CHAPTER 6 *Functional Architecture*

Architecture . . . the adaptation of form to resist force.
—John Ruskin[1]

What Is Architecture?

The architecture of a system defines its basic components and important concepts and describes the relationships among them. There are many different ways to approach systems for Internet commerce, ranging from the simple to the complex. In part, the architecture depends on the nature of the business, and the system architecture developed for a consumer retail system might be very different from that for a publishing system. We believe that many design ideas span a wide range of commercial requirements, and that the similarities among systems for Internet commerce are much greater than the differences. This chapter describes a core architecture for Internet commerce systems, which can be adapted for many applications.

Why should we have a general architecture? Why not simply build the systems focused on a single application? For us, a practical answer is that, in our experience building systems for Internet commerce for several years, reusing the architecture and design work where possible is best for the customers. More important, however, is that as businesses refine and evolve their goals for Internet commerce, their systems need to evolve as well. That evolution may go well beyond the original requirements for the system, so the flexibility of the architecture is critically important in making that growth possible. For example, a software store may begin by taking orders over

1. John Ruskin, *Val d'Arno* (1874).

the Internet and sending out boxes with manuals and CD-ROMs. Later, it may want to deliver software over the network as well. If the original system does not handle online delivery, the store may find itself facing significant development or upgrade costs to add this capability. Note that the original system might not have implemented the online delivery subsystem, but it should be straightforward to add it if the original architecture and design were done with that idea in mind.

In this chapter, we describe the kinds of thinking that go into creating an architecture, explore some of those areas in more depth, and present some examples of practical architectures for Internet commerce.

Core Architectural Ideas

Architectures for commerce systems may look very different, but they all have to address the same issues. These issues must be understood no matter what approach is taken. In some cases, these common questions are considered explicitly during the design phase; in other cases, the questions and their answers are reflected in assumptions about various components in the architecture. In this section, we examine some of the primary elements that go into a commerce architecture, and we hope to make most of the issues explicit rather than leaving them as assumptions.

One word of caution: it may sometimes seem that what we describe in the architecture is so obvious that it need not be written down, or that we are drawing unnecessary distinctions. In our experience, leaving the obvious as implicit can often lead to later confusion and misunderstanding, precisely because everyone thought it was obvious but had different ideas of what was obvious. If we are to be successful at designing and building computer systems, we must be very precise, not only in describing the computational steps but also in our understanding and description of what we are trying to do. The processes of doing business may seem natural because they are so familiar to us, and because people can handle many unusual situations easily and effectively. When we design computer systems to manage some of those processes, we must be especially careful because computers cannot figure out how to keep a customer happy when something unexpected goes wrong.

Understanding of Roles

Two of the most basic questions for any computer system are “Who uses it?” and “What do they do with it?” For some programs, there are a few kinds of users who share similar goals. For example, novices and experts both use word processors with the same goal—producing documents. Internet commerce systems are more complicated: their users include the buyers of goods and services, the sellers of goods and services, and the people who operate all of the machinery.

Understanding the various roles and kinds of users for a system helps us to focus our attention on making sure that all users can use the system effectively to accomplish their goals, whether that is making a purchase or creating accounting reports.

Decomposition of Functions

A second important part of the system architecture is the way it decomposes the system into functional units. The specification of these functional units and the interfaces between them defines the architecture of the system. One of the differences between architectures is often the way that they group functions into units. Are all of the components integrated in a single system? Are the components distributed across multiple systems? What are the interfaces between functional units?

Linking Content to Transactions

The first two considerations for the system architecture, roles and decomposition, apply to the design of almost any computer system. A third part of the system architecture of an Internet commerce system is the way that content, such as a catalog, is linked to the transaction processing. In a paper-based system, the buyer transcribes item numbers and quantities onto an order form or requisition. Obviously, we would like to do this electronically.

There are several key issues in this linkage.

- How the user makes the transition

  In many cases, the user sees a **Click here to buy** button or can add items to a shopping cart for later purchase. The transition to the transaction takes place either at the **Buy now** point or at **Checkout** for the shopping cart.

- How the information is verified

  Depending on the underlying technology, it may be necessary for the transaction system to verify that the purchase information, such as price, item identification, and so on, was not modified when it was sent over the network. Because (as we shall discuss in more detail later) the Web uses a stateless protocol, a commerce system built on top of the Web must manage its own state. If that state is carried by the client in some way, the server must be able to ensure that it was not modified in transit.

- How the information matches up

  Some Internet commerce systems include a real-time inventory check to assure customers that items are in stock. If the system indicates that an item is in stock, however, how long is that indication valid? If the customer puts the item in a shopping cart for purchase at some later time, does the system promise that the item will be available when the purchase occurs? If the system does make such a promise, how long is it valid? What if the customer never returns to the site to buy the items in the shopping cart?

The answers to these questions help make design decisions for the system. Because different answers can lead to very different designs, it is important to think through the issues early in the design process.

Trust Models

In any distributed system, different components trust each other to a greater or lesser extent. Some components may trust others completely for all kinds of access (for example, both reading and writing data elements), whereas other components may disallow any remote access to their data. The specification of these relationships is called the *trust model* for the system. Any system has at least an implicit trust model, but specifying a trust model explicitly helps us to understand the details of the relationships between components when we need to analyze the security of the system.

Roles

Many different people interact with an Internet commerce system, and they need to do different things. Buyers require one set of operations; catalog designers, customer service representatives, and system operators each have their own sets as well. Even though these latter groups may all work for the merchant, they have different tasks to perform. For some businesses, especially smaller ones, the same person may perform all these tasks. Larger businesses have different people fulfilling different roles. Considering roles separately enables us to satisfy the requirements of businesses of all sizes, as well as making it possible to design a system that allows a smaller business to grow smoothly without having to reconsider what people do at each stage.

Speaking in terms of roles also helps to avoid confusion. For example, simply referring to the customer does not distinguish the cases when one person is selecting a product to be purchased and another arranges payment. By defining the operations required by a particular role, we can ensure that everything needed by the role is present in the system, rather than relying on the ability of one person to act in multiple roles.

It is important to keep in mind that there might be individuals playing many different roles in some organizations, and that there might be many individuals playing the same role in a larger organization. For example, larger organizations commonly have many people in the customer service role.

Next we describe some of the primary roles for both the buyer and the seller.

Customer Roles

In any commercial transaction, there is a buyer and there is a seller. We use many different words for the buyer: *customer*, *consumer*, *purchasing agent*, and so on. On the Internet, we sometimes refer to the buyer as a *client* or *browser* as well, referring more

to the software than to the person. But there are some subtle differences among these words, and the distinctions reflect different roles on the buyer side. In some cases, such as a consumer purchase, the same person plays all of the roles without even thinking about the differences. Businesses, however, often make purchases in different ways, so it is useful to consider the various roles.

- Specifier—this person selects what is to be purchased.
- Approver—this person approves a purchase recommended by the specifier.
- Buyer—this person negotiates the terms and conditions of a purchase and arranges for payment.
- Recipient—this person receives the delivered goods or services.

In addition, we can distinguish different kinds of buyers based on their relationships with the seller. An *anonymous buyer* (sometimes referred to as a *walk-in customer*) has no prior relationship with the seller and may not ever create one beyond making a simple purchase. A *member customer* is one who repeatedly buys from a seller and has established some kind of relationship, which we will call *membership* here. A member may sign up because an account is convenient, because it offers some special benefits, or because there is a business relationship that is reflected in the membership.

Membership accounts give rise to another role, the *member administrator.* A person acting in this role may modify or update any stored profile information about the member. If the membership encompasses several individual accounts, such as for different members of a family or multiple purchasing agents for a business, the member administrator may also be able to set limits on the use of the individual accounts. These limits might be on the kinds of items that can be purchased, the amounts that can be spent, the time of day for purchases, and so on.

In practice, of course, a single individual may fulfill more than one role. For example, a consumer buying a sweater selects one in a store, pays for it, and takes it home. In this case, the consumer plays all three roles, and we generally make no distinctions among them. In contrast, consider the electronic components example described in Chapter 2. The specifying engineer determines which part will be purchased, a purchasing agent negotiates the payment terms, and the manufacturing group receives the components for inclusion in the final product.

What this breakdown tells us is that a general-purpose Internet commerce system needs a way for different people to handle different parts of a transaction, but it should also be very simple for a single person to handle all of them. Consumers do not expect to change roles explicitly at every stage, but they do expect to have a quick and easy process for buying. Companies, on the other hand, that make distinctions in the various roles want to be able to hand off the transaction from one role to another smoothly and efficiently. Even consumer systems get value from a good implementation of

roles. A wish list is something a specifier would use, and most systems permit orders to be shipped to different addresses.

Business Roles

On the other side of a transaction is the seller. There are many roles in an Internet commerce system for sellers. Smaller businesses, and even larger ones beginning with small-scale efforts in Internet commerce, may have just a few people playing all the roles—so much so that the different roles enumerated here may seem overly complicated. Thinking about the roles early, however, makes it possible for an Internet business to grow more smoothly as more people join the team and the roles become more distinct in reality. For the seller, there are two main groups of roles: the business and content creation team and the operations team. The most important business roles are as follows.

- Business manager

  The business manager is responsible for the business approach on the Internet, creating and operating the Internet presence for the business, deciding which products and services are to be sold online, determining pricing, and establishing the key business relationships needed to make the venture successful. (We will discuss implementation strategies for these kinds of operations in Chapter 7.) This is primarily a business role, with particular attention paid to the success of the online business and the bottom line.

- Internet commerce architect

  The Internet commerce architect is generally a systems analyst who turns the business requirements into a system design that incorporates the creation and management of content (such as catalogs) and the transaction processing, fulfillment, and technical aspects of customer service. In short, the architect fills in the next level of detail for the commerce value chain.

- Content designer

  The content designer is responsible for the look and feel of an Internet commerce system, including graphic design, page layout, user experience, and so on.

- Content author

  The content author creates or adapts product information in a form that can be used for Internet commerce, working within the design laid out by the content designer.

- Implementor

  The implementor is responsible for creating any programs or software extensions needed to make the Internet commerce system work. For example, an implementor might write the software that takes product information from a database and dynamically renders it into a Web page.

- Database administrator

  If a database of product information is used, the database administrator (DBA) manages the creation and operation of the database to ensure its correctness, integrity, and performance.

- Sales and marketing team

  The sales and marketing team is responsible for focused efforts in promoting Internet-based commerce for the business.

- Customer service representative

  Customer service representatives for the business answer questions about products, assist buyers with registration or the purchasing process, respond to inquiries about order status and problems after the sale, and handle product returns and payment disputes. A business may have different people specialized in different areas of this role.

Of course, a particular organization may have more than one person in one or more of these roles, or one person may act in many of them. Some of the decisions may be determined by software purchased from a particular vendor, in which case many people on the commerce team described previously must select which product to buy and how it fits in with their plans for Internet commerce. Some members of this team may be outside consultants, depending on the skills and availability of the organization's staff.

The operations team installs and operates the Internet commerce system, making sure that it is running and available for customers. Some specific approaches to system operation are discussed in Chapter 7. The roles include the following.

- Operations manager

  The operations manager is responsible for managing all service activities for the Internet commerce system.

- System supervisor

  The system supervisor manages the system staff.

- System administrator

  The system administrator is responsible for the technical operations of the computer systems and networks.

- Security officer

  The security officer ensures that appropriate security measures have been taken in the design and implementation of the Internet commerce system.

- Fulfillment agent

  The fulfillment agent is responsible for shipping and handling of physical goods or delivery of services. In the case of digital goods, the fulfillment agent is responsible for overseeing the operation (and staff, if any) of the fulfillment system.

- Accountant

  The accountant is responsible for ensuring that the proper accounting procedures have been followed for Internet-based transactions, managing the relevant business records, creating reports on the transactions handled by the system, and other accounting functions.

Roles and Reality

The roles we have described are probably not exactly the roles found at any particular business, and it is unlikely that there is any one-to-one correspondence between these roles and people doing real work. Thinking about roles instead of people, however, enables us to ensure that all of the work gets done and that we are not missing an important function as we design a system and put together a team to operate it. Making these distinctions also helps if some of the work will be outsourced. As we shall see in Chapter 7, there are many approaches to implementing a system, and sometimes it makes sense to outsource all or part of the operation of an Internet commerce system. Some of the roles here—the operational ones, for example—can be outsourced relatively easily. Others, such as deciding which products are to go into the online catalog, are business decisions that cannot be handed off to others.

Components

Another important aspect of the system architecture is the set of components that comprise the system. For Internet commerce, we frequently try to take advantage of general-purpose Internet applications, for three reasons.

1. If general-purpose applications can be used, we need not build them again.
2. General-purpose applications are widely distributed, so we need not create the distribution channels to put a specialized tool in the hands of customers.
3. Customers are already familiar with the application, so they do not need to learn how to use a specialized tool.

In the next two subsections, we introduce the basic components used for Internet commerce systems. We will present a more detailed technical discussion of these components in Parts Two and Three of this book. Although our focus is on using general-purpose tools, such as the browsers and servers of the World Wide Web, there are, of course, times when it is appropriate to create and distribute a specialized tool for commerce. Even so, we think the components described here are a good starting point for designing such tools.

Customer Components and Clients

For customers, the primary tool for using the World Wide Web is a *browser,* sometimes called a Web *client.* We discuss browsers and other Web clients in Chapter 8. The system architecture is clearly influenced by the basic structure of the Web, and in particular by the capabilities of browsers. As we shall see, one of the important questions in deciding exactly how to structure a system is "What browsers do the customers have, and what are their capabilities?"

Some companies have also designed specialized client applications for commerce, particularly for payment. These applications, often called *client wallets,* are designed to implement one or more payment methods that require additional processing, such as cryptographic operations, on the client computer. Client wallets may also be used to keep track of what transactions have been made, to check on order status, or to manage other information related to transactions. The main problem with client wallets is that hardly any customers have them. Over the past several years, companies have not succeeded in distributing them on a large scale, so they are essentially irrelevant for most Internet commerce systems. An important lesson from these attempts is that new client software is a barrier for customers, and they will likely look for alternatives rather than proceed with installing new software. We look at different payment systems, including their requirements for client software, in Chapter 15.

Sometimes customer components can reside on third-party systems. For example, server-side wallets are Web sites at which consumers can register payment credentials and release them to sellers without having any special client software. Another example is a buyer home community system that provides authentication and customer service facilities in a federated commerce system, as we shall see later in this chapter. Such systems are not yet common, but many companies continue to build them.

Seller Components and Servers

On the other side of a transaction is the seller, whom we might also call the *merchant* or *vendor.* The seller provides all of the components of the commerce value chain, from content to customer service. In practice, a seller may provide some of the stages in the value chain directly and contract with others to provide the rest. Different sellers may make different decisions about which stages to provide directly, and these decisions may even change over time. Again, therefore, we separate the stages of the value chain in the general architecture so that different components can be handled differently.

Some components of the value chain are more easily outsourced than others. Content, for example, is a presentation of the actual products or services offered for sale by a business. Although a company may look outside for creative presentation ideas or development of the actual content, what the products are and how they are sold are the foundation of the business. Payment services, on the other hand, are very important to

the business, but the details of the processing can be hidden as long as the results are correct. Here are some of the components.

- Content management system

  This can refer to the seller's catalog or to the entire corporate Web presence. Content management systems permit the creation and management of dynamic and continually updated content.

- Transaction processing system

  The seller's transaction processing system keeps track of all information related to transactions: what was ordered, who ordered it, how much it cost, the status of payment, the status of fulfillment, and so on.

- Payment processors

  Payment processors manage the movement of money or other payment instruments in the system. For example, when a consumer pays with a credit card, the seller connects to a credit card payment processor to authorize the transaction (by checking for sufficient available credit) and, later, to settle the transaction.

- Fulfillment systems

  Companies operating mail-order businesses often contract with a fulfillment company to handle packing and shipping orders. A business taking orders over the Internet for tangible goods might do the same. Indeed, a business selling digital goods over the Internet might even work with a fulfillment company to operate the servers used to deliver the online products. Or, in both cases, a business might choose to manage the fulfillment process in-house.

One logical grouping of these functions results in what we will call the *front office* and the *back office.* The front office is concerned with marketing and selling goods and services. Content and presentation are very important, and the focus is on attracting the customer to buy the product or service. The back office is concerned with managing the details of the transaction, from placing the order to payment to fulfillment. Proper handling of the transaction is important, such as ensuring that the relevant information is delivered to the right places and that the payment is collected correctly.

Examples of System Architecture

As we have suggested previously, different answers to different issues can result in very different system architectures. In this section, we look at four different architectures and discuss how they are constructed. The four architectures are a Web server with an order form, an approach to distributed transactions originally created at Open Market, a large-scale federated commerce system, and an approach to business-to-

business commerce originally developed by the Open Buying on the Internet (OBI) Consortium. There are, of course, many other approaches to Internet commerce; we chose these to illustrate many of the points discussed in this chapter.

For analysis of architecture, we have found it convenient to consider four primary components of Internet commerce systems.

1. Client

   The client is a computer system, typically a personal computer, connected to the Internet either directly, through an Internet service provider (ISP), or indirectly, through a corporate network. The buyer uses the client computer for browsing and purchasing.

2. Merchant

   The merchant is the computer system or systems that contain the seller's electronic catalog and, in the case of online goods, products for over-the-Net fulfillment.

3. Transaction system

   The transaction system is the computer system or systems that process a particular order and that are responsible for payment, record keeping, and other business aspects of the transaction.

4. Payment gateway

   The payment gateway is the computer system that routes payment instructions into existing financial networks such as for credit card authorization and settlement.

Various architectures use these four components in different ways. In some systems, some of these components are combined into a single computer system, whereas in others these four system components are implemented by separate computer systems.

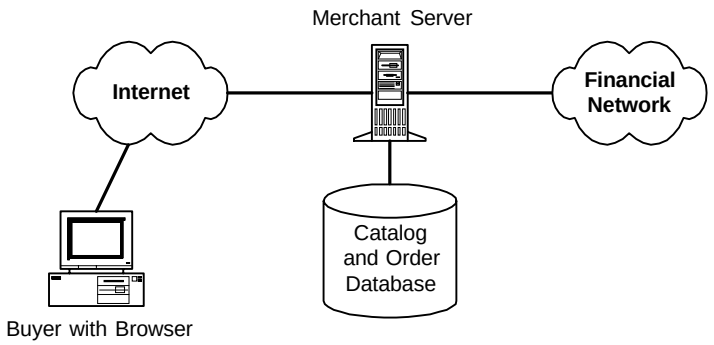

FIGURE 6-1. Merchant Server: Physical View

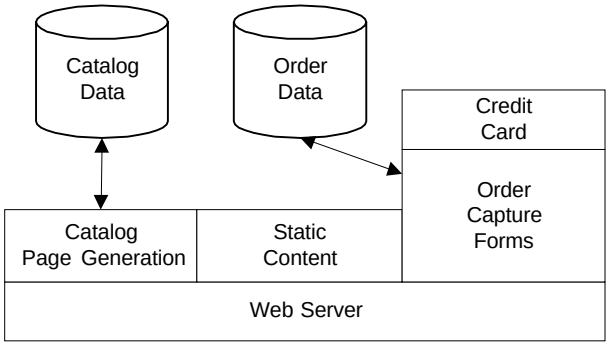

FIGURE 6-2. Merchant Server: Logical View

Once the designers of a commerce system have selected a gross division of function, there are still many decisions to be made at lower levels of functionality. For example, the order aggregation function, which permits the assembly of individual items into a complete order, can be implemented as part of the merchant, the transaction system, or the client.

Web Server with Order Form

Building a Web server with catalog pages and an order form is one of the simplest ways to construct an Internet commerce system. This approach is typically called a *merchant server.* A diagram of a representative system is shown in Figure 6-1, and a logical diagram of the structure of the merchant server is shown in Figure 6-2. Many of the technical details will become clearer as we discuss the technology in later chapters, but we will sketch the basic ideas here.

In this example, a single Web server provides both the catalog content and the order form. In other words, the merchant server and transaction server are combined into one system, and there is no explicit payment gateway. The catalog might consist of a set of Web pages describing items for sale, with embedded pictures, drawings, specifications, video or audio clips, and so on. The Web pages might be created as static pages using an HTML editor, or they might be created dynamically from a database of items and descriptive information. Next to each item is a button that the customer can click to buy the item or add it to a shopping cart for later checkout. When ready to buy the item (or items, if more than one is present in a shopping cart), the customer clicks a **Checkout** button that starts the payment part of the transaction.

Payment by credit card is by far the most common method used on the Internet today for consumer transactions (as we discuss in detail in Chapter 15). A simple order form might consist of a listing of the items being purchased and a set of fields for the cus-

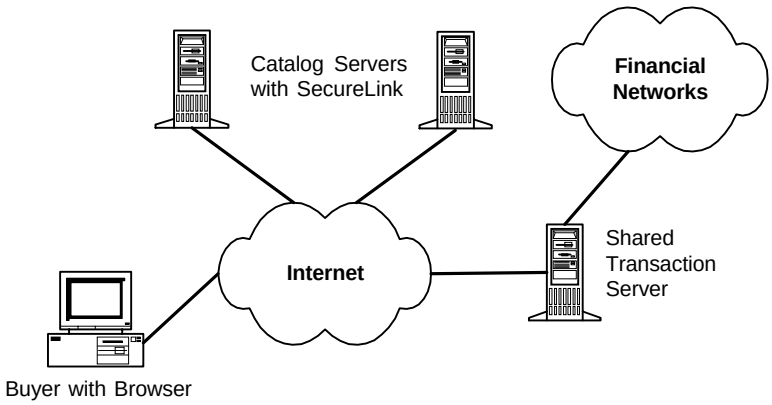

FIGURE 6-3. Open Market Distributed Commerce Architecture: Physical View

tomer to enter credit card payment information, including the card number, the expiration date, and the address for delivering the items, if they are physical goods. The form might also ask for the billing address, because some credit card systems use the billing address as part of the verification of the holder of the credit card.

It is possible, of course, that the Web server might use a different payment mechanism. In the simplest version of this model, the Web client has no special capabilities for commerce, so the commerce application does not require additional software for payment mechanisms. Credit cards, purchase orders, and other kinds of account-based payments may be used with such systems, thus taking advantage of the basic security capabilities common on the Web today.

This basic architecture may be appropriate and sufficient for some kinds of Internet commerce applications. Its primary virtue is its simplicity. On the other hand, it may be more difficult to expand it as the online business grows, or to incorporate new technologies and components as they become available.

Open Market Distributed Commerce Architecture

The core architectural idea of this architecture is to separate the management of content from the management of transactions through a technology called *SecureLink*. This idea permits multiple catalog servers to share the capacity of a single transaction engine and allows the content-oriented parts of the system to scale independently from the transaction-oriented parts of the system. Separation of these functions allows separate management of several system facilities—most notably, security. This approach also permits service organizations to provide transaction management services on an outsourced basis for other companies. Figure 6-3 shows the physical architec-

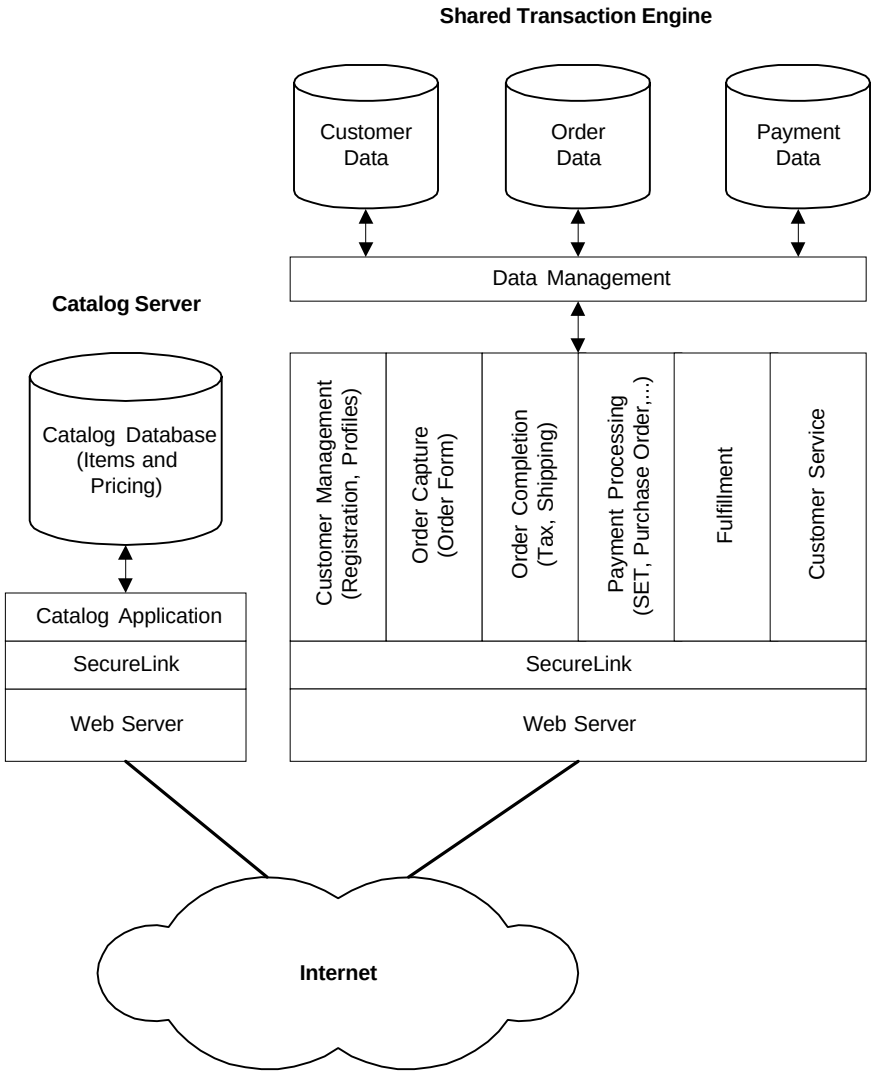

FIGURE 6-4. Open Market Distributed Commerce Architecture: Logical View

ture of this approach, and Figure 6-4 shows how functionality is distributed across different elements of the system. In this architecture, the transaction server is separated from the merchant server, and there may or may not be a separate payment gateway depending on which payment methods are supported.

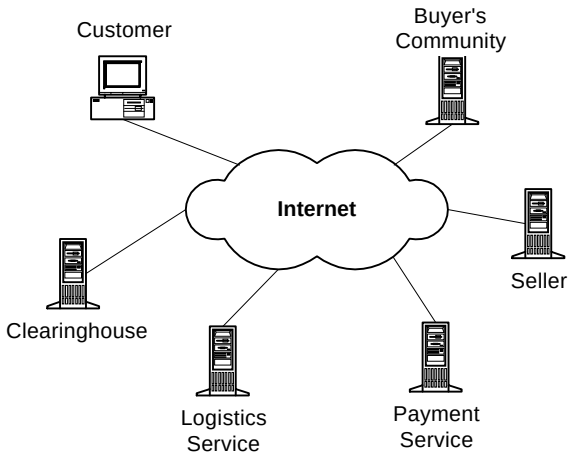

FIGURE 6-5. Federated Commerce System

Federated Commerce System

A federated commerce system (FCS) is a system made up of servers operated by different organizations and tied into an overall, perhaps global, commerce system by Web services and a collection of service agreements. In this system, consumers are members of communities of interest, which provide authentication services and customer service points of presence. When a consumer buys something, the necessary payment and shipping information is routed from her home community to the seller. The seller may also use some of the federated services for payment and other functions. At the conclusion of the transaction, a record is posted to the consumer's online statement held by her home community. From the consumer's point of view, only the home system is responsible for keeping online information. The essential architecture of this system is shown in Figure 6-5.

The federated commerce system includes

- Clearinghouse

  The FCS clearinghouse is responsible for tying the network together. It permits the seller to locate a buyer's home community and helps establish peer-to-peer connections among the various participants.

- Customers

  Customers browse for information, make purchases, and request customer service.

- Home communities

  Consumers join home communities on the basis of interests, services offered, or the degree of privacy protections offered.

- Sellers

  Sellers include merchants, service providers, and publishers.

- Payment provider

  The payment provider provides support for various payment mechanisms as a Web service. It uses a common interface, which helps make sellers more independent of the payment systems.

- Logistics provider

  The logistics provider handles shipping, returns, and other services.

We will return to this example in an extended form in Chapter 21.

Business-to-Business Purchasing

A closely related problem is managing purchases between businesses using the Internet. One architecture for such systems was proposed by the Open Buying on the Internet (OBI) Consortium, which was formed in 1996 to develop standards in this area. The consortium is a group of buy-side organizations, sell-side organizations, payment organizations, and technology companies that is addressing the problem of business-to-business commerce on the Internet. The core idea of OBI is to split the functionality of the commerce system between buy-side activities and sell-side activities so that each organization manages those functions logically connected to it.

The OBI design is based on a model of business commerce shown in Figure 6-6. In this model, the logical breakdown of activities is to place the customer database, requisitioner profiles, and approval processes on the buy side, and to place the catalog, order management, fulfillment, and payment activities on the sell side. This structure results in the architecture shown in Figure 6-7. The key idea in OBI relevant to our functional components is the splitting of the transaction server into its sell-side and buy-side parts.

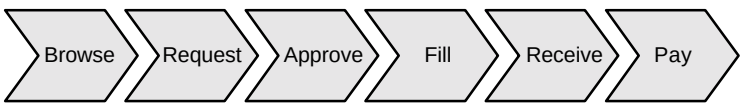

FIGURE 6-6. OBI Business-to-Business Purchasing Process

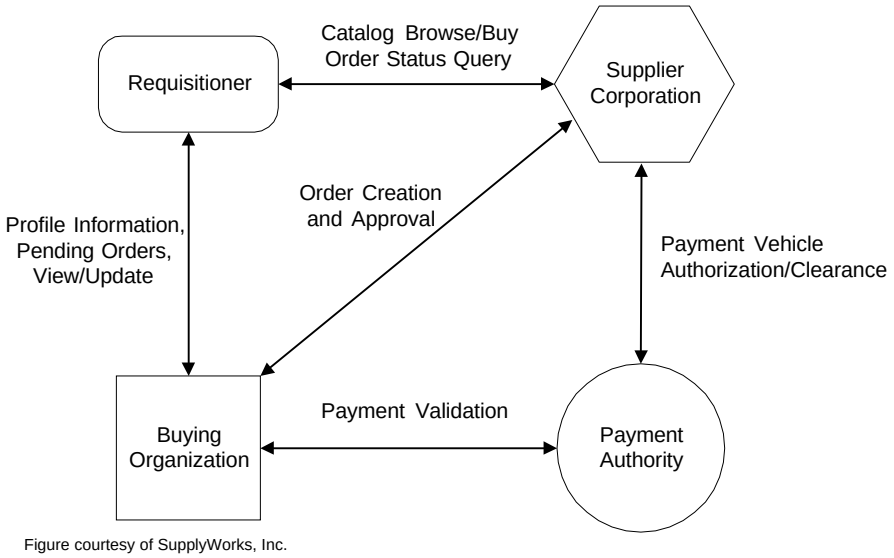

FIGURE 6-7. OBI Architecture

In order to make this architecture work, two elements of interoperability are needed between the buy-side and sell-side components: requisitioner authentication and order handling.

1. Requisitioner authentication

   Because the buy-side organization assumes the responsibility in the OBI model for managing the pool of requisitioners, the sell side must have a standardized means of authenticating prospective requisitioners as authorized by the buying organization. OBI uses public-key certificates for this purpose. When the requisitioner browses the supplier catalog, he presents a certificate signed by the buying organization to validate himself. This approach implies that at the time the trading relationship between the companies is set up, the supplier catalog must be configured to accept the certificates.

2. Order handling

   In OBI, the requisitioner builds up an order by interacting with the supplier catalog. That order is then sent in a standardized format called the *OBI order request* from the sell-side OBI server to the buy side. Once it is there, any necessary approval processes proceed. After the order is finalized, it is returned to the sell side as an OBI order for fulfillment.

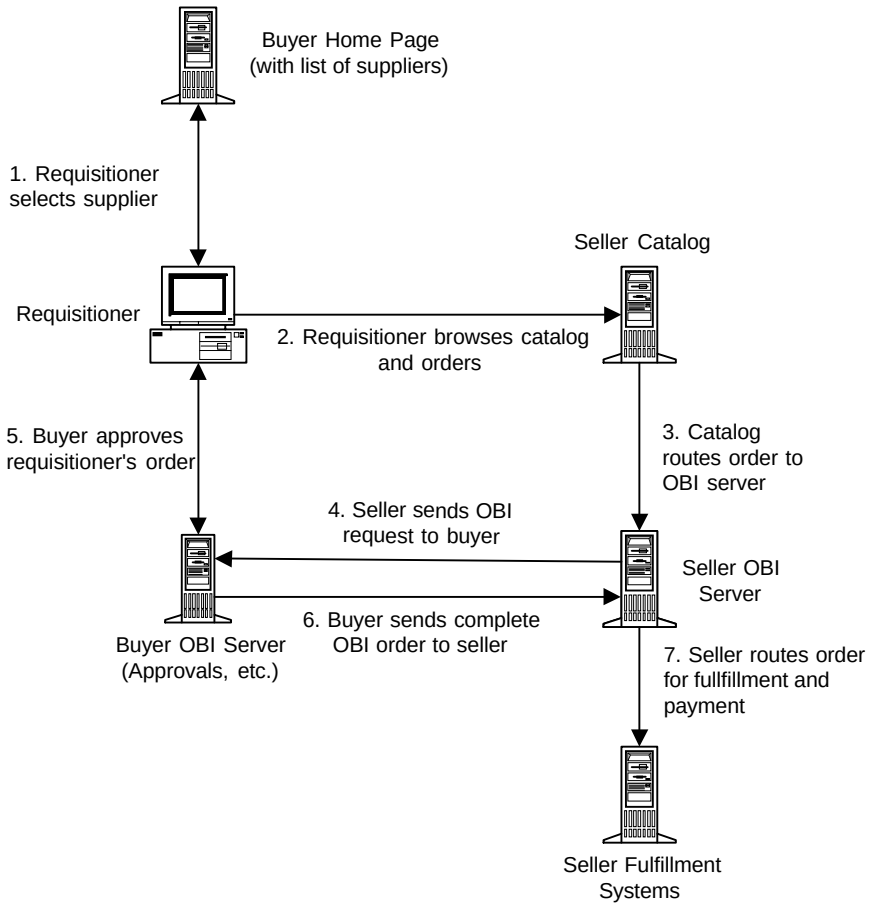

FIGURE 6-8. OBI Transaction Flow

The real benefits of the OBI choice of placing functionality can be seen only when there are multiple buy-side companies trading with multiple sell-side companies. When this happens, the buy side is able to manage its requisitioner database and approval system centrally, and it can use those systems seamlessly with multiple trading partners. Similarly, the selling organization can leverage a master catalog and order management system against multiple purchasers. In this ideal situation, information is not duplicated on either side.

Transaction Flow in the OBI Model

In this section, we walk through an OBI transaction. The descriptions in the following list correspond to the numbered arrows in Figure 6-8.

1. The requisitioner uses a Web browser to connect to the buying organization's purchasing server and selects a hyperlink to the selling organization's catalog server.
2. The selling organization's catalog server authenticates the requisitioner based on a digital certificate and then allows the requisitioner to browse, select items, and check out.
3. The content of the order is transferred from the catalog server to the selling organization's OBI server.
4. The sell-side OBI server maps the order into an OBI order request, encapsulated in an OBI object (with optional digital signature), and transmits the order request to the buying organization's OBI server over the Internet.
5. The requisitioner specifies any necessary annotations to the order, and internal approval processes take place.
6. The completed and approved order is mapped to an OBI order format, encapsulated as an OBI object, and transmitted back to the selling organization via the Internet.
7. The selling organization obtains payment authorization, if necessary, and begins order fulfillment.

For additional information on OBI, see www.openbuy.org.

Summary

Ultimately, the architecture of an Internet commerce system (like the architecture of any complex computer system) has a tremendous effect on the long-term success of a project. It is almost always easier to slap something together quickly to solve a particular problem, but the resulting system won't be able to handle the challenges of tomorrow and will quickly become obsolete, even for its original purpose. By carefully creating an architecture, taking into account the business challenges to be addressed and possibilities for change over time, the system can evolve and adapt to growth, new challenges, and technology changes. Over the long term, the up-front investment can have enormous return. Trading those advantages against "let's get it running now" is an important decision that should be made very carefully. Indeed, one lesson from the dot-com craze of the late 1990s is that moving too quickly is not always an advantage, particularly for established businesses with loyal customers.

Assuming now that there is an architecture—at least a simple one—in place, we can consider some strategies for implementing the commerce system, the subject of the next chapter.