The Realities of Client/Server Development and Implementation

Presented by

Mary Ann Carr
Director of Administrative Computing
Carnegie Mellon University
Pittsburgh, PA 15213

and

Alan Hartwig
Manager
Deloitte & Touche Consulting Group
Pittsburgh, PA 15222

for

The CAUSE 1995 National Conference
November 28 - December 1, 1995
New Orleans, LA

ABSTRACT

Carnegie Mellon University has made the initial transition to client/server technology. Most previous systems were developed using relational database, fourth generation language tools, C and Unix platforms with a character based user interface. The client/server model involves the use of the client desktop, a Netware server and a database server.

The paradigm shift to client/server should not be taken lightly. There are a number of issues that had to be addressed. These included which client/server tool(s) to use, which project should be the first client/server project, project planning and estimating, development standards and training for both development staff and end users. We also had to worry about end user desktop computing and network issues.

This paper will address these issues and reflect on the lessons learned throughout this process.
Introduction

Carnegie Mellon University is a private research university located in Pittsburgh, Pennsylvania. Carnegie Mellon enrolls approximately 7,000 students and has an annual budget of more than 300 million dollars.

The administrative computing environment at Carnegie Mellon is a mixed bag of hardware, operating systems and applications. These systems are supported by the Administrative Computing and Information Services (ACIS) department. ACIS consists of twenty-six professional and support staff. There are three primary hardware platforms supporting the administrative systems. These platforms are shown in the table below.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operating System</th>
<th>Primary Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vax Cluster (6610,6430)</td>
<td>VAX VMS</td>
<td>Payroll, Alumni, General Ledger</td>
</tr>
<tr>
<td>Sequent S2000/490</td>
<td>Dynix PTX</td>
<td>Student Systems</td>
</tr>
<tr>
<td>Data General Aviion 9500</td>
<td>DG/UX</td>
<td>Financial Systems</td>
</tr>
</tbody>
</table>

The applications are a combination of commercial and Carnegie Mellon developed systems. The table below summarizes the key applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Vendor/In-house</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Systems</td>
<td>In-house</td>
</tr>
<tr>
<td>Financial Aid</td>
<td>In-house</td>
</tr>
<tr>
<td>Human Resources</td>
<td>In-house</td>
</tr>
<tr>
<td>Payroll</td>
<td>Cyborg Systems</td>
</tr>
<tr>
<td>Alumni/Development</td>
<td>Business Systems Resources</td>
</tr>
<tr>
<td>Financial Systems</td>
<td>In-house</td>
</tr>
</tbody>
</table>

The development tools used by Administrative Computing and Information Services have been primarily the Ingres fourth generation language (Application By Forms) and C with the Ingres relational database management system.

The primary desktop computer for users of administrative applications is the Apple Macintosh. The desktop environment is 80% Macintosh, 15% Windows PCs and 5% Unix workstations or terminals. The percentage of Windows machines has been growing in recent years.

The campus network is an ethernet backbone. The two primary network protocols being used are AppleTalk and Ethernet.

There are several Novell Netware servers being used for administrative applications. Most of these have been implemented in the last couple of years.

Client/Server Tool Choices

Administrative Computing and Information Services formed a team to evaluate alternative development environments. This team developed a set of criteria for a third party development tool. Several of these criteria were determined to be “drop dead” criteria. In other words, if a vendor did not support these criteria, their product would not be considered.

The drop dead criteria included supporting the Macintosh platform for both development and deployment, supporting the hardware and operating systems already in use, accessing a variety of relational databases, the ability to access more than one database simultaneously...
and the ability to provide a character based interface without creating a separate set of code.

Indiana University had already gone through a very similar and very thorough evaluation of client/server development tools and shared the results of their work with Carnegie Mellon. This additional information provided some valuable insight into the strengths and weaknesses of a number of client/server development tools.

One of the primary goals in this process was to become independent of any particular database vendor. We wanted to be able to change database management systems without having to re-write applications. Another key goal was to be able to bring information stored in a variety of databases together on one screen. The tool selected needed to be able to access a variety of databases simultaneously. The final goal was to be able to offer a graphical user interface (GUI) front end.

Support for the Macintosh as a development and run-time platform became one of the most difficult criteria along with the ability to support a character based user interface with the same code. There were very few products available that met these criteria.

Uniface, now owned by Compuware, was identified by the team as the leading candidate for a client/server development tool. Uniface met these objectives by varying degrees of proficiency, but better than the competition overall. A Uniface evaluation (proof of concept) was performed by Administrative Computing and Information Services with support from Uniface. An evaluation plan was developed and Carnegie Mellon sent two developers to a week long Uniface training seminar.

The evaluation consisted of developing a simple Uniface application in the Carnegie Mellon University environment. Once the application was developed, it was ported to a variety of different platforms or databases to test these functions.

Uniface was chosen as the preferred client/server development tool upon the completion of the evaluation.

Project Choice

At the same time that Uniface was acquired, a relatively large module of the Human Resources Information System (HRIS) was under development. The majority of the detailed specifications had been completed by the project team. The project team was composed of key campus users, representatives from central administrative offices (payroll, benefits, student employment) and three people from Administrative Computing and Information Services. This project was replacing a paper based process for adding and updating employee and employment information. The project was very visible and had substantial end user involvement. There was concern about developing this important new application in a character based, database specific tool. A discussion was held with the project team regarding the pluses and minuses of switching to Uniface at this point in the project. The primary minus was that it would push back the schedule an estimated four to six months while Administrative Computing staff members became proficient with Uniface and built the infrastructure to support it. The primary plus was that the result would be a client/server application with a native graphical user interface that would be database independent. After weighing the pluses and minuses, the project team agreed to go ahead with the Uniface development. Thus, this application became the first significant Uniface (client/server) application developed at Carnegie Mellon University. This application is known as DRIVE (Distributed, Real-time, Interactive, Validated, Entry).
Major Issues

There were a number of major issues that needed to be addressed once the decision had been made to switch gears and use Uniface as the development environment for the DRIVE project. These issues included project planning and estimating, developer and end user training and infrastructure issues such as desktop computers and network connectivity.

Project Planning and Estimating

Preparing accurate project estimates and keeping a project plan on schedule can be a difficult task when you are very familiar with the tools being used. It is a much more difficult task when working with brand new technology. Because we had no experience with Uniface, it was difficult to estimate how long it would take to develop the application or even small pieces of the application.

There were two major facets to the project planning. The first facet was the planning for the introduction of Uniface. This included training ACIS staff, development of standards and environment configuration. This was the first major application for ACIS using Netware servers as part of the application delivery. Standards for access and security for the Netware servers to both Windows PCs and Macintoshes had to be developed.

In general, we knew that the learning curve for Uniface was rather steep and that it would take some time for the developers to become proficient with the Uniface tools. In addition to formal vendor training, Carnegie Mellon co-organized a local Uniface users group to share in on-site training and provide another layer of support. We also had access to Uniface technical support and user bulletin boards for additional information. The Uniface pre-sales force was well aware of the learning curve and provided assistance in our start-up.

The second facet was the planning for DRIVE which was well underway at this point. The schedule needed to be adjusted for the time necessary to introduce Uniface, plus the time needed to determine what changes needed to be made to the existing design.

General screen designs had been done before the switch to Uniface was made. The graphical interface made these screen designs basically obsolete. The screens were re-designed to take advantage of the graphical interface capabilities.

Uniface provided the ability to do rapid prototyping. We planned to use this ability extensively in the re-design work, as well as to make up time on screens that had not been fully specified before the tool switch.

Training and Development Standards

Obviously, with a new tool and environment there were a variety of training issues. We needed to train developers in the Uniface tools, to train systems support staff in Uniface technical and communications issues and to provide training for the end users on both the application and the Uniface environment.

The first developers assigned to the project were sent to a couple of vendor training sessions. The second group of developers were trained by the first group of developers. Some amount of self-study and trial and error was necessary in making the paradigm shift from the Ingres fourth generation language toolset to the Uniface environment. The training model became “on the job training” for the most part.
One of the issues that confronted the project team was the creation of standards. Since this was a new tool, there were no screen design, coding, deployment or documentation standards in place. Several people worked on various pieces of the standards document. The standards were modified as needed as the project progressed. For example, the Main Menu screen design had to be changed at the last minute due to a technical problem which could not be resolved.

There was a great deal of discussion on how to do end user training. The hope was that the graphical user interface would be relatively intuitive and not require a great deal of training. An inquiry only version of the primary screen was developed and released to interested users. Formal training was not provided but a “quick reference” card and basic documentation were available. The project team hoped this would significantly reduce the training requirement at the time of the production release of the system.

**Infrastructure**

Infrastructure turned out to be one of the biggest issues of the project. There were a variety of key infrastructure issues including desktop computers, network connections, deployment strategies, security, database access and remote access.

As mentioned earlier, the predominance of Macintosh computers on the desktop was a key factor in the choice of a client/server tool. Many of the Macintosh computers communicated with the campus network using the AppleTalk protocol. AppleTalk is a relatively slow network connection and the client/server model moves a lot of data over the network. A concerted effort was made to encourage end users to convert their AppleTalk connections to Ethernet. Administrative Computing and Information Services worked closely with the Data Communications department to provide timely and low cost conversions. A large number of users did convert their connections, but many still use AppleTalk.

Many of the Macintoshes are older models and do not have adequate resources to run client/server applications. Many of these can have additional memory installed which would help in some cases. The advent of the DRIVE application at the same time as the release of a sophisticated Excel spreadsheet for use in sponsored research budget development brought the issue of desktop upgrades and replacements into the forefront.

Administrative Computing recommends that departments should budget to replace 25% of their desktop computers every year. This, at first, seemed excessive to many users. The current pace of change in desktop hardware and software is so rapid that a machine over four years old is of limited value and it may be difficult to continue to upgrade it at a reasonable cost. When the desktop computer was being used primarily to do terminal emulation this was much less of an issue.

Performance in a client/server environment is much more difficult to monitor and improve. The problem can be the client machine, the network connection to the application server, the connection to the database server, the database system or a combination of all of the above. It is also difficult to simulate a large system load during testing due to the number of variables, especially network traffic, in a complex network environment such as Carnegie Mellon University and even more so in the summer months when student network traffic is at its minimum.

The client/server model that was used involved client software running on the desktop, application software on a Novell Netware server and the relational database management system on a database server. Originally, the database server was on a VAX/VMS machine, but the database was moved to a Data General Unix machine which had a lighter load in order to improve performance.

A key deployment issue was what to put on the local desktop machine and what to put on the Netware server. Application files could be stored in either location. The advantages to storing the application files on the server included not having to worry about whether
the user had the most current version running on their machine. On the other side, there was a performance advantage to having the application files on the local machine rather than accessing them over the network. The initial deployment strategy was to put the application files on the server due to the frequency of fixes and enhancements which would be taking place in the early production release of the system. This would make version control easier. A future version of Uniface promises to provide version control when the files are stored on the desktop machine.

Security was another key concern. Security exists at the file server level, database server level, application level and database level. Security at the Novell Netware server level was a minimal issue. No data or source code was stored on this server. In order to reduce the systems management load a single account and password was created on the Netware server for all users of the application. The secure logon occurred at the database server level.

The database can only be accessed by users who are defined as having access to that database. If the user has access to the database, they are given permission to access all data and the application controls which data they have access to. This again was done to minimize the amount of system maintenance work. There are approximately 200 defined users for the application. The user must have an account on the system that is serving as the database server. The user logs into this account at the beginning of running the application. This account connects to a restricted shell that prevents interactive logins.

The application has a fairly sophisticated access and approval module. The access model is basically hierarchical. Once the top level, executive managers, were defined to the system they defined the users who would have access to their data. Each succeeding level in the hierarchy performed the same process. Thus, the application restricts the employees that any user has access to. The application requires a separate ‘save’ password before any transactions can be saved or passed on to the next level in the process. This prevents someone from finding an unattended computer running the application and using it to change data.

One of the biggest security concerns turned out to be network security. The client/server model sends a great deal of information over the campus network. That information is not encrypted. Carnegie Mellon continues to explore options for securing the network which will provide the desired level of protection at a reasonable cost.

Ingres is the underlying relational database management system that was used for the DRIVE project. Carnegie Mellon entered into a site license agreement with Ingres in 1986. The site license agreement has ended, but Carnegie Mellon retains a substantial number of Ingres licenses across a variety of hardware and operating system platforms. Thus, there was no additional cost involved for a relational database system for the DRIVE project.

Lessons Learned

There have been a great number of lessons learned throughout this process. Probably the least surprising, but still important is that it will always take longer and it may cost much more than you think.

Client/server Tool

In general, Uniface served and continues to serve us well. This conclusion was initially clouded by serious issues with a Macintosh product which was in its infancy. Uniface provides the graphical user interface, database independence and multiple platform support that were desired. This was born out by the relatively painless move of the database server...
from a DEC VAX system running VMS to a Data General system running DG/UX two weeks before the release date.

The Macintosh version of Uniface caused several problems. At the time of our tool evaluation, the Macintosh version of Uniface was in beta release and while we saw the demo, we did not have hands on experience. By the time DRIVE was selected as the project, Macintosh support was available. What we found was that the Macintosh platform was simply not as robust as the Windows version. Few Uniface sites using Macintoshes meant that we invested hours in debugging and designing work arounds with very little supportive services to rely on. We have subsequently developed an excellent, post-sales relationship with Compuware/Uniface. (As a note, do not expect the same level of expertise and support post-sales and you will not be disappointed.)

We did not plan adequately for the amount of time that would be spent in working with the vendor and getting answers from technical support. We found that it is well worth the effort to identify all the support services you can before you begin work with any new tool in general. Carnegie Mellon made considerable use of user groups and electronic bulletin boards for acquiring information and finding solutions. More often than not, other users had the answer, and the right answer, before technical support even returned the call. Given all the unique combinations of hardware, network, database and application requirements possible now, a nation-wide search is more likely to provide the depth of experience you need rather than one organization, even if it is the vendor.

Another area that did not get complete attention was the quality of documentation and training. Understanding that the possible number and complexity of client/server configurations is huge, we nevertheless expected to find some documentation depth in areas that pertained to our installation. After substantial trial and error with incomplete and inaccurate documentation we discovered that the only way to get the information we needed was through technical support. In many cases the vendor had additional documentation on specific topics and configurations. We quickly learned to be more proactive about seeking this information before wading through the volumes of outdated manuals. We also quickly dismissed vendor training as a cost-effective way to deal with certain topics. Certain training curricula by nature are more broad than others, standards and guidelines for example. As such, these sessions were worthless to us. Pre-read of training materials and references would have made the difference between time well spent and wasted time. While documentation and training were not high on the priority list, more attention to these topics at the start may have uncovered more efficient methods for building product knowledge.

Another issue that we did not adequately plan for was the possible need for a 3GL routine to perform functions that Uniface could not perform effectively. This was made more difficult by the cross platform nature of the application.

Initial Project Choice

Conventional wisdom and advice from the vendor indicated that the first Uniface project should be a relatively small application with limited scope. The timing for the DRIVE project would not allow for the project to be put on hold for an extended period of time while other smaller Uniface applications were developed. There was an overwhelming desire to not release the DRIVE application using the old, character based technology.

Our project plan called for developing test application scenarios in order to test all of the elements of a client/server application. Developing throw-away applications or test application scenarios in order to test all of the elements was a luxury that we did not have in reality. We achieved the same results by staging the release of a more complex production application.
Carnegie Mellon issued an optional, inquiry-only, pre-release of the DRIVE system. DRIVE Inquiry enabled end-users to become accustomed to the look and feel of the new GUI environment while testing network connectivity, desktop performance, software distribution strategies and printing capabilities. We discovered that many users were connected via low performance networks or had substandard desktop machines. During this pilot period, we were able to concentrate on providing solutions to the configuration problems without having to deal with a live, transaction oriented system. We should have more strongly encouraged the users to work with this application. It was a very low risk application and additional usage would have helped us identify and resolve some of the problems earlier in the project.

Unfortunately, not everyone took advantage of the early release, leaving a huge number of end-users with serious problems when we did go live. Users who made use of the early release also tended to be more proactive about upgrading and developed a more positive attitude about the application and the new computing paradigm. For those who did not, trying to sort out the system configuration, database performance, and network traffic issues from appropriate use of a complicated update system was nearly unmanageable. The experience we had in troubleshooting during the inquiry phase paid off as many of solutions and work arounds were already documented and in production, but the ill-effects of unsatisfied end-users lingers.

Project Planning and Estimating

Project Planning is one thing, plan execution is another. With client/server the complexity increases dramatically and the control decreases sharply. All the rules change. It was difficult enough for us just to identify all of the variables, let alone estimate for them. In the end, there were numerous iterations of the project plan. As mentioned above, we reneged on our original plan to release the application in total and opted instead to stage the release.

Rapid prototyping proved to be both a blessing and a curse. Because it was so easy to change screens, at least on the surface, there were many more iterations of key screens than had been planned for. Scope creep became a difficult management challenge. It was also difficult for the documentation and training committees to work with a product that was constantly changing. Furthermore, we found the learning curve prohibitive for rapid application development in our timeframe. If developers are not fluent in the tool, you will not get rapid application development. If the learning curve is steep, development will not be rapid in the short term.

We found the need to enlarge the scope of the development team. Implementing a successful client/server project required the commitment and teamwork of networking, desktop, mainframe systems, and software development, and of course, end-user groups. It was extremely important to verify that each of these groups understood their place in the critical path of a project as soon as possible. As an organization, we were creating a new breed of project team. We believe that the organization as a whole benefited from the development of cross-functional expertise and by the involvement of end-users in the transitioning to client/server.

Training and Development Standards

After years of developing Ingres 4GL applications, the infrastructure of our development environment was very rich: shared libraries, screen templates, all manner of standards, reusable modules, and a development methodology. Experienced developers devised and conducted training sessions defraying training expenses. Very little of this investment translated to the client/server scenario. Two months of product evaluation and testing produced a rough standards document and a few screen templates. Weeks of off-site training were necessary just to develop an elementary understanding of the development tool. The lack of infrastructure contributed to the additional costs and additional time needed to
complete the project. A lack of experience in optimizing for performance cost precious hours in development and troubleshooting.

**End User Computing and Networking**

Carnegie Mellon has an extremely heterogeneous computing environment. Outside of the student computing clusters, there is no centralized control or published standard for networking or desktop computing. Lack of standards created the potential for each and every client to have a completely unique configuration. When problems occurred, what was once a question or two in the host-based scenario and a simple scan of a central log file, became a litany of inquiries, involving database administrators, network gurus, desktop support personnel and system managers.

In our business division alone, 24 different models of Macintosh computers were in use and 6 different PC processors from a wide range of IBM compatible vendors. Printers run the gamut. Fifteen months after initial campus exposure, we are still working on local printing on a case by case basis.

Factoring in support for multiple user interfaces, Macintosh, Windows and character based, added degrees of difficulty not only for system maintenance and troubleshooting, but also for upfront application design. We found that we began excluding GUI objects from screen design in order to accommodate the limitations of the character interface and thus were beginning to chip away at end-user benefits. In the end, Carnegie Mellon choose to abandon the character mode interface platform because of design barriers, additional training burden, and poor user acceptance.

While client/server boasts the benefit of utilizing the ‘power of the desktop’ we found that in many cases ‘power’ was nowhere to be found. Since our existing host-based systems required very little from the connecting ‘terminal’, many of our end-users had minimal desktop configurations. There are hard and soft costs for the discovery, evaluation, and installation of machine upgrades and replacements when multiplied across hundreds of users. At Carnegie Mellon, the decentralized nature of hardware procurement makes it difficult to calculate the total cost.

Network connectivity must also be evaluated as early as possible. ACIS actually developed a program in concert with the Data Communications department to assist end-users in the conversion of their network connections from apple-talk to ethernet at a reduced rate. Budgetary constraints necessitates early communication to the end-user community about upgrade paths and options. Secondarily, it might be useful to establish a fund for grants to those users who cannot afford the necessary upgrades.

Even though we were proactive in discussing the possible need for desktop hardware and network connection upgrades, this cost still came as a surprise to many departments. The cost of the desktop computing upgrades depended on the equipment in use and the nature of the upgrade required. The costs for the network upgrades were relatively low (< $200), but still caused a great deal of concern among a number of customers. We also used this as a forum to discuss planning and budgeting for normal desktop computing upgrades.

On the plus side, network and desktop upgrades, while imperative for acceptable performance of our client/server application, also increased productivity and performance of the other desktop applications that our end-users were already using.

**Other Lessons**
We found that we needed to clarify our goals and to refer to them often to stave off both criticism and frustration. Our primary goals were platform independence and a GUI front-end. This was accomplished.

Our developers became very frustrated. Client Server and Event Driven Programming was a paradigm shift for our 3GL and 4GL developers. At the beginning, middle, and even the end of the learning curve, developers felt that they could have programmed faster in their favorite host based, character mode 4GL. In addition, client/server placed an additional burden to have or acquire expertise in other areas like server configuration, network administration, and cross-platform development. Cross-function training was a critical step down the client/server path as responsibilities and requirements changed. Constant reminders of all ‘wins’ was an important part of winning the development team over to the new paradigm.

It was necessary to teach end-users what client/server entails beyond the ability to use a mouse. Client/server requires certain capabilities of the client machine and the underlying network. Early education and communication was imperative for enabling our users to acquire the minimal and optimal desktop computing configurations.

Conclusion

Administrative Computing and Information Services’ first campus-wide client/server application is in production use. Many lessons have been learned along the way. The strong role of the user community was absolutely essential to the success of this project. Our development team has created a strong foundation for future projects. The second release of DRIVE is in development and should be ready for production in January.