

Alternative Technologies

Enterprise Integrity: The Ultimate Server Vol. 2, No. 1

We've had servers around for a long time: terminal servers, database servers, file servers, network servers, and so on. The current brood includes application servers, Web servers, message brokers, event brokers and process engines, the ever-evolving database server, and XML servers, each with its special function. Surely it won't be long before the next server innovation appears. But, given the high cost of training, management, and integration, are we willing to buy and integrate more and more types of servers? I think not.

Each of these special servers has evolved since its beginnings. Application servers once were simply server platforms that hosted applications built as services. Application servers today can be differentiated based on which services are provided: transaction management, load balancing, service migration, state management, object persistence, memory management, types of components supported, and more. Web servers were once simply a means to provide HTTP/IOP connections and to serve up HTML pages. Web servers today can be differentiated based on which services are provided: load balancing, connection pooling, connection migration, state management, connection persistence, types of Web pages and applets, and so on. Message broker functionality has also improved. Once they simply provided predetermined routing. Today they provide much more intelligent services such as directory, message transformation, rule-based routing, content-based routing, and load balancing, making them useful across a much broader range of applications. Indeed, Gartner Group now talks of "integration brokers."

Of all the servers that provide new services, database servers have changed the most. Although the traditional database server is often thought of as little more than a data store, database servers have always provided some form of connection management, transaction management, and scheduling and load balancing of resources. Today's database servers provide forms of connection pooling, connection and session migration, state management, load balancing, XML and HTML, HTTP/IOP connections, applet and component invocation, event and message, queuing, asynchronous replication (of data, objects, and messages), and many more types of services. Vendors of all types of servers seem not to have noticed that today's database servers have evolved into a much more than data stores.

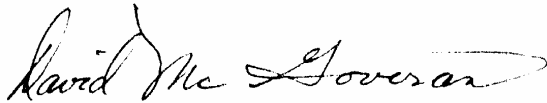
Curiously, opportunity blindness seems to have affected database server vendors as well as others. The result is that services are often neither as general nor as flexible as they might be. Nonetheless, the various types of servers are rapidly evolving to support a broad and overlapping suite of *fully distributed, replicated* services. Just suppose that some server vendor gets it: we could end up with a single code line that can be implemented (i.e., configured) as an application server, a

Web server, a database server, a message broker, or an event broker. What a concept! A real universal server. I propose that this universal server be called an *integration server*, despite that fact that some vendors are already using the term in other, inconsistent ways.

An integration server would greatly simplify the physical architecture of integration (as well as many other IT agendas). Think about it. Instead of needing to determine what type of servers to purchase and install, we could focus on capacity planning. Instead of being concerned about routing between servers, we could concentrate on establishing the sequence of services that are needed. (That sounds like a business process, doesn't it?) And most important, the work of integration would become more standardized. That means lower cost of ownership due to more efficient maintenance, training, development, deployment, and use.

I have no doubt that the suite of services provided by database servers, application servers, Web servers, message brokers, and event brokers is converging. My only concern is that these services become more general, distributed, and replicated across multiple platforms rather than specialized versions for each type of server. And, of course, the interfaces need to be standardized. Competing integration server vendors will need to differentiate their service (that's right, *service* not *server*) products by RASS characteristics.

When we can finally buy an integration server, maybe we can understand and achieve the goals of EAI just a bit more easily. Along with integration agents, integration servers will be the infrastructure supporting enterprise (application) integration. In the meantime, your EAI architecture can reflect the idea. The obvious integrity of such an architecture can only improve the integrity of your enterprise.

A handwritten signature in black ink that reads "David Mc Govern". The signature is fluid and cursive, with a long horizontal flourish extending to the right.