Typical Web Services Designs

When a nonprogrammer asks for a quick description of how Web services are different from other distributed technologies, he normally hears a moment of silence while the programmer tries to figure out how to describe it in nontechnical terms.

In truth, many of the words that we use to describe Web services also describe other distributed computing technologies. Throughout this book, we have attempted to describe Web services directly, and through analogies. We realize, however, that sometimes a few examples are worth more than a ream of written description. This is why we have chosen to spend this hour providing examples of real-world systems. Because Web services implementations are still not yet common at the time of this writing, we will describe how you would design three different systems to use Web services.

These descriptions will be presented abstractly without referring to any one tool, language, or development environment.

In this hour, you will learn

- What types of Web service applications are possible
- The rules of thumb for determining if a system is a good candidate to be a Web service
- A basic functional design methodology

You also will examine the following systems:

- Conglomerate reporting system
- Shop floor system
- CheapestCameras.com Web site

Designing the Conglomerate Reporting System

XYZ Corporation is a traditional conglomerate. It buys companies, and it sells companies. Because of this dynamic environment, corporate management does not embark on massive Information Technology projects to integrate the systems of each company into one big system. Each company tends to keep the systems that it has traditionally used, and it makes system improvements based on the business needs of that individual company.

This "islands of technology" approach is much less expensive in the short run than massive system conversions would be. (If the companies continue to grow and if they are not going to be sold, a full conversion might be less expensive in the long term.) The basic weakness of the islands of technology approach is corporate reporting and accounting. Each of the 14 subsidiaries has an accounting team who is able to determine how that company is performing. They print reports showing this performance and mail them to corporate headquarters each quarter. These reports are formatted differently by each subsidiary's accounting department. This arrangement is shown in Figure 5.1.

Corporate accounting receives these different reports, extracts the information from them, and types it into a corporate accounting system. This system combines all the subsidiary's information and produces a corporate-level report that shows the corporation's performance for the quarter.

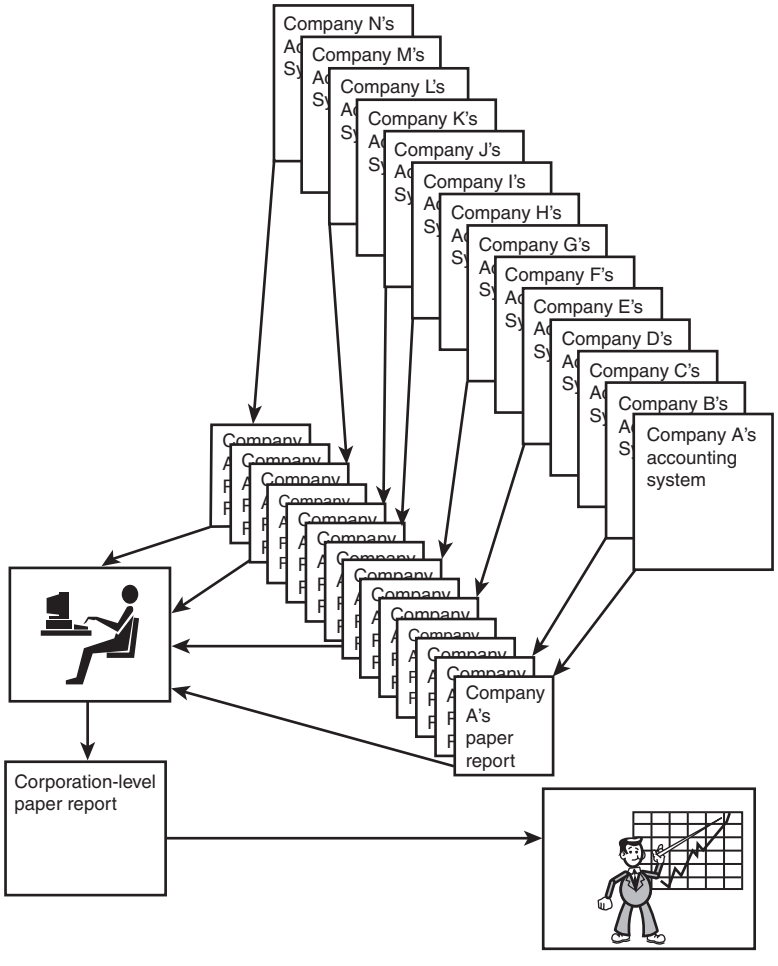

FIGURE 5.1
The original XYZ Corporation's distributed accounting system.

Reasons for Dissatisfaction

The president of the corporation is fundamentally dissatisfied with this procedure for determining profitability. His primary concern is that he only gets the information that he needs once per quarter. The board of directors asks for updates from him more often than that, however. To report to the board, the corporate president has to rely on estimates produced by the presidents of each company. These estimates are sometimes not very accurate. In addition, the corporate president has to weigh the results to reflect the size differences between companies. The end result is that his estimates are not very good. His stated goal is to receive accurate information on a monthly basis.

The corporate accounting department is not happy with the current state of affairs either. The reports are all on paper, which requires that they be entered by hand. In addition, the corporate report must be produced within the three days following the receipt of the individual company reports. This causes the accounting department to sometimes work weekends in order to make the deadline.

Past Attempts at a Solution

A proposal was made to standardize all the companies on one platform. The huge expense of performing the conversions, together with a lack of agreement as to what the standard platform should be, killed this suggestion.

CORBA was suggested, but the widely dispersed geographic locations of the different data centers made this hard. The corporate security decided against allowing direct, non-HTTP socket connections from outside the firewall.

Finally, the idea of having every subsidiary create a flat character-based file in a standard format was suggested. These files could be sent to the corporate server where they could be consolidated. Because this is basically the Web services approach, the decision was made to investigate the use of Web services instead of inventing some proprietary approach.

Basic Analysis

Several characteristics of this system make it a good candidate for a Web services–based solution:

- The volume of data to be transferred is relatively small. For this reason, performance is not an issue.
- The geographic locations involved make an Internet-based approach attractive.
- The security concerns associated with allowing HTTP through the firewall are considered manageable if they use a combination of authentication and encryption.
- All the IT departments in the individual companies can use programming languages that they are familiar with to create their servers.
- The potential acquisition of future companies (whose hardware and operating system configurations are unknown) requires that new companies be able to integrate rapidly—with a minimum of side effects to the rest of the corporation.

These considerations suggest that a Web services approach might be appropriate.

Designing the Conglomerate Web Services Solution

Web services is such a new topic that there is not yet enough experience to suggest a formal methodology for designing a solution. The approach used in this hour might serve as a starting point in this discussion, however.

Defining the Server and the Clients

We will start our discussion by considering the simple case in which there is only one client and one server. More complex Web service transactions can involve any number of Web services chained together.

NEW TERM The first step is to decide which computers will serve as the client and the servers. The client is defined as a consumer of services, and the server is defined as the provider. However, two different types of transmissions are commonly used in the type of Web service that we are designing here: *request/response* and *solicit/response*. With the request/response transmission, the client initiates the action by making a request of the server. In the solicit/response transmission, the server makes the first move by soliciting a response from the client.

Logically, we could envision this example either way. We could have 14 clients interact with one server. On the other hand, we could create 14 servers and one client. Either approach can be made to work, so this is completely dependant on what sorts of business needs we have and what problems we are trying to solve.

Normally, more expertise is required to create a Web service than is required to create a client. The Web service programmer must create, by hand or with tools, both the implementation of the service and the Web services Description Language (WSDL) document. The client, on the other hand, can use the WSDL to generate a good-sized portion of the client code. For this reason, we typically want a design in which the server is defined once and the clients are defined once for each subsidiary.

When there is a one-to-many relationship in your design, make the "one" computer the server and the "many" computers the clients.

Deciding on the Transmission Primitives

The second decision that we must make is whether to require the clients to initiate communication with the Web server or to have the server solicit the responses from the client.

There is another logical reason to prefer the solicit/response transmission style. If we were using a request/response style transmission, what would the server do if no communication were received from a certain client by a certain date? It would have to send a message asking the client to provide the missing data.

Because this "pull" message must exist anyway to handle the case in which a client hasn't sent the data, it makes sense to create the system using the solicit/response transmission style.

Always begin the transmission with the party that wants the results.

Designing the Messages

The next logical step in creating a system is the design of the messages that will be exchanged. In our case, we have decided that the corporate computer is going to be our server and that the communications will follow the solicit/response transmission style. This suggests that two messages need to be sent from the server to the clients:

- Is the report ready for uploading?
- Please give me your report.

Likewise, the clients need two different response messages:

- Yes, the report is ready. (Or no, it is not ready.)
- Here is my report.

The usage of the system would be fairly simple. On a certain date each month, the server solicits the report from each client by sending out the "Is the report ready?" message. If it is ready, the client answers "Yes." Upon receiving the "Yes" response, the server sends a "Please give me your report" message. Following that, the client gathers all the data and sends the "Here is my report" message containing the report.

If the client answers "No" because the report is not ready, a time for the next retry is assigned and the server sleeps until it is time to try it again.

Figure 5.2 shows what the redesigned system will look like.

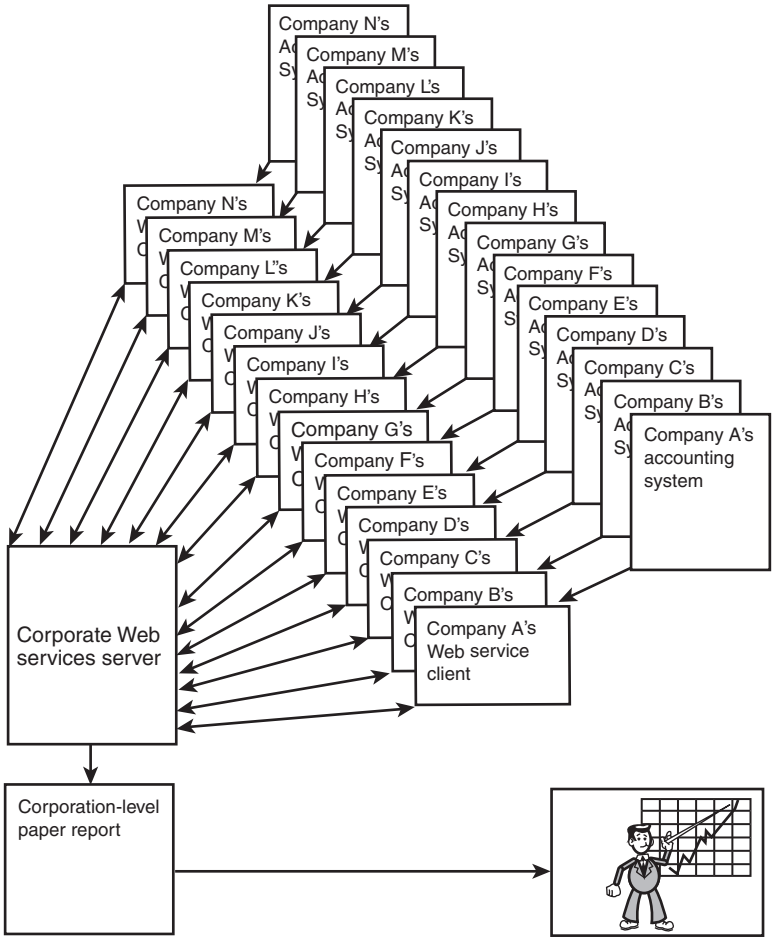

FIGURE 5.2
The Web services version of XYZ Corporation's distributed accounting system.

Now that all the reports are solicited and received in an electronic version that follows a specific format, the amount of work required to produce monthly reports is greatly reduced.

Designing the Project

All that is left is to actually program the parts. The details of the coding process will depend on what tools you choose to implement with. One of the big advantages of Web services is the fact that every one of the 14 different accounting systems can use a different software development tool to create its client code. In addition, the logic in each client will be different because it has to obtain its data from its own accounting system. We will create our example by following these steps:

1. The first step in the design is to write the server software. The WDSL document is then created from the server software—either by hand or automatically.
2. The WSDL is then sent to the programming staff in each subsidiary. Using this document as a guide, they create the client program either by hand or using the development tool that fits in best with their programming language background and interfaces best with their legacy system.
3. As each of the individual subsidiaries completes its work, it is run against a special test version of the Web service.
4. When all the subsidiaries complete their work, the whole system is tested and the results are compared to data that is known to be correct.

At the end of this process, the system is ready to go into production.

Redesigning the Shop Floor System

A fictitious company, MNO Corporation, is an aircraft manufacturing company. It designs and builds airframes, which are airplanes without engines. It purchases the engines from other companies and installs them on the airframes to produce a complete airplane.

The assembly of the aircraft is very complex because there are many different versions of each airplane. Each version is a set of options that the customer can specify. As a result of this, instructions for how to assemble each part of the airplane are created for each version. These documents are called the “work instructions.”

The work instructions only list procedures that can vary from version to version. Procedures—such as drilling, cleaning, sealing, and so on, which are always the same—are described in other documents called “standard procedures.”

Some of the work instructions and standard procedures contain steps that require the use of potentially hazardous solvents and chemicals. Federal law requires that companies make their employees aware of the safety risks associated with each substance by providing a “material safety data sheet” for each of these substances.

In addition, the work instructions have a place on them where a factory worker can use a personalized rubber stamp that indicates the completion of a step. This data is used to show progress to management and the customer. Figure 5.3 graphically shows this system.

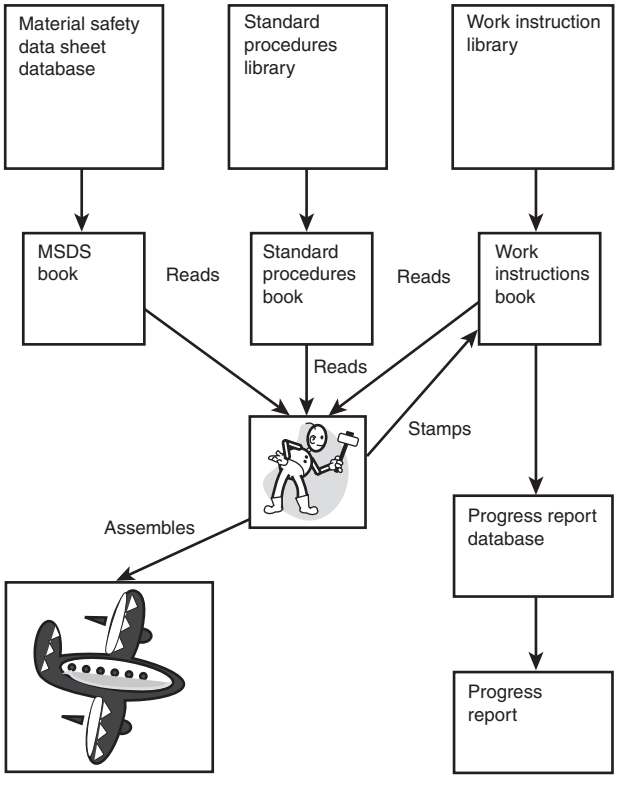

FIGURE 5.3
The original MNO Corporation's manufacturing support system.

The work instructions contain hyperlinks to standard procedures and material safety data sheets. The stamps on the work instructions provide input to the progress report.

Reasons for Dissatisfaction

The printing of the work instruction books for every procedure on every airframe is a very time-consuming, error-prone process. Last minute changes to the instructions cause the books to be broken apart and new work instructions to be inserted. The books are also heavy and cannot be carried from a desk to the work area. This causes the factory workers to under-use the instructions.

If the worker needs a standard procedure or material safety data sheet, he has to go to another book and look it up based on a number in the work instructions.

The stamping of the books is problematic also. Some of the stamps are not legible, which causes the supervisor to have to figure out who did that work and have him stamp it again.

Progress report generation is a manual process. A person goes through by hand and tabulates the percentage of completion. Finally, it takes a warehouse to store all the books that are created.

Management wants a solution that is completely paperless. The workers will view the work instructions on a terminal screen. If they want to look at a standard procedure, it will be a hyperlink that will display right on the screen. If an employee needs to carry a copy back to the work area, a nearby printer can produce that copy on request.

Updates to the instructions will be reflected immediately, so manual updating isn't necessary.

A badge reader will be attached to each terminal. The employees will swipe their employee badges to "stamp" the work as complete. If there is a read failure, they will get a message immediately to swipe the badge again. The progress report can be generated at any point in the process. The archival storage will be online.

Basic Analysis

Several characteristics of the Shop Floor system make it a good candidate for a Web services–based solution:

- The frequency of data to be transferred is relatively low. For this reason, performance is not an issue.
- All the feeder systems—such as the work instructions authoring, the standard procedure authoring, and the material safety data sheet database—are written in different programming languages and are hosted on different platforms.
- All these different systems run on separate networks in the company. The interconnections of these networks are not robust. All the platforms are capable of hosting Web servers, however.
- The Web servers in this company are connected to each other in an intranet. They are not connected to the outside Internet. This reduces security concerns and enables unencrypted data to be transferred from computer to computer.

These considerations suggest that a Web services approach might be appropriate.

Designing the Shop Floor Web Service

We will use the same design process in for the Shop Floor Web service that we used in the Conglomerate Reporting System earlier in this hour.

Defining the Shop Floor Servers and the Clients

This system has many nodes, which are shown here:

- The Work Instructions Authoring System runs on a Mainframe computer.
- The Standard Procedure Authoring System runs on a UNIX server.
- The Material Safety Data Sheet Image Database runs on a Windows NT server.
- The Shop Floor System runs under UNIX.
- The Progress Reporting System runs on a Windows NT server.

Even though there are no true one-to-many relationships between these systems, there is a quasi one-to-many relationship between the Shop Floor System and the systems that feed it: the Work Instructions Authoring System, the Standard Procedure Authoring System, and the Material Safety Data Sheet Image Database. In every one of these cases, the Shop Floor System needs to be capable of requesting a document and receiving a response that contains the requested document. This means that one set of messages could be used for all three of the feeder systems. This suggests that the Shop Floor System should operate as the Web service and the feeder systems as the clients.

The reporting system's relationship to the Shop Floor System is different, however. It requests data from the Shop Floor System, and then formats the response for management reporting. Because this is a true one-to-one relationship between systems, either could be the server. In this case, however, the developers assigned to the Shop Floor System will soon have more experience creating Web servers because of the work they will do in connecting to the feeder systems. For this reason, it makes sense to assign them the more complex role of creating the Web service and allow the reporting system to act as a client. Therefore, two Web services will be located on the server. The work instructions Web service solicits the correct work instructions and the associated documents. The reporting Web service simply provides status information whenever it receives a request from a client.

Because more expertise is required to create a Web service than is needed to create the client, the development team with the most Web services experience should be given the task of creating the service—all other considerations being equal.

Deciding on the Transmission Primitives

The second decision we must make is whether to require the clients to initiate communication with the Web server or to have the servers solicit the responses from the client.

Following the rule that the party who wants the data should initiate contact would suggest that we make the communications between the Shop Floor System and the feeder systems of the solicit/response type. Whenever the Shop Floor System needs a document, it can solicit a response from the client that produces that document.

The reporting system is the one that wants to consume the data provided by the Shop Floor System. This would suggest that a request/response transmission would be the most appropriate. When the reporting system needs the data, it will issue a request to the Shop Floor System's reporting Web service. The service will respond by sending the data needed to the reporting system. The reporting system will then format and display the progress report.

Designing the Shop Floor Messages

The next logical step in creating a system is the creation of the messages that will be exchanged. In our case, we have decided that the Shop Floor System is going to host both of our Web services and that the communications with the feeder systems will follow the solicit/response pattern. The following message will be needed to do this:

- Please provide me with a document with ID=*XXXXX*.

The clients, however, need two different response messages:

- Here is the document that you requested.
- An error message indicating that the document doesn't exist.

The usage of the system would be fairly simple. The Shop Floor System would contain a menu to the different work instructions. The factory worker would choose the instructions for the work that he will do next. At that point, the Shop Floor System Web service would send a message soliciting that work instruction from the Work Instruction Authoring client. That client would return the requested instruction. This instruction could be returned in the form of an XML document, a complex data type, or as an attachment. We will explore the details of these different types of messages in Hour 12, "Sending Attachments with Web Services."

While the factory worker is looking at the instructions, she might also choose to look at a standard procedure. When she clicks the hyperlink to that procedure, the Shop Floor Web service will create a message that solicits a document from the Standard Procedure Authoring client. That client will return this document that the Shop Floor System will display or print.

The Material Safety Data Sheet client will be contacted in the same way. Its response will be different, however, because these documents are scanned in versions of the safety

data provided by each vendor. This type of document will most likely be sent as an attachment.

The factory worker will perform the work, and then use the magnetic strip on his Employee ID to indicate completion of the work.

A second Web service, the Reporting Web service, must be created to respond to requests for status data. Having two Web services associated with one software application is entirely appropriate. We have chosen to create two separate Web services because the two operations are so totally different. Here is the message that the second service exchanges with its client:

- Please provide me with the status of airframe *XXXXX* as of yesterday at noon.

The server, however, needs two different response messages:

- Here is the information that you requested.
- An error message indicating that the request failed for some reason.

The reporting Web service waits for requests to come from the reporting system. When a request arrives, the Web service creates a response by examining its database and extracting the requested data. The response is then sent to the client for processing.

Figure 5.4 shows what the redesigned system will look like.

Now the system will be capable of obtaining work instructions, standard procedures, and material safety data sheets when they are needed. Any last-minute changes to these instructions will be reflected in the documents that arrive. There is no need for a person to update the books by hand because all the books are gone. Status is available at any time via the reporting system's user interface.

Writing the Web Services Code

All that is left is to actually program the parts. Once again, the details of the coding process will depend on what tools you choose to implement with. Regardless of the tools chosen, the following tasks must be performed:

1. The first step in the design is to write the software for the two Shop Floor Web services. The two WDSL documents are then created from the Web services software—either by hand or automatically.
2. The first WSDL is then sent to the programming staff for each feeder system. Using this document as a guide, they create the client programs either by hand or using the development tool that fits in best with their programming language background and interfaces best with their legacy system.

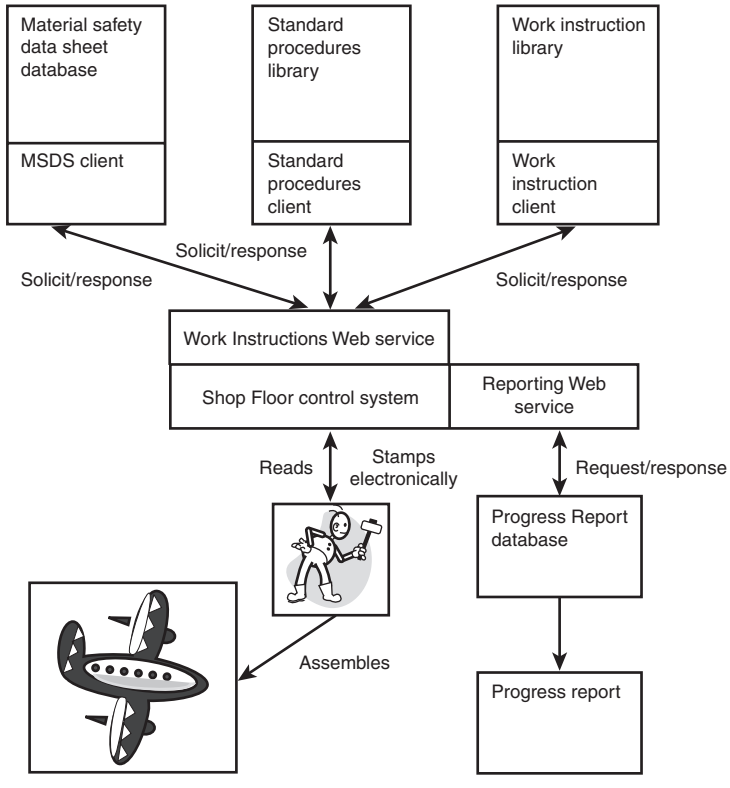

FIGURE 5.4
The Web services version of the MNO Corporation's manufacturing support system.

3. The second WSDL is sent to the reporting system where it is used to generate the Web services client that obtains messages from the shop floor.
4. As each of the individual feeder system programming teams completes its work, it is tested against a special version of the Shop Floor Web service.
5. Test data can be created to represent the progress on an airframe. The Reporting client can then make requests to the Shop Floor Progress Web service to test its functionality.
6. When all the project teams complete their work, the whole system is tested and the results are compared to data that is known to be correct.

This system demonstrates how Web services can be used to perform a variety of functions in an intranet-based system. You can mix and match the transmission types to accomplish a variety of push and pull transactions in your solution.

Designing an E-Commerce Site

The Internet has become a popular place to shop for better prices. This is especially true for items such as expensive cameras. It is theoretically possible for a single Web site to search the best camera prices available on the Internet and display them as if they were in your own warehouse. This would allow you to mark up the products a little and sell them for less than any other site. As new lower-priced wholesalers expose their price lists via Web services, our system would discover them and add their offerings to our Web site. We will call this Web site "CheapestCameras.com."

In the preceding sections, we described two high-level designs that integrated existing systems within one organization. We applied the relatively new approach of creating Web services to integrate the types of systems that have been integrated with RMI, CORBA, or DCOM in the past.

In this section, we will describe a new scenario that is very exciting, but less certain to be implemented on a large scale. This new scenario involves the discovery and interconnection of clients to Web services without human intervention. This notion is a little scary because it establishes a business relationship between parties that don't know each other.

We will use the same process in designing this system that we used in the Conglomerate Reporting System earlier in this hour.

Defining the CheapestCamera Servers and the Clients

This system has many nodes, which are shown here:

- The CheapestCameras.com Web site that is designed to allow a consumer to purchase a camera using a credit card
- A credit card server, which will be used to validate the customer's credit card
- Potentially hundreds of wholesalers' sites that have exposed their inventories as Web services

In this case, it is easy to figure out who is the client and who are the servers. All the camera wholesalers, as well as the credit card service, will be Web services. The only client will be our humble Web site. Customers will access the Web site via a browser.

In addition, our site will use the services of a commercial Web service registry to locate new vendors when they become available. The details of how these registries work can be found in Hour 11, "Advertising a Web Service."

Deciding on the CheapestCamera Transmission Primitives

The second decision that we must make is whether to require the potential customers to initiate communication with the Web service, or to have the service solicit the responses from the client.

A rule of thumb states that the party that wants the data should initiate contact. This would suggest that we make the communications between the client and all Web services conform to the Request/Response style. This means that the client will initiate all requests.

Designing the CheapestCamera Messages

The next logical step in creating a system is the creation of the messages that will be exchanged. In our case, we have decided that the Web site is going to be our client and that all Web services will respond to its requests. The following messages will be sent to each of the camera wholesaler Web services:

- How many cameras of model number *XXXXX* do you have for sale?
- What is the price of camera model number *XXXXX*?
- Please send one camera, model number *XXXXX* to address *YYYYY*.
- Please send the current price list.

The camera wholesaler Web services respond with these messages:

- We have *n* cameras with model number *XXXXX*.
- Camera model number *XXXXX* is $399.
- This is a confirmation that *n* cameras with model number *XXXXX* were sent to address *YYYYY*.
- Here is the current price list.
- An error message indicating that the camera is out of stock or that it doesn't exist.

The CheapestCameras.com Web site will need to send this message to the credit card validation Web site:

- Will you approve a purchase of $399 on credit card number 1111-2222-3333-4444?

In addition, the credit card validation Web service needs to be able to send these messages:

- A purchase of $399 on credit card number 1111-2222-3333-4444 is approved/not approved.
- An error message stating that the credit card number is unknown.

The operation of the system would be fairly simple. At a certain time every night, the Web site would search the Web service repository for new wholesalers to buy from. When the repository finds a new one, it adds the wholesaler to the list of current vendors.

The Web site will then send messages to each wholesaler's Web service asking for the current price list. Each wholesaler's Web service will return the latest price list. The Web site will recalculate its prices based on the information received in these price lists. The Web pages will now reflect these changes.

When a customer goes to the site, she chooses a camera to purchase. During checkout, the Web site sends a message to the credit card validation Web service. If the credit card number is approved, a message is sent to the wholesaler for that model asking if that item is still available. If it is, a message ordering that camera is sent. The wholesaler's Web service will confirm the order and ship the product directly to the customer.

If the camera is not available from this wholesaler, the next cheapest vendor's Web service is sent the same message. This continues until the requested item is located. Figure 5.5 shows what the system will look like.

Now the Web site will be able to function as an online store without carrying any inventory.

Programming the CheapestCamera Web Service

All that is left is to actually program the parts. Once again, the details of the coding process will depend on what tools you choose to implement with. Regardless of the tools chosen, the following tasks must be performed:

1. The first step in this project is to perform a search and find camera wholesalers who are Web services enabled. At the time of this writing, you would not likely find very many. The hope of Web services proponents is that these vendors will appear over time.
2. Each wholesaler Web service will provide a WSDL document. These documents need to be downloaded.
3. The Web site software will be written to act as a client and communicate with each wholesaler Web service to obtain prices and to place orders.

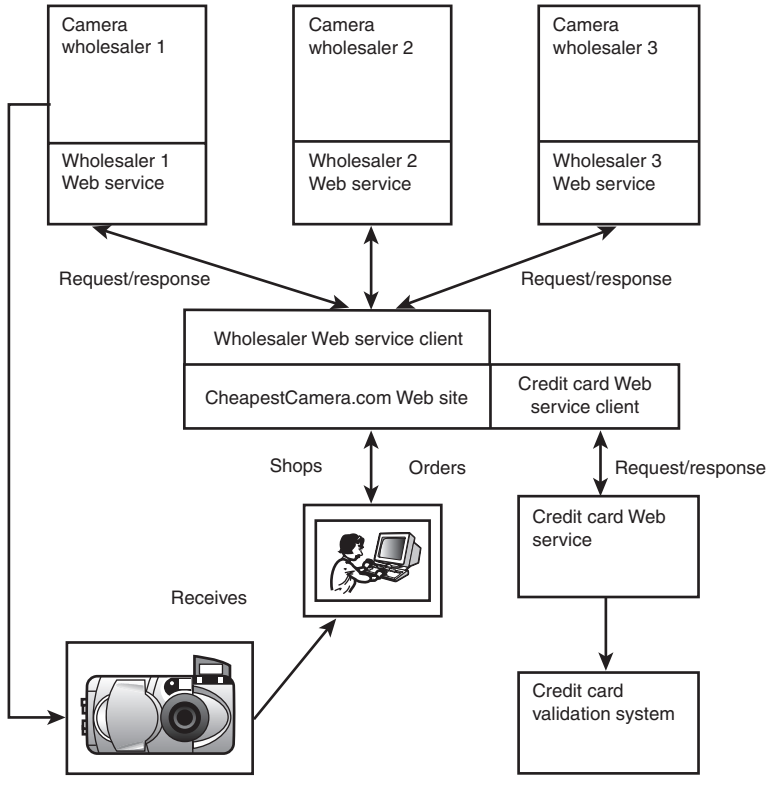

FIGURE 5.5
The high-level design of the CheapestCameras.com Web site and the Web services that it uses.

4. Part of the Web site software must also be written to process the dynamic discovery and setup of new wholesalers that it finds in the Web service registry. This will be the most difficult part to write if it is going to operate without human intervention.
5. The software that is capable of acting as a client to the credit card Web service must be written.
6. The Web site's screens (JSP, ASP, HTML, and so on) must be written.
7. When the project team completes its work, the whole system is tested.

This system demonstrates how Web services can be used to create new types of Web sites that are more dynamic than those presently available. Some hurdles must be overcome to make this practical, though. The first entrepreneur to attempt to create this type of site will have to contact the wholesalers and convince them that they want to invest the money to develop a Web service wrapper around their current systems. Another hurdle will be found in writing the dynamic discovery software. Obviously, the more similar

the messages that each vendor is expecting to receive, the easier this will be to write. If a future standard emerges defining the messages that a vendor must use, the task will be easier still.

The dynamic part of the system is nice, but not strictly required to get the Web site going. If the other parts of the system can be made to work, and if the site is indeed capable of offering better prices than the competition, the business could succeed without dynamic discovery in the short term.

Uniqueness of Web Services Designs

If you have a great deal of experience in using other distributed technologies, you might notice that certain similarities exist between designing for those technologies and designing for Web services. This is certainly true. There are, however, a number of unique characteristics that you must consider when designing a Web services solution:

- **Performance**—Web services ordinarily run over the public Internet. Because the performance of the Internet varies, the response time that you can expect also varies. Systems in which timing is critical are not good candidates.
- **Rollback**—At present, there is no standard approach to rolling back part of a Web service transaction if a later step fails. If this is required, a Web service solution might not satisfy the requirements.
- **Business process execution**—There is currently no standard way to chain together set of Web services into a single transaction using logic. The only way to design multiple server systems is to send them in a single unalterable series. If...then...else logic simply isn't possible with current standards.
- **Sophisticated security**—At present, only Secure Socket Layer (SSL) can be used in Web services. If you need sophisticated encryption and decryption or digital signatures, you will have to wait for standards to emerge or use a different technology.

On the other hand, Web services frees you from many restrictions that exist for other technologies:

- The geographic location of each computer participating in the transaction disappears with an Internet-based approach.
- The security concerns associated with allowing HTTP through the firewall are normally easier for your Computer Security Department to approve than many of the other types of connections such as direct sockets.
- All the potential client programmers can use programming languages that they are familiar with to create their clients.

- Because of the existence of the WSDL, your clients might be able to generate the code needed to create a connection to your Web service.

If the benefits provided by Web services outweigh the current drawbacks, a Web service design might be appropriate. If not, you might eventually be able to use Web services once some of the drawbacks are lessened or eliminated by the further maturing of the core Web services specifications.

Summary

This hour shows you how Web services can be used to integrate different systems within the same organization. In these solutions, the Web services architecture is used primarily as a means of communication between heterogeneous hardware and software platforms.

You first saw a system design that gathered the same information from all of its subsidiary companies and merged them together to form a corporate financial statement.

Next, you saw a design that featured two very different Web services implemented by the same software system. These two services use two different transmission styles to communicate with their clients.

Finally, you saw a design for a Web site that takes advantage of Web services to provide products to its customers at a lower price.

Q&A

Q **Generally speaking, does creating a Web service require more time and effort than creating a client?**

A Yes. The designer of the service must create the messages and operations that will be described in the WSDL. The client just uses the WSDL.

Q **What are the rules of thumb for deciding whether a certain node will be a Web service or a client?**

A There are two rules. A one-to-many relationship suggests that the one be the server because servers are harder to write. In a one-to-one relationship, you would want to make the node with the most experienced Web service programming staff the server. If both sides are equally experienced, choose the node that makes sense to you.

Q What is the rule of thumb governing which node should initiate the communication?

A Normally, the node that is the consumer of the data should request it. The reason for this is that the consumer must have a way of requesting data anyway if the other node never sends it on its own.

Q Shouldn't the client always request and the server always provide information to it?

A No. Servers can request, or solicit, data if the design calls for it. Of course, clients can request data too.

Workshop

The Workshop is designed to help you review what you've learned, and begin learning how to put your knowledge into practice.

Quiz

1. What language are clients written in?
2. What is the best hardware platform for Web services?
3. What development tools should I purchase (or download) for developing Web services?
4. Do all the development tools on the market provide about the same functionality?

Quiz Answers

1. Clients can be written in any language that can be used to create text documents, which is any language.
2. All platforms are capable of producing Web services because Web services are abstract enough to allow any platform to implement them. The declaration of the best one will remain a subject of debate.
3. Normally, you should purchase a tool that supports your current development environment, which means that tools such as Apache Axis make sense for Java shops. Microsoft .NET makes sense for Windows programmers, and WebLogic Workshop would make sense for WebLogic shops.
4. No. The tools on the market vary greatly in the amount of automation that they provide. Some tools, such as Apache Axis, are appealing to programmers who like to work at the nuts-and-bolts level. At the other end of the spectrum, WebLogic Workshop is highly automated and generates almost all of the boilerplate code necessary for your service to function.

Activities

1. Using the previously outlined process, design a Web service that your organization might want to implement one day.
2. Specify which computers would host the Web services and which would act as clients.
3. If this Web service would replace an existing system or systems, create a detailed statement of reasons for dissatisfaction with the current system.