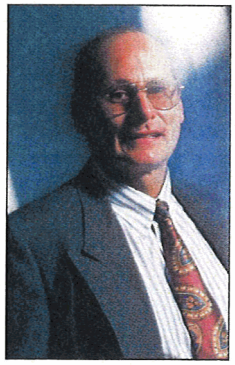


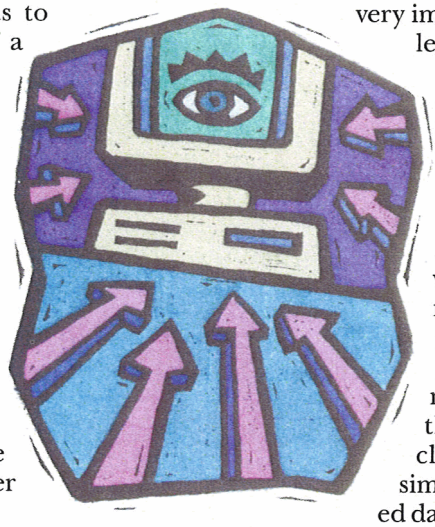
GEORGE SCHUSSEL

ADVANTAGES TO THE CLIENT/SERVER ARCHITECTURE

Client/Server Today welcomes you to the inaugural run of Convergent View. Columnist George Schussel, a seasoned consultant, analyst, and lecturer, will be covering key industry trends and technologies. He will be choosing issues that are important in the near future—issues covered from the point of view of technology's business/management implications. In addition, Schussel is interested in the companies that provide the technology, and he will comment on those he visits.

I have been a fan of client/server as a computing architecture since the late 1980s. The fact that it's taken several years for the engineering that allows us to realize the full benefit of a client/server architecture to evolve should not surprise us too much: Major architectural shifts in computing usually take a decade to reach maturity.

One major advantage of the client/server architecture is the great diversity that it allows for client-side, server-side, and connectivity solutions. That great diversity, however, enormously complicated the development of client/server software.



In the beginning (about 1987), the idea of a relational database using the SQL language and running under Unix or VMS was most likely to be your server. The client side, however, was more problematic, since the choices included workstations (too big, too expensive, not enough of them), ASCII terminals (are you kidding?), and DOS personal computers (no standards, too much diversity, too difficult).

The arrival of Windows changed the client/server computing environment. As Windows became the desktop standard, it quickly reduced the interface choices that users had to learn and, more importantly from an industry development point of view, that developers had to write to.

A GUI makes the use of applications much easier, but the development process is actually tougher for the software vendor than building to a character interface. The Windows API has about 600 calls, for example, more commands than most procedural languages. Migrating an application from Windows to the Mac or OS/2, then, is a far from trivial matter. Since Microsoft's Windows became the dominant client, developers could now focus on that platform and deal with the others as secondary marketplaces.

Emergence of this dominance of Windows is what gave client/server computing an important, de facto standard. Client/server computing's "vanilla" architecture, then, has become one or more relational DBMS servers talking to multiple client applications running under Windows.

The GUI client offers far more than a pretty, easy-to-learn interface (although that's very important). Beyond the ease of learning and uniformity that it brings to applications, a GUI provides tremendous benefits in application integration, using features like cut and paste and OLE. OLE-like object support is, in fact, changing fundamentally the way software is built (but more on that in a later column).

The client-side GUI is the most compelling reason for the coming dominance of client/server. The reason is simply this: If you want integrated database computing with a GUI front end, there is no other technology choice but client/server. While there are other compelling reasons to choose a client/server architecture, the costs involved would make me seriously recommend reconsideration if a GUI client isn't part of your environment.

Next month, we'll investigate in more depth some of the other compelling advantages of client/server architecture.

George Schussel has been a CIO, consultant, industry analyst, writer, and lecturer on computer topics for 30 years. He is the founder and chairman of Digital Consulting Inc. (DCI) in Andover, Mass., and chairman of the Database & Client/Server World trade show. His new book, Rightsizing Information Systems, *coauthored with Steve Guengerich, was just published by SAMS Publishing. He can be reached at CompuServe 74407,2472.*