E-Business Architecture Design Issues

If you haven’t been asked to implement an e-business system yet, chances are you will be—soon. Beyond providing customers with ordering convenience and inventory selection that brick-and-mortar businesses can rarely match, e-businesses also offer your business operating efficiencies and automation opportunities. To maximize these benefits, however, you must expand the client-server model in favor of an approach that fully leverages the power of Web-based systems.

Traditional client-server-based applications rely heavily on the type of hardware or software used, and call for the client to be very aware of the server and vice versa. This approach is inherently incompatible with the open nature of the Internet—to maximize the benefits of e-business solutions you must leverage the open standards and network openness.

E-business solutions are not just about technology—they use Internet technologies and innovative business processes to develop applications that extend beyond the traditional boundaries of time, space, departmental, organizational, and territorial borders. These solutions add value by cutting bottom-line expenses through a lower cost of ownership, higher efficiencies, reduced transaction costs, and process improvements. They also fuel top-line growth by helping companies move faster, embrace new markets, explore new distribution channels, and inspire creativity in their organizations. While e-business applications share many aspects with distributed client-server systems, they also contain several subtle but important differences—shown in Table 1—that necessitate the introduction of a new thin-client architecture to replace the conventional 2- and 3-tier architectures used in client-server applications.

We suggest that in determining how best to develop an architecture for e-business applications, you think beyond conventional client-server architectures and move into the realm of thin-client architectures that are built to leverage the true potential of the Internet and its supporting technologies.

E-BUSINESS ARCHITECTURE DESIGN ISSUES

E-business architectures leverage Web technologies to implement mission-critical e-business applications. These architectures use small-footprint clients to access services, provided by resource managers that can be accessed across a strong and reliable network. These clients can be browsers running on personal computers, network devices, personal digital assistants, cell phones, and other pervasive computing devices.

The e-business architecture is more than just a collection of technologies and products. It consists of several architectural models and is much like a city plan in that it defines a blueprint that will meet the current and future needs of a diverse user population, and will adapt to changing business and technology requirements. The key elements that help influence an e-business architecture include:

- the organization’s overall e-business strategy;
- business drivers such as time to market, one-to-one customer service, and others;
• the current IT environment;
• IT vision, objectives, and strategies;
• organizational constraints such as staff, budgets, risk tolerance, and others; and
• new and emerging technologies.

In addition, e-business architectures should be built to meet a key set of operational requirements. In this context, we define an operational requirement as a quality requirement or constraint that an IT system must satisfy. E-business application operational requirements often include:

• the scalability to support many users,
• the capacity to handle volume loads that can vary dramatically over time,
• acceptable transaction response times under most conditions,
• a secure computing environment that protects sensitive information from unauthorized access, and
• continuous system availability.

Several methods and techniques—such as faster machines or replicated machines, caching, and mirroring—let you implement these operational requirements. While most of these techniques can be applied to an application after it has been designed, doing so provides less than optimal results. An application cannot be retrofitted to meet operational requirements such as security, scalability, performance, and availability—it must be engineered specifically to address them.

**ENGINEERING E-BUSINESS ARCHITECTURES**

The process of developing a well-engineered architecture starts with gaining a clear understanding of the current environment and a clear vision for the future. Next, you must create an effective plan that will let you move from the current environment toward the future state by taking into account the resources available for the effort and developing a good understanding of the constraints under which these architectures must be developed.

Based on these factors, we recommend that architects building e-business architectures use the following guiding principles:

• Develop architectures based on open standards and technologies. Doing so will provide you with the flexibility to move the application to a different hardware platform or to move the application to a different hardware environment.
platform, or segment the application’s workload. To build in such flexibility, you must understand the services model and choose products based on open technologies that provide these services.

- Segment your application architecture into distinct layers, then make sure the layers are tightly coupled internally and loosely coupled to the model’s other layers. This approach will provide the flexibility to determine the appropriate placement for these layers on the physical tiers of the architecture.

- Make sure the technologies used in these architectures are proven and stable—which means, in most cases, that you should avoid working with beta and prerelease software.

- Leverage investments in existing systems. The true power of e-business applications comes from developing innovative processes by integrating with existing systems and databases, not by reinventing them.

- In a tight labor market, finding people with good technical skills is both hard and expensive. To significantly enhance the odds of implementing a successful e-business architecture, select a set of technologies and products that will allow you to leverage skills that exist within your organization.

The services model and layered-application model can provide a helpful framework for implementing these principles.

**Logical-services model**

This model provides a layout of the basic architectural components that deliver the services required to support e-business applications. These services satisfy the functional and operational requirements of the applications and the high-level interactions among them. As Figure 1 shows, the logical-services model that supports the IT infrastructure provides several essential services for business support, presentation, application, resource management, and system management.

**Layered-application model**

As you analyze and design specific e-business applications, you will create business domains and draft the corresponding design documents that describe detailed software component models. These models operate within the context of the high-level models shown in Figure 1. The layered-application model, shown in Figure 2, defines the seven components that make up the services framework, as follows.
Client. In the Web environment, clients will play a more diverse role than in a client-server model. In this framework, the client uses a “thin” presentation mechanism such as a browser-based front end to accept and process user input data. This layer interacts with visual components to handle presentation-related tasks, and with nonvisual components to handle the interface with back-end applications. The client in this architecture makes calls to interfaces on nonvisual components that correspond to business-level interfaces, such as get_stock_price or calculate_order_value. This type of structure isolates the client from the details of how the back-end applications are accessed and allows an organization to reuse the back-end components across multiple client types. This component could be implemented using HTML Web forms, DHTML, and JavaScript.

Application service proxy. This component exposes interfaces that correspond to business-level calls the clients need and ensures these calls are processed by the appropriate application service. The application service proxy encapsulates the details of the distributed-communication mechanism used to shunt data from clients to back-end services. It also handles any communication errors or problems that arise when communicating with back-end applications. This component is typically implemented using URL extensions within a browser.

Formatter. A formatter component removes device- and client-specific information from incoming data, and replaces the information when data returns to the client. The formatter also isolates the client-side components from the proprietary technologies and protocols used in back-end components. It could also be used, via compres-

Figure 2. The layered-application model defines in detail the seven components that make up the Web framework. Paired 1’s next to an arrow show a one-to-one relationship; a 1 and an asterisk show a one-to-many relationship.
sion, to reduce the amount of data sent across the network or, via encryption, to secure such data from unauthorized access. This component is implemented using dynamic-page-generation capabilities provided by Web applications servers such as Java Server Pages or Active Server Pages.

**Application service provider.** This component exposes interfaces that correspond to business abstractions within the problem domain. The application service provider encapsulates business logic and structures it into components or objects that can be reused across multiple applications. It handles the logical unit of work and manages each transaction, using the transaction router component to access data from one or more back-end resource managers. This component is implemented using component technologies such as JavaBeans, Java Servlets, and COM components.

**Transaction router.** This component plays the role of an intelligent traffic cop, translating the business-level API call issued by the application service provider into a call implemented using a specific product and protocol. The transaction router handles error conditions and encapsulates the product-specific communication details from the architecture’s other components. This component is implemented using a programming language such as Java or C++.

**Translator.** This component often plays the role of a proxy for individual transactions within back-end legacy applications. The translator also shields other components from the proprietary technologies used by back-end products. It could, for example, convert between text formats like EBCDIC and ASCII, or hide the details of stored procedure calls.

**Resource manager.** This component contains the data and information required to provide the application’s business functionality. The resource manager can be made up of legacy-application transactions, or of procedures stored in databases or any other application type.

Choosing which products and systems you use for these components will be one of your most important design decisions. The “Shopping for E-Business Infrastructure Components” sidebar provides guidelines for determining which products will work best for your e-business project.

When planning your end-to-end e-business architecture, focus on the appropriate user interface, application, middleware, database design, tools, and deployment strategy. Doing so will provide you with the analytical functions needed to answer your users’ questions. Before you can make logical choices in these areas, however, you must fully understand your users’ specific information needs; the questions they will ask, and the decisions they must make. The true value of an end-to-end e-business architecture lies in its ability to provide integration with back-end systems and databases.
Selling E-Business to Upper Management

Athough the clients who have consulted us about implementing e-businesses come from a range of industries, our work with them has revealed several common business characteristics that define a prime e-business candidate. A company that can best benefit from an e-business implementation has

- large manual transaction volumes,
- complex, information-rich processes that require easy access to large amounts of data, and
- large numbers of employees who are often distributed across several sites or who work at remote locations.

To keep your e-business project grounded in reality, it’s best to take an incremental approach. Implement in intervals, then reassess your progress at each milestone, obtain upper management’s continued buy-in, then proceed to the next stage.

Beware of simply throwing more people at a project. Because software development requires that the project team manipulate and connect many mental models that are difficult to communicate between team members, larger groups work less efficiently than smaller ones. Thus, having fewer people on a project team actually reduces communication effort and distortion.

When it comes to making the case for proper funding and deadlines, hard numbers can be your best friend. Arm yourself with explicit details about your Web investment’s business case, measurement framework, strategy, and assessment. You’ll also need models that tell you how product size relates to the effort and schedule required to build the product. The following sites can help:

- http://www.ibm.com/e-business/ecommmerce/roi.html: The references and case studies found here can help you define and clarify business methodology to better understand the financial impact of becoming an e-business.
- http://sunset.usc.edu/research/WinWin/index.html#downloads: Barry Boehm, who originally defined the Constructive Cost Model in 1981, shares the work he’s done with his collaborators during the 1990s to modernize Cocomo.
- http://www.computer.org/tse/ts1997/e0485abs.htm: In this document, the authors propose use of an analytical model for cost estimation.
- http://xanadu.bmth.ac.uk/staff/kphalp/students/bsi/predict/tsld002.htm: The slides found here depict new approaches to software cost estimation, including the application of different techniques to Boehm’s Cocomo, such as function points and estimation by analogy.

Ultimately, you will achieve the most benefit from e-business solutions by engineering them and proactively planning for better performance, scalability, security, and other operational requirements. A thin-client and server-centric architecture, implemented using the techniques we’ve described, can give you the flexibility to achieve and maintain acceptable performance as your e-business grows and analytical complexity increases.

Srinivas Koushik is an IBM distinguished engineer and director of the E-Business Architecture Center of Excellence at IBM Global Services. Contact him at skoushik@us.ibm.com.

Pete Joodi is the senior member of the E-Business Architecture Center of Excellence at IBM Global Services. Contact him at pjoodi@us.ibm.com.