

Northern Illinois University

Client/Server: What Exactly is the Problem?

A Thesis submitted to the

University Honors Program

In Partial Fulfillment of the

Requirements of the Baccalaureate Degree

With Upper Division Honors

Department of OMIS

by

Allan W. Furman Jr.

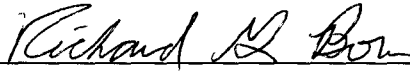
Dekalb, Illinois

May 1996

Student name: Allan W. Furman Jr.



Approved by: Dr. Richard Born



Department of: Operations Management & Information Systems

Date: December 8, 1995

Author: Allan W. Furman Jr.

Thesis Title: Client/Server: What Exactly is the Problem?

Advisor: Dr. Richard Born

Dept: OMIS

Discipline: Information Systems

Year: 1995

Page Length: 14

Bibliography: YES

Illustrated: YES

Published: Pending

ABSTRACT:

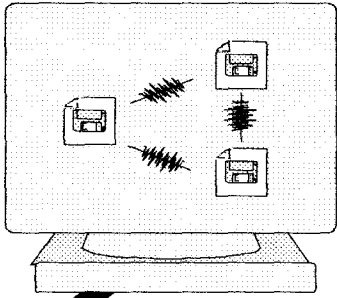
Various organizations have made a commitment to migrate their systems to a new client/server architecture. Due to it's diverse nature, many problems have arisen inherent to the way the system works. For instance, the use of different tools and platforms to accomplish various tasks is required. Getting these tools to work well together in disparate situations is a key issue. How can a company determine which tools, platforms, and systems changes are desired? One way to do this is to communicate, share the information concerning the problems one encounters with others who find themselves in the same boat.

Therefore, a survey was distributed focusing within six facets of client/server application development: performance, capacity, scalability, portability, connectivity, and functionality. The objective: to discover various common links and/or connections that contribute to successful client/server application development. The result: information was gathered from the people who are out there right now producing it.

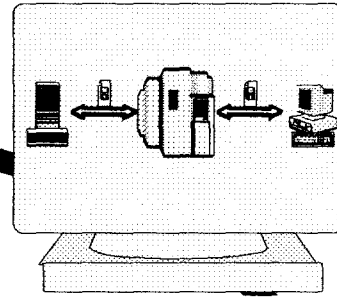
The four page survey was disseminated to over 500 information systems professionals throughout the domestic United States. Participants were selected by the respondent's title(manager or director of MIS or Systems & Programming), the

corporation's listed systems, and were obtained from the Spring 1995 Computing IS Managers Guide. The design of the survey was critical in obtaining specific, relevant information. Utilizing the inverted funnel sequencing style, the task of getting a given respondent to open up was accomplished. The respondent contributed the answers as opposed to the survey suggesting them.

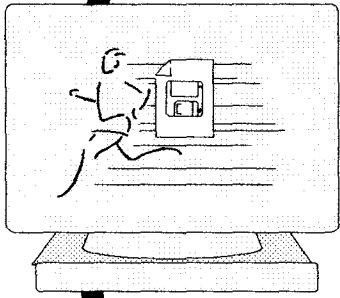
15% of the surveys(74 of 500) were returned; 40 of which the respondent indicated no background in client/server and 34 of which were completed in full. Through analysis of these surveys, I have gained a greater appreciation for exactly how diverse client/server really is. I was able to discover some common threads in each area. This paper is an excellent starting point to further analyze client/server problem areas.



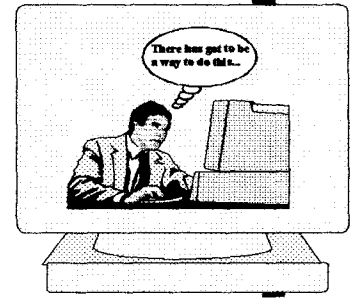
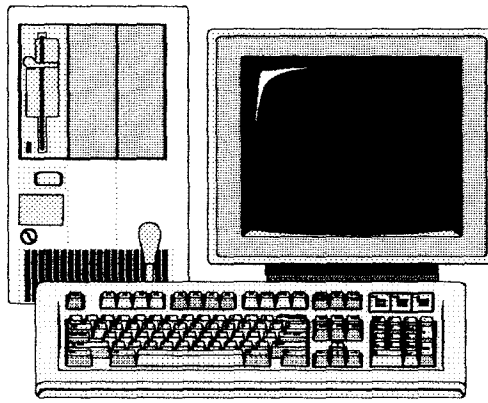
CONNECTIVITY



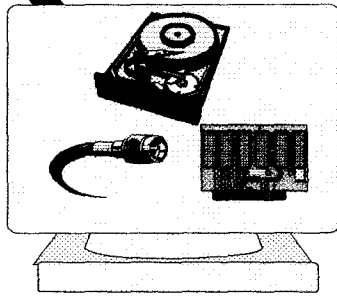
PORTABILITY



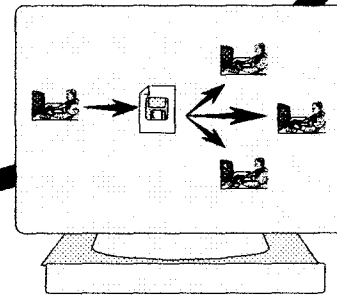
PERFORMANCE



FUNCTIONALITY



CAPACITY



SCALABILITY

For the past few years business computing has been evolving from megacomputer centers to what is known today as client/server. This means that multiple computers on a given network share in the processing responsibility. The computer that a user is sitting at (the client) does some of the work, and another computer which sits somewhere else on the network (the server) also does some of the work. This method of computing is a more open (diversifiable) and efficient way to process data.

Various entities have made a commitment to this environment even though problems have arisen inherent to the way the system works. Because it is very open, people use many different tools to accomplish their various tasks. Therefore, getting these different tools to work well together in different situations is a key issue. The only way to do this is to communicate, or simply share information concerning the problems one encounters with others finding themselves in the same boat.

This paper presents the results of a survey disseminated to over 500 information systems professionals throughout the domestic United States. These participants were selected at random, the only prerequisites being the respondents title (i.e. manager or director of MIS or Systems & Programming) and the corporation's listed systems. They were obtained from the Spring 1995 Computing IS Managers Guide. The average respondent had 2.48 years programming experience in client/server and 15.32 years programming experience total, lending credibility to the responses received.

Six facets of client/server application development were investigated. These six are:

- Performance the overall speed and efficiency that a developed application displays
- Capacity reflects the amount of hardware resources such as disk space, memory, and bandwidth that an application needs

- Scalability the ability of an application to work with an increasing number of users or across various network setups
- Portability reflects the ability of an application to be transported from one system and platform to another
- Connectivity the ability of various levels of tools used for development to communicate with each other
- Functionality the ability of a given tool to do what a programmer wants or needs it to do

Each respondent was first asked to rank the categories as they applied to their specific experiences. The ranking ranged from 1 to 6, a '1' meant the respondent felt the given area had the most problems, whereas a '6' meant it had the least. As can be seen

from figure 1, performance was the hands down winner(or loser, depending on how one looks at it) with 32% ranking it as number one. Connectivity

CATEGORY	% ranked as # 1	Average Rank
Performance	32	2.794
Connectivity	23	3.470
Capacity	3	3.970
Functionality	9	4.676
Scalability	9	4.941
Portability	0	5.235

Figure 1 - Ranking of Categories

followed closely behind. Capacity, functionality, scalability, and portability were deemed less of an issue. Although it appears functionality and scalability should be ranked above capacity, with 9% ranking them each as number one, the average ranking shows that capacity was overall more problematic.

It is necessary to remember, however, that problems in the other five categories can be construed as performance related. Therefore, more in depth discussion of these categories will help lend reason to the performance problem itself.

Performance

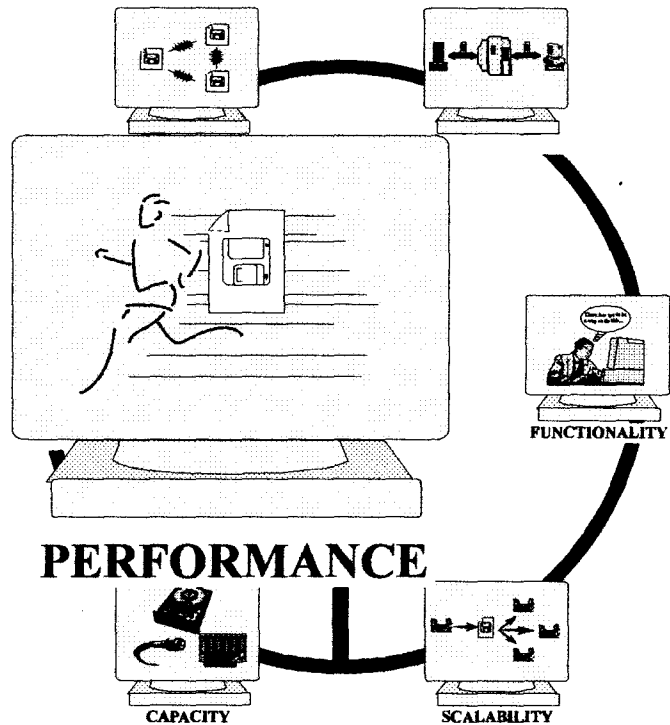
This survey did not attempt to limit the types of problems that could possibly occur related to performance. Respondents were simply asked, in the form of an open ended question, to discuss specific problems they had encountered.

Typical responses such as slow processors, insufficient memory,

and high volume of transactions were plentiful. There were, however, several other specific responses which connect performance to the other five categories.

The overwhelming consensus was that processing speeds(i.e. response times or throughput) were entirely too slow. Large data requests from OLTP systems against SQL databases were good examples of this. The efficiency of queries, due to inefficient keys and indexes on files, was unacceptable to the end user.

Extremely large quantities of data, coupled with WAN vs. LAN transmission speeds, was an instigator of many problems. In particular, file capacities on mainframe and mid-range systems tended to be more than PC's could handle. As a result, many programmers were forced to set up intermediate logical files to handle the excess.



Capacity

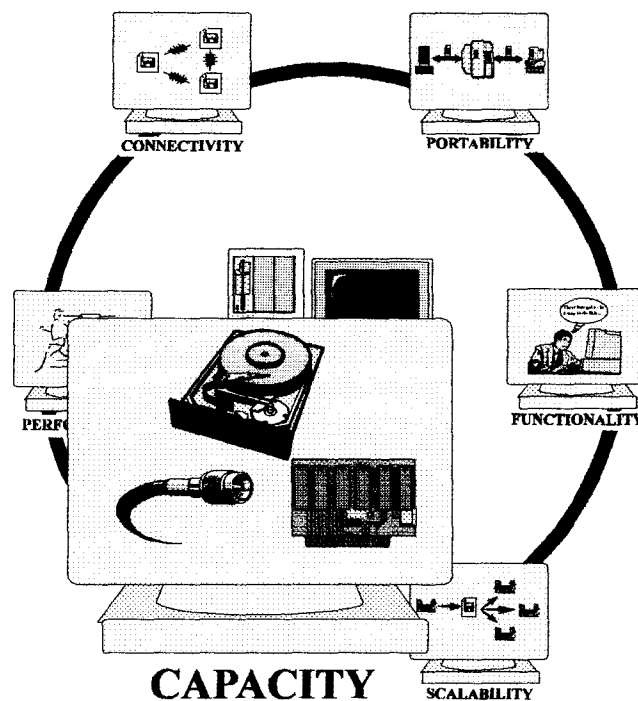
As previously defined, capacity deals with a given system's hardware capabilities. This area is perhaps the most important one to properly plan and implement.

Thorough capacity planning can help prevent bottlenecks and breakdowns before they happen.

In the days of the mainframe, capacity planning had become the norm. With a mainframe, it isn't much of a problem since it's operating system controls and knows everything that is going on within the network. With client/server, however, it isn't such a pretty picture.

Various operating systems and network topologies make it difficult to produce any one tool that can capture all the relative measurements. Added to the determination of future computing needs, this can become laborious at best. Although presently treated as optional, many companies will soon see that it will become a requirement.

Respondents were asked to relate specific capacity problems occurring in three areas; the client, the server, and the network itself. In general, there was a fairly even distribution of respondents indicating problems in each category. Approximately 65% indicated the client, 56% the network, and 42% the server. Currently there really are no 'one size fits all' tools to test all of these areas. If we can better understand where the problems are occurring, however, we can build better tools to fix them.



On the client side, insufficient CPU speed and insufficient memory were the major instigators, each receiving 45% of the vote from respondents. This, of course, makes sense. It does not matter how fast a computer's processing unit is if it has insufficient memory to support the current task. The remaining 10% indicated I/O speed to be a problem on the client. This also makes sense since most experienced programmers develop applications with the intention of accessing the disk as little as possible.

Other specific problems, mentioned on the client, dealt mainly with hardware considerations. Some said their hardware was too old, while others said they didn't have enough (memory, disk space, etc.). There was an overwhelming agreement that, at the *absolute* minimum, 8 meg of RAM is needed at the client. However, most suggested as much as 16 to 32 meg.

Similarly, along the lines of processors, 486 was deemed the minimum but a Pentium was stressed as the optimum. The overall link is that with each new version of the operating system or database, the new features are nice but to use them you really need the power.

On the server side, approximately 50% of the respondents indicated insufficient CPU speed as the major problem with 25% each for I/O speed and insufficient memory. With most of the processing still going on at the server, this was not surprising.

Many respondents suggested that a Pentium processor was the minimum for an efficient server, along with 64 to 128 Meg of RAM. Some suggested using the mainframe as the server.

On the network itself, both insufficient LAN/WAN bandwidth and bottlenecks at routers, bridges, and gateways drew 42% of the respondents. The reasons dealt mainly

with high volume OLTP traffic, data problems due to large graphics files, and large files from mainframes. With real-time applications, besides the congestion causing sluggishness, another problem to deal with is the loss of messages over the network. It seems everyone these days is screaming for more bandwidth, a testimony to the merit of capacity planning.

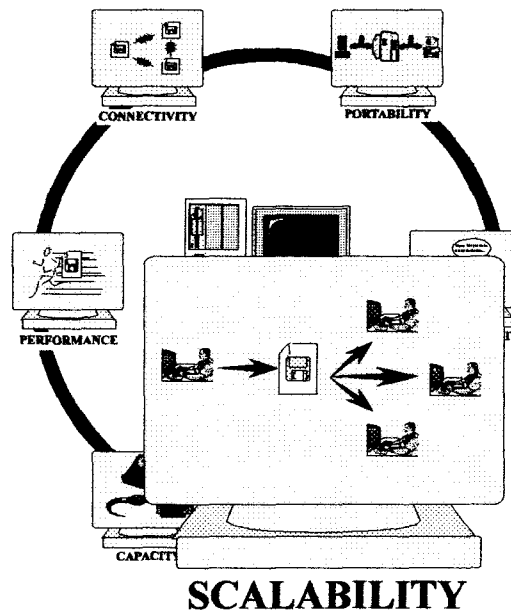
The remaining 16% were votes for an incorrect network topology. With client/server demands ranging from SQL database queries to video teleconferencing on the rise, this author expects this category to be a major future concern. A reason for the response in this area being so low may be the software-oriented rather than hardware oriented nature of the respondents.

Most interesting, over all, was the number of respondents who said they had performed some formal capacity modeling before implementing their current systems. Approximately 77% did not utilize modeling. Of the 23% who said they did, only one named a specific tool used to do the modeling. Those who did perform the modeling still experienced many of the same bandwidth and memory problems as those who didn't. The concept of capacity modeling is not understood or utilized as extensively as need be.

The biggest problem of capacity planning may lie in a totally different area -- the cost. The tools available today, that can do the job right, are considerably expensive. This is due to the fact that it tends to be the large companies with complex systems which take the time to do the capacity modeling. The time needed to develop a capacity modeling application, and support it, is considerable.

Scalability

Building applications that are truly scaleable in a client/server architecture is quite a challenge. Trying to cross differing platforms or utilizing GUI, as well as character based, interfaces can be troublesome. Finding the right tool is critical. Many programmers find it necessary to drop down into a 3GL to solve more complex tasks.



This survey did not try to solve the scalability issue, it simply attempted to get a feel for where programmers are having the most problems. The three main issues discussed dealt with breakdown due to:

- complexity
- increased number of users
- crossing of platforms.

As noted earlier, scalability was, on average, ranked the second least common area for problems. This is expected to change over time, however, as the number of enterprise-wide applications increases. Therefore, it still merits discussion.

Of the respondents who did have problems, there seemed to be a split down the middle. The breakdown of applications, due to an increasing number of users, received 42% of the vote as well as did an application's ability to work across different platforms. The remaining 16% felt that the actual complexity of the application itself caused the breakdown.

The main concerns seemed to involve accessing mainframe and mid-range data from windows based tools. Some of the other specific problems, here, dealt with vast amounts of data from a mainframe, large amounts of old legacy code mixed with new, and the general inability of GUI development tools to work well in a complex environment.

To solve scalability issues programmers can either, one, piece together their own open systems solution or, two, buy a proprietary solution. This survey attempted to discover which most programmers prefer and why. It turned out that 75% of the respondents said 'no' to the proprietary solution; however, reasons for going either way were very convincing. Figure 2 lists the advantages and disadvantages for going with the proprietary solution.

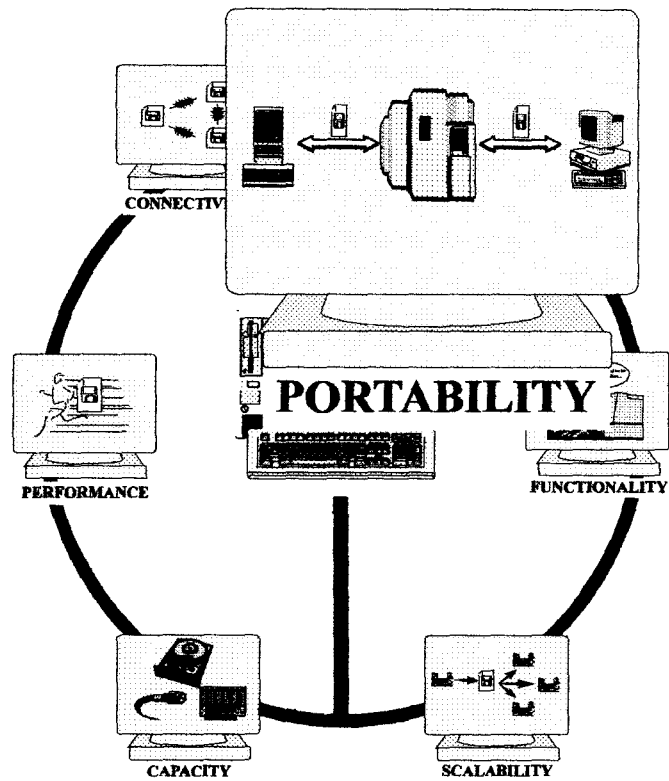
Advantages	Disadvantages
<ul style="list-style-type: none">• Optimal if early in the project before investing too much into the system.• Reliability, standardization, warranty• Inexperienced staff can't build own• Relative quickness to solution	<ul style="list-style-type: none">• Cost too high• Inflexibility• Inability to perform adhoc queries• Dedicated to open systems

Figure 2 - Adv. & Disadv. of Proprietary Solution

Portability

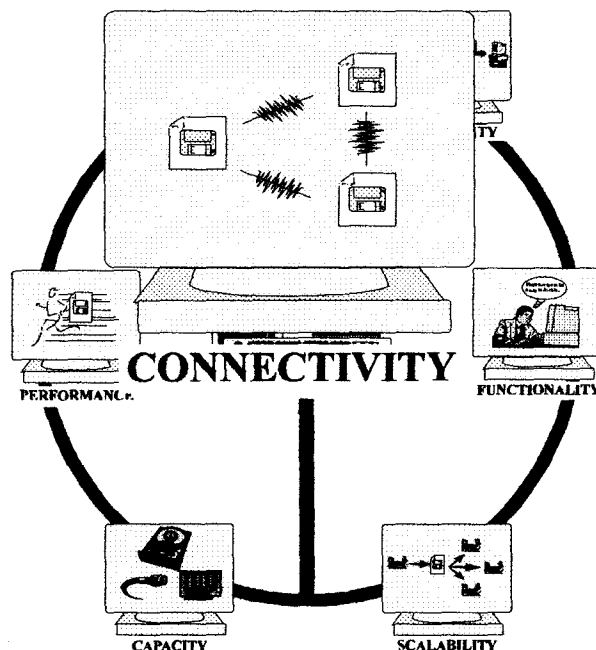
With an average ranking of 5.235, portability was considered the smallest problem area. This may be due to the uncommon nature of actually porting an application and may change as more large businesses continue to perform systems integration. The specifics mentioned were as follows:

- necessity that all users use the same versions of software packages
- differences in UNIX versions cause design and tuning to be difficult
- going from UNIX to PC/DOS is difficult
- stored procedure reliability is uncertain



Connectivity

Concerning connectivity, respondents were asked to discuss problems as they might occur in three separate areas: front end, back end, and middleware. Each area received several, specific responses worth mentioning. The front end and middleware really had the most complaints. In addition, there was a general consensus concerning the viability of programming in client/server at all due to connectivity struggles.



Front end problems really varied. Loading client tools on different PCs, in a consistent manner, seemed difficult. Respondents complained that what worked on one didn't work on another. Difficulty getting certain terminal emulators to work with certain servers was also a problem. The scripting and mapping of keyboards to make use of special keys seemed to be a hassle. And, last but not least, the issue of large volumes of data causing applications to fail, again, appeared.

The complaint, in the back end area, was the incompatibility of SQL commands from front ends with the back end itself. Secondary, there was also mention of a problem connecting the clients to the server database, which even the vendor in question couldn't explain!

Middleware, by far, received the most notoriety. In particular, ODBC was a popular topic. Certain ODBC drivers were considered weak and forced programmers to

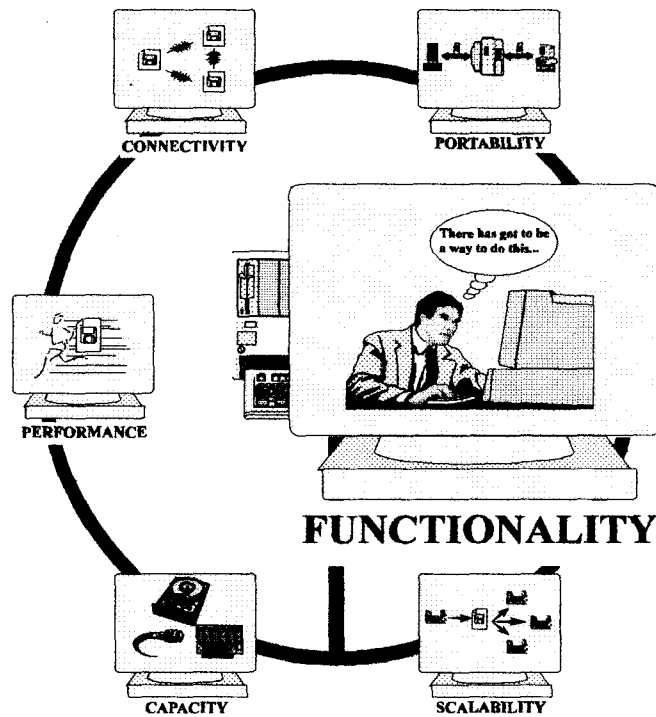
write their own or use remote procedure calls. With the lack of standards, this proved to be extremely time consuming.

Large volumes of data, again, was a consideration. Backlogs at routers, bridges, and gateways were mentioned. Many who worked with mainframes complained of connecting them in with other servers. Because of their proprietary nature, some found they had to execute additional, special transactions to take care of the data on separate systems.

According to a paper by Mitch Kramer, in the April '94 edition of Client/Server Computing, there are two ways to approach middleware programming. One is to provide APIs on both the mainframe and the client applications; the other is to make the mainframe data appear relational. ODBC drivers, for instance, fall into the second option. As one can imagine, the overall lack of expertise in this area, coupled with the extremely steep learning curve, feeds the inability to develop good middleware. Respondents felt that the need for extensive application development, as well as extensive network experience, was a *minimum*.

Functionality

The availability of 3GL, 4GL, and 5GL languages or software packages gives programmers a wide variety of choices within which to build their client/server applications. This gives rise to confusion as to which tools are the best for which jobs. Therefore, this survey attempted to get some feedback from the people choosing the tools right now.



An absolute myriad of software packages were mentioned by respondents, with varying degrees of satisfaction and dissatisfaction. The object is not to label any one tool as of ill quality, but rather to see if certain kinds of software development tools showed up numerous times with the same complaints.

The responses were so diverse that the best way to present them is simply to list the most prominent. The following list contains the functionality comments which seemed the most important:

- Steep learning curves especially coming from mainframe environment
- Lack of good tools for migration of applications, deployment, and data transfer
- Lack of standardization
- Report writing tools in their infancy
- Although environments are improving, they still don't meet vendor claims
- High costs after training seem to outweigh benefits
- Unavailability of batch work in some GUI development tools

Two excellent papers, one by Gregory C. Garry, in the August 1995 edition of Client/Server Computing, and the other by Pat Koleini, in the March 1994 issue of the same, discussed weighing the differences between the 3GLs and 4GLs. Some of the major considerations were reliability and stability, availability of 3rd party support, and the ability to generate true enterprise-wide client/server applications.

In general, the 3GL are considered more functional and portable, generate less overhead, and are less expensive to utilize because of the prevalence of knowledgeable programmers. The 4GL, on the other hand, obviously offers the visual appeal along with the ease of point and click. It seems that a combination of the two actually is often the best solution, and that none of the language generations can be considered obsolete or on their way out.

Respondents were asked in this survey to indicate their preferences for 3GL or 4GL. Only 13% of the respondents to this survey preferred the 3GLs. This author found this extraordinary, considering that most other literature indicates a strong reliance on 3GLs. Although this may or may not be indicative of actual industry distribution, it certainly merits further investigation. It is notable that 55% of those who preferred the 4GL had used some type of 3rd party support for that tool and that 50% for the 3GLs did the same. By these figures, it appears that, the 4GL tools have gotten harder rather than easier to use. Users require more training and outside support and they simply aren't as functional as of yet.

Conclusion

This research was conducted as a first step in finding out where client/server programming problems are most often occurring. With this baseline information, further study into the most pressing or relevant areas can be accomplished.

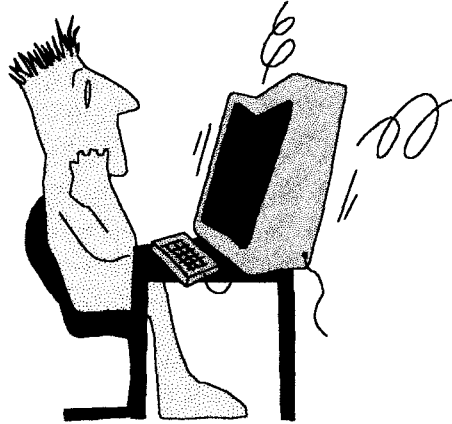
In this author's opinion, the areas which stand out most are capacity, connectivity, and functionality. Even though performance received the greatest mention from respondents, it is the other areas which cause problems leading to poor performance.

Therefore, in order to propagate solution of client/server programming problems, the next step is to take this information and apply it to further study. Capacity modeling, connectivity standards, and 4GL weaknesses are just three of many good examples. One must remember, however, that the nature of the responses received is extremely diverse. Comparison of the data is difficult and time consuming, but must be done thoroughly if the right answers are to be found.

References

- Bort, Julie 'A Rapacity for Capacity Planning', Client/Server Computing, October 1994, pg. 88+.
- Bort, Julie 'Modeling Helps End the Network Bottleneck Squeeze', Client/Server Computing, June 1995, pg. 39+.
- Garry, Gregory C. 'Weighing Trade-Offs Between 4GL, 3GL Alternatives', Client/Server Computing, August 1995, pg. 76+.
- Koleini, Pat 'Tried But True 3GLs Represent Versatile Alternatives for C/S Developers', Client/Server Computing, March 1994, pg. 59+.
- Kramer, Mitch 'IS Managers Take the Middle Road', Client/Server Computing, April 1994, pg. 60+.
- Lawton, Stephen 'Catching the Right Network Transport Mode', Client/Server Today, September 1994, pg. 43+.
- Radding, Alan '4GL Developers Finding Scalability Has Multiple Meanings', Client/Server Computing, May 1995, pg. 63+.

Client/Server: What Exactly Is The Problem?



October 21, 1995

Dear IS Professional:

As the business world continues to migrate to the client/server architecture, the need for successful application development techniques becomes ever more pressing. This survey has been sent to you with the hope that you might participate by relating some of the various problems you may have encountered within the client/server environment. Your contribution to this study is invaluable.

My name is Allan Furman and I am a student at Northern Illinois University. I am presently working on my senior year honors thesis. I am sending this survey to over 500 IS professionals around the United States. My plan is to investigate and uncover any common links between various problems and solutions being encountered while developing client/server applications.

This project is being sponsored by the Operations Management & Information Systems Department at Northern Illinois University. Dr. Richard Born, a professor in MIS, is advising on the research/analysis process. All information you provide is confidential.

If you could return this survey by November 15th it would be greatly appreciated. If you would like a copy of the results please include your business card with your completed survey. Thank you very much for your participation!

Sincerely, Allan Furman

1.) Are you currently developing or have you developed applications within a client/server architecture?

Yes : Please continue to question # 2.

No : If possible, please pass this on to someone who has. Otherwise, just return the survey as is.

2.) Please list the client/server platforms(the hardware and software) that you have worked with.

3.) Please rank the following possible problem areas from 1 to 6, with 1 having the most problems and 6 the least. If one of the areas does not apply to your situation, simply leave it out of the ranking.

Performance	___	Capacity	___
Scalability	___	Portability	___
Connectivity	___	Functionality	___

For the remaining questions, answer only the ones which correspond to the areas you have included in your above ranking.

Performance

4.) Please briefly describe what performance issues/problems you encountered, how you went about solving them, and what the final resolution was.

Capacity

5.) My capacity problems were encountered on the :

Network:	Insufficient LAN or WAN bandwidth	___
	Bottlenecks at routers, bridges, gateways	___

Inappropriate topology _____

Other (_____) _____

Please briefly explain those you have checked:

Client: Insufficient CPU speed _____

I/O speed _____

Insufficient memory available _____

Other (_____) _____

Please briefly explain those you have checked:

Server: Insufficient CPU speed _____

I/O speed _____

Insufficient memory available _____

Other (_____) _____

Please briefly explain those you have checked:

- 6.) Did you perform any capacity modeling beforehand? Yes / No
If so, was any particular capacity planning tool used? Yes / No
Name of tool: _____

Scalability

- 7.) Did your application breakdown because of:
Complexity of the application itself _____
Increasing number of users _____
Inability to work across different platforms _____
Other (_____) _____

Please briefly explain those you have checked:

- 8.) Would you consider an expensive proprietary solution as opposed to piecing together your own open system solution? Yes / No

Please provide any specific reasons why:

Portability

9.) Please briefly describe the portability issues you encountered.

Connectivity

10.) Did you use any particular middleware product? Yes / No

Name of product: _____

11.) Please rank 1 to 3 where you feel the most problems occurred. (1 having the most problems and 3 the least)

Front end software (client side) _____

Middleware _____

Back end software (server side) _____

12.) Please briefly describe any specific connectivity problems you experienced within those areas you have checked.

Functionality

13.) Please list the programming tools that you have been using for client/server work.

- 14.) What percent of the time do you feel you came to a point in which you just simply couldn't do what you wanted with the tool you were using? (Check one for each tool listed)

Name of tool: _____	Name of tool: _____
0 - 5 % _____	_____
5 - 15 % _____	_____
15 - 25 % _____	_____
25 % or more _____	_____

- 15.) At the present time, I would prefer to develop client/server applications in a 3GL / 4GL . (please circle one)
- 16.) Have you found and taken advantage of any 3rd party support for your programming tools? Yes / No (please circle one)
- 17.) Please describe any other functionality issues you have encountered, know of, or feel are pertinent.

Demographics Section:

Any information supplied in these sections is confidential and will not be released in any manner. It is simply for the researchers use in analysis.

- 18.) Please list any degrees and/or certifications you presently hold.

- 19.) Number of years programming experience in client/server? _____
- 20.) Number of years programming experience total? _____

Thank you very much for participating! Again, if you would like a copy of the results, simply include your business card.