Client/Server Conversions: Balancing Benefits and Risks
by Leonard J. Mignerey

Currently only a small fraction of the information created and used by an organization is stored in electronic format. An even smaller percentage is readily accessible to its potential users. Client/server computing and the broader concept of distributed computing have the potential to significantly increase both the percentage and the utility of organizational information captured in electronic format. The excitement surrounding these technologies is warranted: the potential benefits of distributed computing are manifold. However, general user excitement, coupled with intense vendor marketing campaigns, have created levels of expectation that outstrip the current level of technological reality.

For students with inventive minds, the gap between expectations and reality can provide opportunities for solving real world problems in the rapidly evolving arena of distributed computing. However, such a gap is less positive in the environment of production-grade business support systems. Here the intense rate of change in underlying technologies, coupled with inflated expectations, often results in a high-risk environment. The client/server forms of distributed computing concepts are now old enough that there have been many valuable “lessons learned” from those who adopted technology early on (particularly in the private sector). There are many areas where tremendous benefits can and should be gained from investing in client/server technology. There are also at least an equal number of areas where this technology is not yet appropriate. This article explores and makes a case for why this is so.

Establishing a technology context

Since the 1950s, as computers became more powerful and multi-functional, they assumed more and more of the manual work performed within the business units of most organizations. The natural outgrowth of this process was a single powerful machine capable of supporting the core business functions. The hardware and software that supported the functional systems could be thought of as an environment in a box. Over the years mainframe architectures were refined to an extreme level of efficiency and reliability. These systems are able to service large numbers of users, enforce extremely complex sets of business rules, and deliver very rapid response time, while accessing extremely large databases of information.

Like the process that took place with mainframe hardware, the process of application development was gradually refined to several broadly acceptable methods for implementing the entire process—the software development life cycle, structured code, structured walkthrough, change control, documentation, and so forth. The result has been (for the most part) the evolution of highly reliable code that is currently supporting the mission-critical functions of most businesses and universities.

Over the last five or six years natural questions arose—“Why can’t the ease of use and rapid development of applications that have become the norm in the microcomputer/local area network environment be extended to the mainframe environment?” and “Why not bring the mainframe into the network environment and make the data that it has been maintaining all these years available to everybody?” These are natural questions with difficult and rapidly evolving answers.

The traditional mainframe environment has the following components: a terminal attached directly to the mainframe, a CPU, main memory, data input and output electronics, and large storage devices. Today such environments run with a level of reliability that exceeds 99 percent up-time. This is called a “mainframe-centric” environment. The client/server concept spreads computing components across the network for a “network-centric” environment. It is a mix-and-match environment in which almost any one component can either ask for services from any other component (be a client), or provide services to other components (be a server). It expands the very deterministic (because all the pieces are tightly controlled) mainframe environment and creates a non-deterministic environment of loosely coupled components. Current industry wisdom indicates that the best reliability statistics for environments that are complex...
enough to be considered client/server are experiencing about 85–95 percent up-time and provide significantly inferior response times. “Mainframes deliver 99.6 percent availability and sub-second response times, while most LANs [just the LAN, not the entire client/server application] are available only 92 percent of the time and take almost three seconds to respond to users.”

Environmental differences

In the mainframe environment, applications were written that supported enterprise-level (as opposed to department-level) applications. Typically such applications need to enforce the myriad rules and exceptions to those rules, which in total account for the operational essence of an enterprise. These applications are known as “mission-critical” applications. If mission-critical applications malfunction, some essential part of the enterprise will be seriously affected. One example of such an application is a payroll system.

In the monolithic software model, mission-critical applications generally have a large business-rule component, a small user-interface component, and a large data-interface component. PC departmental applications generally have a different structure. Here the majority of the code (generally machine generated) is in the graphical user interface (GUI) and in the data interface layer (also mostly machine generated). The business rule component is extremely small. Traditionally these applications are not mission critical, and are limited to a relatively small sphere. An error in such an application is typically easy to spot, and will not affect the general functioning of the enterprise. Since the majority of the code is machine generated and the domain of the application is limited, alterations to the application are generally easily accomplished. If necessary, an entire application can be scrapped and re-written from scratch.

This type of application development has spawned the concept of rapid application development (RAD). The tools of RAD have been termed fourth-generation languages (4GL) and fifth-generation languages (5GL). Much of the enthusiasm surrounding the client/server concept comes from the desire to create enterprise-level applications using this machine-generated coding concept. Extravagant claims of success notwithstanding, there have been a number of stumbling blocks to achieving this goal. The business rule component is the major difficulty. The complexity involved in coding applications that are mission critical to the entire enterprise is orders of magnitude more difficult than developing departmental-level, non-mission-critical applications. Some 4GL/5GL products have been fairly successful in dealing with the general rule portion; however, it is the exceptions to the rules that are the Gordian Knots of RAD.

For a few years high expectations were placed on a class of technology called computer-aided software engineering (CASE). Few technology professionals would argue the point that the usefulness of CASE tools fell far short of expectations. Attempting to represent the exceptions to business rules with CASE logic was often more difficult than not using the tool at all. We are seeing a new wave of CASE-like tools and techniques (distributed object computing is one) that are attempting to solve this problem.

Politics vs. technology

The major factor contributing to such failures is that the business rules, and particularly their exceptions, are really political issues, not technical issues. Technology cannot now, nor will it ever be able to, replace the political process that is necessary for making decisions or coming to consensus on the business rules (or their exceptions) in an organization. The process of examining business processes, facing and managing emergent personnel issues, reaching consensus, and so forth, is time consuming. Technology has been able to do little to compress the time that is needed to do it properly. The wider the scope of the project, the more people and departments involved, the more difficult and lengthy the political process is. It is easy to slip into thinking that there must be some new technology to deal with this tedious process. Individuals familiar with the limited scope of departmental projects often fail to understand why those at the enterprise level “just can’t get it done.” The complex political process is the primary reason.

The current interest in business process reengineering (BPR) is an outgrowth of the problems caused by process vs. technology issues. BPR attempts to put the process before the technology. With BPR the organization must first redefine what it wants to do, then establish the political and functional structure to get it done, and only then select and implement a technology to facilitate the redefined process. There have been documented successes with BPR, and there have been at least an equal number of failures. There is a school of BPR thought that believes that the real process with BPR is “failing until you succeed”—the first one or two iterations of process reengineering should be expected to fail. Without debating the merits of the argument, I think it at least makes the point that BPR is not rapid.

Exhibit 1 offers a list of some of the business

Exhibit 1: Business issues to be resolved in conjunction with choosing a technology solution

- Does the institution know what its key business strategies are? Is there upper level, administrative consensus on what major projects should be undertaken and in what order? Where does the proposed project fit in that list?
- Are the business assumptions of the proposed project documented, and is there agreement on the definition of those assumptions?
- If the proposed project is accepted, what effect will its implementation process have on other proposed projects? If this project is undertaken, could its scope delay the benefit that could be derived from equally important alternative projects with a smaller scope?
- If the planned project will be replacing an existing system, has the institution really changed (or can it really change) the organizational structure that the original system is supporting? If not, then the actual benefit of the new system will likely fall far short of the expected benefit.
- If the project will be supporting organizational change, do the key individuals in the respective functional areas have a deep understanding of their jobs? Do they have the ability to conceptualize how to cross the gap from the old model to the new model? Can they lead this effort for their area? While some individuals are excellent in the functional demands of their existing jobs, they do not always have a deep theoretical understanding of their areas of responsibility. If this is the case, is the institution willing to delay the project and rectify key problematic personnel issues?
- Few (if any) organizations actually run by their formal documented procedures. This fact can bring significant resistance to the change process. The informal procedures are often buried deeply in the organizational structure and are difficult to ferret out. Will the project have an appropriately powerful administrative body to find and resolve the resultant issues in a timely manner?
- Does the institution understand the risks of project failure? Based on initial projections, if the project is not implemented within a reasonable time frame or fails outright, will any mission-critical functions be in jeopardy?
- Are expectations for the new system based on proven product performance or on vendor promises? How closely has this been examined?
- Does the institution understand the full costs of the project? For example, if it is a client/server project, is the necessary infrastructure in place? If not, can the institution afford to purchase and maintain all the desktop equipment that will be necessary to fully deploy the new system? If the institution does not have or cannot afford a robust network infrastructure, and if the full project rollout cannot be centrally funded, then a client/server, distributed computing project is probably not the right institutional choice regardless of the inherent benefits of the actual product.
- Has the institution examined the transition costs of maintaining the old system while implanting the new system?
- Related to the previous issue, will the institution be able to retire the old system when the new system is implemented? If not, does it understand the costs and ramifications of running both the old and new systems?

Issues that should be resolved prior to discussions of what types of technology would be appropriate for a given project.

Huge technological advances and associated high expectations (too often inflated by eager vendors), often lead organizations into a premature attempt to find technological solutions to the difficult but necessary political process. It is often at this point that consensus for a purchased package gains substantial momentum. The process of evaluating, selecting, and budgeting for a package is a process that produces tangible results. The participants in the process feel that they have finally accomplished something. In reality, the problems of BPR have simply been deferred. The organization now has the much larger problem of attempting to do BPR simultaneously with or after a major software installation/conversion. The disruptive force on the organization is extreme and becomes a prime reason for project failure. The culmination can be a large financial loss and organizational disarray.

Some client/server realities

Much of the confusion and misunderstanding surrounding client/server stems from the fact that it is the wrong subject to focus on. The fundamental environmental change in technology is not client/server technology; it is the existence of ubiquitous networks and the technology to operate them at a reasonable level of reliability. Networks add to the computing environment what air, road, and rail systems add to an industrialized nation—the ability to effectively distribute goods and services. Networking brings distribution of services to the computing environment and leverages the effectiveness and power of the connected computing components in a way analogous to the industrial example.

To continue the industrial analogy, it is clear that the establishment of a transportation network did not eliminate large centralized manufacturing plants. The reverse is true. Larger, more centralized manufacturing plants grew to replace many of the smaller regional production facilities. And, as with computing, aggregate investment in industry did not decline, but expanded exponentially.

Cost issues

The issue is not “mainframe bad, UNIX good, PC better.” It is a question of which technology to use where, for the greatest efficiency and the lowest cost. Lower cost per function does not mean that the cost of computing in general is less now than it was in the past. In fact computing in aggregate is becoming more expensive at an expanding rate. Certain parts of the process may be accomplished more cheaply with smaller
machines, but the overall number of computing tasks is increasing; and the necessary hardware, software, and staff to support these tasks are also increasing.

LANs cost nearly three times as much per user as do mainframes, according to a study by International Technology Group (ITG), a consultancy based in Los Gatos, California. After surveying 250 sites and analyzing 20 percent of them in depth, ITG found that the yearly cost per user of mainframe installations averages $2,282, while LANs average $6,445. ITG also found that the average cost per mainframe transaction is 3 cents, compared to 43 cents for LANs. In April of 1994 the Gartner Group released A Guide for Estimating Client/Server Costs. In what has become a widely cited report, they estimated that the total five-year costs for client/server applications per user were approximately $50,800 for small enterprises; $66,350 for medium-sized enterprises; and $48,400 for large enterprises. It was also interesting to note from the component cost breakout that hardware costs represented a small fraction of the total costs. Labor costs were, overwhelmingly, the largest component (see Exhibit 2).

Conversion risks

There is large risk, large investment, and tremendous complexity involved with mass conversion of existing central systems into the client/server model. Even if successful, such conversions do not necessarily provide significant additional access to central data and may actually delay the ability to do so. (Is it really necessary to replace transactional processing systems that are functional when what the campus really needs are decision support systems, which can be created without systems replacement?) In addition, one must seriously question the conversion of a highly stable system into a less stable system, unless there is some overriding benefit to be gained from the conversion.

Organizations often fail to grasp the depth to which centrally maintained systems are woven into the fabric of the enterprise. These systems are not layered on top like a hat; they cannot be swapped on a whim. In the monolithic model of traditional systems, the business role section is the predominant issue. To convert these systems, regardless of the target technology, the business rules must be re-examined. This is the BPR process that was discussed earlier, and there are few shortcuts in that process. Mission-critical applications almost always have a significant business rule component. The cost of converting an application with a significant business-rule component will be high because of technical complexity, conceptual complexity, testing to ensure that the application still does what it is supposed to do, and the non-compressible political/process component.

Rapidly changing technology

Client/server computing is in its infancy, and the architectural models are changing constantly. With each client/server application the question must be asked, “Can we afford to scrap, re-build, or re-buy this application in two years?” In view of the factors surrounding mission-critical applications, it is a very serious question. In industry, this question can often be answered in the affirmative. In the private sector, even a marginal competitive advantage can have extreme benefit. The windows of competitive advantage are generally very narrow. Two years is a long time. An application that provides or supports such an advantage, and that can be developed quickly, is of high value. It is worth a significant short-term investment to gain the advantage. There are few analogs to this scenario in the higher education environment.

Increased complexity

The risk factor of distributed systems requires some explanation. There is a well-accepted rule that each additional component introduced into a system (any system) raises the complexity of the system geometrically. Over the last decade automobile manufacturers have been applying this principle to automobile production. They have intentionally eliminated as many parts as possible from their designs; a reduction in the number of parts reduces the failure rate of the vehicle. The complexity of distributed systems is at least an order of magnitude greater than traditional systems, regardless of the target technology.

Conversion risks

There is large risk, large investment, and tremendous complexity involved with mass conversion of existing central systems into the client/server model. Even if successful, such conversions do not necessarily provide significant additional access to central data and may actually delay the ability to do so. (Is it really necessary to replace transactional processing systems that are functional when what the campus really needs are decision support systems, which can be created without systems replacement?) In addition, one must seriously question the conversion of a highly stable system into a less stable system, unless there is some overriding benefit to be gained from the conversion.

Organizations often fail to grasp the depth to which centrally maintained systems are woven into the fabric of the enterprise. These systems are not layered on top like a hat; they cannot be swapped on a whim. In the monolithic model of traditional systems, the business role section is the predominant issue. To convert these systems, regardless of the target technology, the business rules must be re-examined. This is the BPR process that was discussed earlier, and there are few shortcuts in that process. Mission-critical applications almost always have a significant business rule component. The cost of converting an application with a significant business-rule component will be high because of technical complexity, conceptual complexity, testing to ensure that the application still does what it is supposed to do, and the non-compressible political/process component.

Rapidly changing technology

Client/server computing is in its infancy, and the architectural models are changing constantly. With each client/server application the question must be asked, “Can we afford to scrap, re-build, or re-buy this application in two years?” In view of the factors surrounding mission-critical applications, it is a very serious question. In industry, this question can often be answered in the affirmative. In the private sector, even a marginal competitive advantage can have extreme benefit. The windows of competitive advantage are generally very narrow. Two years is a long time. An application that provides or supports such an advantage, and that can be developed quickly, is of high value. It is worth a significant short-term investment to gain the advantage. There are few analogs to this scenario in the higher education environment.

Increased complexity

The risk factor of distributed systems requires some explanation. There is a well-accepted rule that each additional component introduced into a system (any system) raises the complexity of the system geometrically. Over the last decade automobile manufacturers have been applying this principle to automobile production. They have intentionally eliminated as many parts as possible from their designs; a reduction in the number of parts reduces the failure rate of the vehicle. The complexity of distributed systems is at least an order of magnitude greater than traditional systems, regardless of the target technology.
centrally based systems. The controlled environment in a box has been spread out over a network, and the optimization that had been achieved with the former model is lost in the latter.

Response time is a key area affected by increased complexity. Colleges and universities generally run public networks, and all students are entitled to a computer account that includes network access. There is no control over the amount of network traffic that the student (or anyone else on the network) can generate. The exponential growth of network traffic, coupled with the exponential growth of new uses for the Internet, makes it impossible to ensure acceptable levels of on-demand capacity for mission-critical applications.

The software development environment suffers from increased complexity as well. For example, in a distributed computing project, different parts of the project may be developed by different entities both internal and external (vendors) to the organization, under completely different management domains. The design and project management that was formerly centrally controlled becomes design and project management-by-committee. This causes many levels of indirection. Slippage in one area of the project is leveraged as it propagates throughout the entire project. Scheduling and re-scheduling of the project pieces, and even the committee meetings to manage the project, can become a logistical nightmare. In addition, such a project is often made up of components that are supplied by different vendors and yet must interact as an integrated unit.

Need for standards and reliable level of service
This is an area where the current lack of distributed computing industry standards is of particular concern. To successfully implement a project one must ensure that all the pieces are going to “hand shake.” Once the project is implemented, it must be continually monitored as the various components from disparate vendors go through version upgrades. A version upgrade to the product of one vendor must be compatible with the existing version levels of all the other vendor products.

The need to keep pace with the version levels that the various vendors are supporting can often mandate conversion/upgrade projects that are entirely vendor driven, meeting no expressed internal need. For these types of projects the cycle time on version upgrades is extremely frequent; twice a year is conservative. If a project has three components (a very simple project), and each vendor changes versions twice a year, and there is no coordination between the vendors on the timing of version upgrades, and all of the components must continue to support each other, then the maintenance problem becomes obvious. Sometimes everything does work fine together. However, significant time needs to be invested with each change to ensure that it will. Otherwise the application cannot be considered a production-level application, i.e., an application that delivers a reliable level of service.

It will be a number of years before the distributed computing environment approaches the level of stability that has become the norm for traditional computing architectures. Client/server application-development models from as little as a year ago are being questioned and replaced. The vaunted “open environment” is still a long way away from having standards for application development and deployment. Industry standards are the “glue” of the client/server distributed computing environment. The much advertised “open” concept should not be confused with industry standards. Open means that a vendor has published the application programming interfaces for a product. This simply means that other vendor products can interface with the open product. That does not make the product an industry standard, although obviously most vendors would love for that to be the case, for at least some of their products.

Client/server technology cannot stabilize until standards are in place. Therefore, investment in this arena is risky. In the absence of standards, companies that are investing in pre-packaged client/server applications are becoming increasingly dependent on their chosen vendor’s proprietary solutions. “There is a danger that IS managers will buy applications or tools that have proprietary middleware underpinnings that don’t interoperate. Many vendors are using middleware as a convenient way to lock users into their product lines.”


44 CAUSE/EFFECT Fall 1996
with the vendors’ version upgrades, which have associated high costs.

The vendors and other devotees of client/server technology claim that the new development tools are powerful enough to rapidly convert existing application suites to keep them current. The fallacy of this claim lies once again with the business rule component and the political process. Vendors of large-scale client/server applications have yet to prove that major version release transitions can be a trivial affair.

**Alternatives to core systems investments**

Two conceptually different sources of data need to be maintained by an organization. The traditional systems are data accumulation systems. They generally access/add/modify/delete information into the organization’s operational databases one record at a time. This is also known as transaction processing. The newer systems are data access systems. The information that they access is contained in data warehouse databases.

The data warehouse is a concept that is receiving tremendous amounts of attention. The reason for all the interest is that data warehouses provide a new and much needed service and probably represent the greatest benefit in relation to cost of any other client/server project. Data warehouses are the new information access engines into the vast store of data that has been accumulated by the traditional operational systems for years.

The data warehouse has another major benefit. Since it is an access tool and not an accumulation tool, it does not have to incorporate the myriad business rules that are such a large part of the traditional systems development process. Therefore, it is an area where an investment has a low risk of failure, enjoys a high degree of user acceptance, and can be developed in relatively quick time frames. In addition, since the business rule component is the weak link of client/server development, and since this is an extremely small component in the warehouse architecture, it is an excellent place to develop client/server applications.

Much of the perceived need for new core systems stems from data access issues rather than data accumulation issues. The benefits that a data warehouse can bring to an organization, such as improved decision-making, improved service to our students, and so forth, have been documented in many excellent articles during the last two or three years.⁵

**Conclusion**

For all of its problems, there is little if any doubt that distributed computing is the computing environment of the future. As solutions are found to some very difficult technological problems, more and more traditional computing applications will migrate to the distributed client/server model. In the meantime, it is important to clearly understand what it can do well today, and what it cannot do well. In some areas client/server computing can already provide valuable enterprise-level services that are not possible with traditional computing architectures. However, industry wisdom is clear that it is not wise to convert existing mission-critical applications to client/server technology if the only reason for doing so is simply to convert the system to client/server technology. It is also clear that client/server technology does not mean one technology at the exclusion of another. The future of distributed computing will see many types and classes of machines doing some part of the electronic “magic” that each does particularly well. Unless there is some overriding strategic reason to convert an existing system to client/server, there is a much greater potential benefit in a client/server investment that will provide some currently non-existent service.

The information technology market will look much different in the next five years and will be radically different over the next ten years. Five years is an eternity in the technology business, but it is only a heartbeat in the higher education environment. This “time reality” mismatch represents an area of high risk. By evaluating our institution’s current investment in existing administrative systems, and the benefits that some of the newer technologies can realistically provide, we have an opportunity to significantly improve the service level that administrative computing can provide to the institution.

⁵ See http://cause-www.colorado.edu/information-resources/ir-library/subjects/data-warehouse.html