llftp.* microsoft.com.

Microsoft Corporation (1995). *Visual* C++ *Microsoft Foundation Class Library.* Redmond: Microsoft Press.

Scott, C., Shannon, B., Font, F., & Hatfield, B. (1995). *Visual Basic* 4 *Unleashed.* Indianapolis: Sams Publishing.

Webb, J., McKelvy, M., Martinsen, R., Maxwell, T., & Regelski M. (1995). *Special Edition Using Visual Basic* 4, Indianapolis: Que.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096