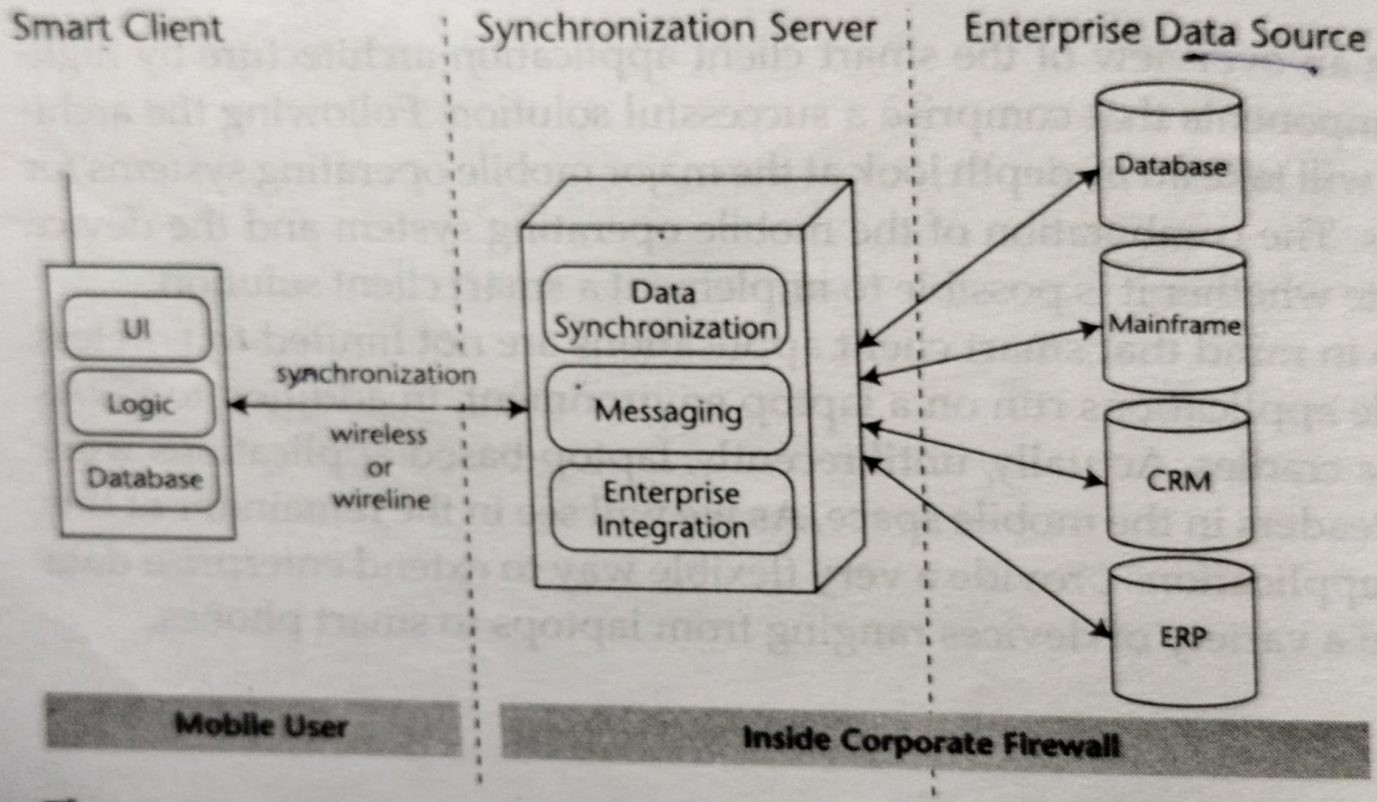


**Figure 7.1** Smart client architecture.

U-4.

I) Smart Client Architecture.

(Diagram)

→ Smart Client applications enable the user to access data when disconnected from the network.

→ On client side we have UI, business logic and persistant data store.

→ This application communicates with backend data source through intermediate Synchronizati Server.

→ The communication stream can run either wirelessly or over a wireline connection. The connection may require an IP based network or an additional communication layer for Synchronization process.

→ The smart logic determines where the application gets its data from, how the data is presented & stored, as well as the set of data that needs to be communicated back to the system via Synchronization process.

## Data Synchronization

When we refer to data synchronization, we are talking about how enterprise data is moved from the back-end enterprise system to the mobile device, and vice versa. This data movement can be accomplished in a number of ways, depending on the solution that is chosen. For a general overview, we will look only at the major components that are required in a synchronization server, as shown in Figure 7.2. These components include the interface to the client application, the synchronization logic, and the integration layer to the back-end data source.

Data can be sent between the client application and the synchronization server in a variety of formats. Transferring data over a wired connection is a fairly straightforward process. Transferring it over a wireless network can be a different story. Depending on the requirements of the synchronization software, some wireless networks may work better than others, and some may not work at all. Discussions about wireless data synchronization are saved for Chapter 10; for the purpose of this overview, we will assume a valid connection exists between the client and the server.

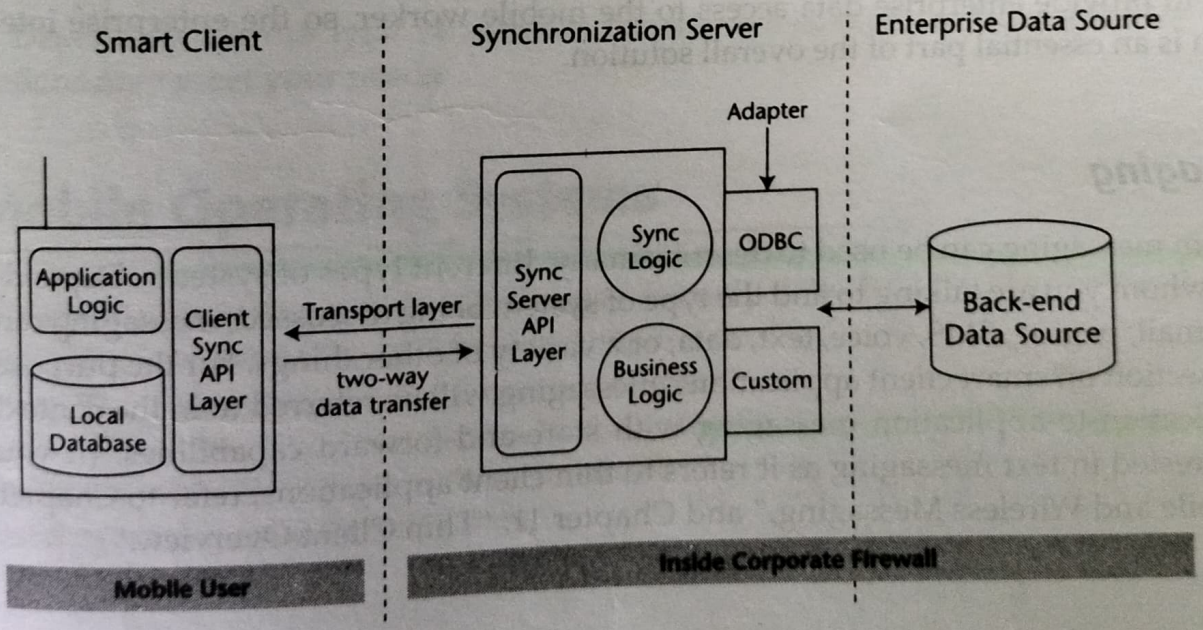


**Figure 7.2** Synchronization architecture.

If you are developing an in-house solution, you may choose to create a proprietary data stream format, possibly basing it on XML, or you may decide to use an industry-specified format such as SyncML. The same options are available in commercial synchronization software. In both cases, a stream of data is being transferred that both the client and the server know how to decipher. When the server receives data, it needs to know how to process it. This capability involves understanding the data layout of the enterprise data source. In the most basic form, the synchronization logic will simply take the updates and apply them to the data store, without considering whether they should be applied or not. In more complex cases, the synchronization logic will perform a variety of duties, including conflict detection and resolution.

The integration with the back-end data source is often a simple process, using Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC) drivers for direct database access. At the same time, integration can be difficult, requiring complex logic to interface with more complicated systems such as Enterprise Resource Planning (ERP) systems. In either case, providing the ability to access enterprise data is critical for an enterprise synchronization solution. Rarely does a company use only one enterprise data storage mechanism, so, ideally, the synchronization server will be flexible enough to integrate with a variety of back-end sources.

## Enterprise Data Source

The final part of the smart client solution is the enterprise data itself. This data will vary in formats ranging from enterprise databases from vendors, such as Oracle, Sybase, Microsoft, or IBM, to flat-file systems and everything in between. For the more common storage systems, you should be able to find a driver or adapter that will provide an integration layer for your synchronization server. If you are using something that is uncommon or something developed in-house, you will most likely have to develop this integration layer yourself.

Keep in mind that one of the most common reasons for implementing a mobile solution is to provide enterprise data access to the mobile worker, so the enterprise integration is an essential part of the overall solution.

## Messaging

The term messaging can be used to describe many different types of systems. Depending on whom you are talking to and the type of system being discussed, messaging can mean email, paging, SMS, voice, text, data, or a variety of other things. For the purpose of this section on smart client applications, messaging will be referred to in the context of application-to-application messaging with store-and-forward capabilities. (If you are interested in text messaging as it refers to thin client applications, refer to Chapter 5, "Mobile and Wireless Messaging," and Chapter 11, "Thin Client Overview.")

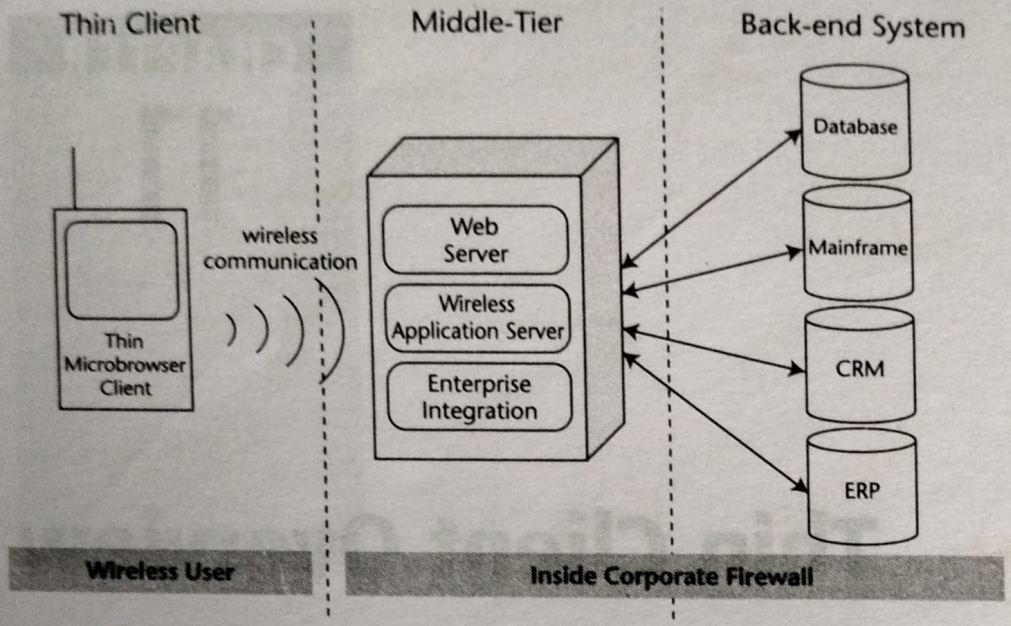


**Figure 11.1** Wireless Internet architecture.

The following are the key components of the thin architecture:

**Thin client.** The client is a microbrowser focused on the presentation of the markup language. It is the part of the application that the end user sees, although it does not execute any of the application logic. The design of the client user interface is crucial to the overall success of the application.

**Middle tier.** The middle tier is where most of the work is performed. A couple of main components run in this tier: the Web server and the wireless application server. The Web server receives the inbound HTTP requests and routes them to the wireless application server. The application server then takes these requests and executes the appropriate logic. This execution may include session management, content management, as well as integration to the back-end system.

**Back-end system.** The back-end system is where the data and related services reside. This system may be a relational database, an email server, business applications, an XML content source, or any number of other enterprise systems. In any case, it is important that this data can be reached, so that it is available to the mobile worker.

## Processing a Wireless Request

With an understanding of the various parts of the wireless Internet application architecture, we can now examine what is involved in each stage of a wireless Internet request. Figure 11.5 shows each step that a request goes through as it is processed.

The following is a detailed explanation of the steps of the process illustrated in Figure 11.5 (the numbers here correspond with the numbers shown in the illustration):

1. *Establish a wireless session.* If the device is not already connected to a wireless network, a connection has to be made. Most packet-data networks such as Mobitex and GPRS allow devices to be always connected, so this step may not be required. On networks that are not packet-based, such as GSM, the user will have to be authenticated to establish a connection.
2. *Submit a request.* The process starts with the user submitting a request through the client browser. The server uses this request object to obtain information about the user. It contains the URL of the page being requested, as well as other information about the device and browser being used. In addition, it specifies whether it is a GET or a POST request. This request is often encoded in a binary format to reduce the data sent over the bandwidth-limited wireless network. The data is transmitted wirelessly to the network tower, or base station, at which time it is transmitted over a wireline connection to the wireless gateway.
3. *Translate a request.* The wireless gateway decodes the request object from the binary format to a text format and forwards it to the enterprise wireless application server. This request is converted from the wireless protocol (for example, WAP) to HTTP and transported over a wireline IP-based network connection. In order to convert from the wireless protocol to HTTP, the data has to be decrypted. After the conversion, it can once again be encrypted, most commonly using Secure Sockets Layer (SSL). This decryption and reencryption process is often called the WAP gap. Because of this moment of weakened security, many vendors allow the enterprise to host the wireless gateway on-premises, so that data conversion can occur within the corporate firewall, adding additional security.

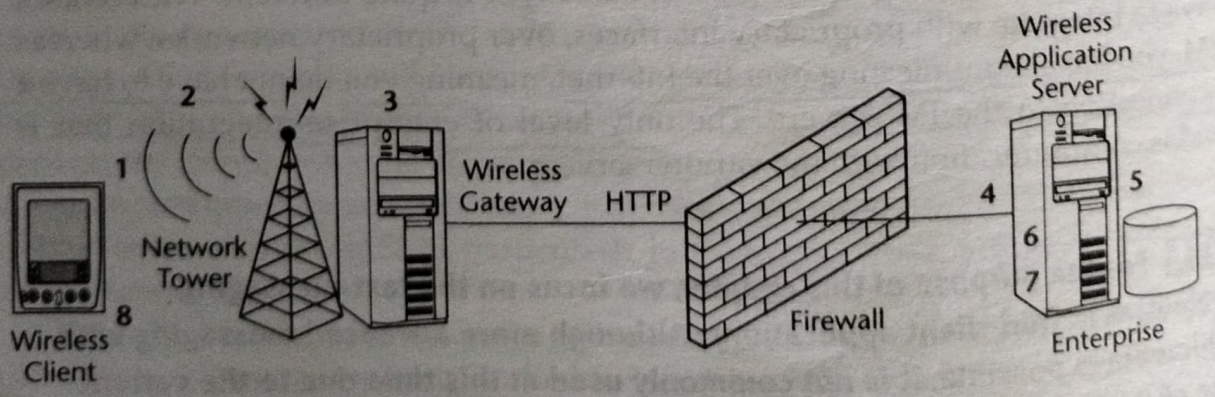


**Figure 11.5** Stages of a wireless Internet request.

4. *Receive a request*. The request is received by the wireless application server. In most cases, the wireless interface to the enterprise application is by HTTP (or HTTPS) using Java servlets/JSPs, ASPs, or other Web interfaces. The wireless application server/Web server can receive the request and perform the appropriate transformation based on the client device/browser combination. The server also performs many other tasks as described earlier in this chapter. If the corporation does not want to maintain this server in-house, many vendors offer hosting services: They host the wireless server and connect to the enterprise only for data access.

5. *Identify the wireless client*. Using the HTTP request object, the wireless server can determine which device and microbrowser is making the request. The request object may also be used to determine other information, such as the image types supported, and preferred language. Using this information, the server can ensure that the appropriate content is sent back to the client. At this point, users may be authenticated to the enterprise system to make sure they have access to the data being requested.

6. *Process the request*. Once the user is authenticated, the server can process the request. The URL will specify the information that is required. If it is a static file, the server will simply return the file to the wireless gateway. If the user requires dynamic content, the data can be personalized based on user information. This will involve accessing one or more enterprise data sources to obtain the dynamic data.

7. *Transform content for device*. At this point, the enterprise information has to be transformed to the appropriate format for the client that made the request. This job may involve changing the markup language, the image types, as well as the richness of the user interface. In most cases this is accomplished on the wireless application server located in the enterprise, although in some situations the wireless gateway performs this translation. Many servers on the market use XML as the base data format and transform it using XSL stylesheets for specific devices and browsers.

8. *Return the content*. Once the content is formatted appropriately, it is sent back to the client. It will once again pass through the wireless gateway where it will be encoded to the format that the browser understands. Any information that the server wants to communicate back to the client browser is contained in the response object.

# Wireless Application Protocol (WAP) Overview

The Wireless Application Protocol (WAP) is a worldwide standard for the delivery and presentation of wireless information to mobile phones and other wireless devices. The idea behind WAP is simple: simplify the delivery of Internet content to wireless devices by delivering a comprehensive, Internet-based, wireless specification. The WAP Forum released the first version of WAP in 1998. Since then, it has been widely adopted by wireless phone manufacturers, wireless carriers, and application developers worldwide. Many industry analysts estimate that 90 percent of mobile phones sold over the next few years will be WAP-enabled.

The driving force behind WAP is the WAP Forum component of the Open Mobile Alliance. The WAP Forum was founded in 1997 by Ericsson, Motorola, Nokia, and Openwave Systems (the latter known as Unwired Planet at the time) with the goal of making wireless Internet applications more mainstream by delivering a development specification and framework to accelerate the delivery of wireless applications. Since then, more than 300 corporations have joined the forum, making WAP the de facto standard for wireless Internet applications. In June 2002, the WAP Forum, the Location Interoperability Forum, SyncML Initiative, MMS Interoperability Group, and Wireless Village consolidated under the name Open Mobile Alliance to create a governing body that will be at the center of all mobile application standardization work.

The WAP architecture is composed of various protocols and an XML-based markup language called the Wireless Markup Language (WML), which is the successor to the Handheld Device Markup Language (HDML) as defined by Openwave Systems. WAP 2.x contains a new version of WML, commonly referred to as WML2; it is based on the eXtensible HyperText Markup Language (XHTML ), signaling part of WAP's move toward using common Internet specifications such as HTTP and TCP/IP.

In the remainder of this section we will take a look at the WAP programming model and the various components that comprise the WAP architecture. Where it is applicable, we will supply information on both the WAP 1.x and 2.x specifications. (More information on the leading markup languages used in wireless Internet applications is provided in Chapter 13.)

## WAP Programming Model

The WAP programming model is very similar to the Internet programming model. It typically uses the *pull* approach for requesting content, meaning the client makes the request for content from the server. However, WAP also supports the ability to *push* content from the server to the client using the Wireless Telephony Application Specification (WTA), which provides the ability to access telephony functions on the client device.

Content can be delivered to a wireless device using WAP in two ways: with or without a WAP gateway. Whether a gateway is used depends on the features required and the version of WAP being implemented. WAP 1.x requires the use of a WAP gateway as an intermediary between the client and the wireless application server, as depicted in Figure 11.6. This gateway is responsible for the following:

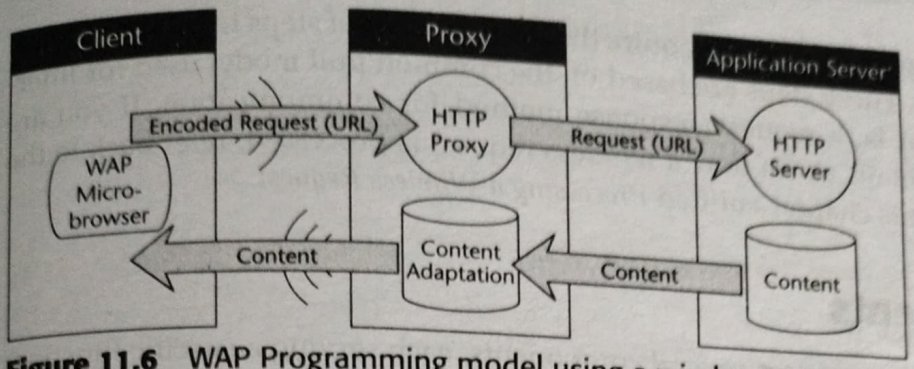


**Figure 11.6** WAP Programming model using a wireless gateway (or proxy).

- Translating requests from the WAP protocol to the protocols used over the World Wide Web, such as HTTP and TCP/IP.
- Encoding and decoding regular Web content into compact formats that are more appropriate for wireless communication.
- Allowing use of standard HTTP-based Web servers for the generation and delivery of wireless content. This may involve transforming the content to make it appropriate for wireless consumption.
- Implementing push functionality using WTA.

**NOTE** **The WAP gateway is often called the *WAP proxy* in the WAP 2.x documents available from the OMA. In this chapter we continute to refer to it as the WAP gateway; just be aware that both terms are used to refer to the same technology.**

When developing WAP 2.x applications, you no longer are required to use a WAP gateway. WAP 2.x allows HTTP communication between the client and the origin server, so there is no need for conversion. This is not to say, however, that a WAP gateway is not beneficial. Using a WAP gateway will allow you to optimize the communication process and facilitate other wireless service features such as location, privacy, and WAP Push. Figure 11.7 shows the WAP programming model without a WAP gateway: Note that removing it makes the wireless Internet application architecture nearly identical to that used for standard Web applications.

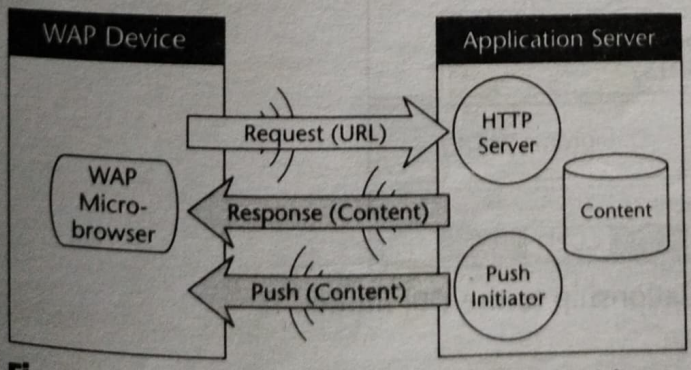


**Figure 11.7** WAP programming model without gateway.

## Needs Analysis Phase

A wide variety of mobile solutions are on the market, including voice-based solutions, wireless Internet applications, smart client applications, and messaging-based applications. Which type is best suited for your current project? The answer to that question must be determined during the needs analysis phase of your mobile application development process.

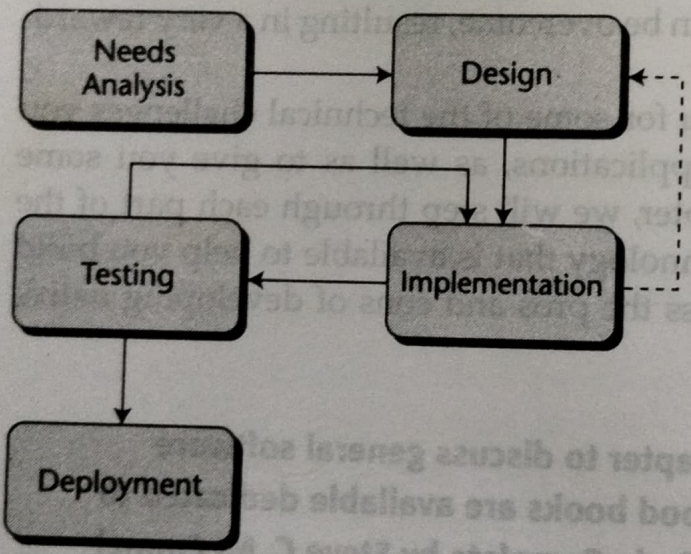


**Figure 8.1** Smart client development cycle.

## Questions to Ask When Researching User Requirements

While you are researching the user requirements, the following are some questions you may want to ask yourself:

- Who are the end users of this application?
- What is their technical skill level?
- What is the overall goal of this application?
- What data integration is required? Does the user require data access at all times?
- Does this application require wireless connectivity? If so, what type of wireless access does it require, and in which geographies?
- What are the primary usage scenarios for this application?

Try to answer most, if not all, of these questions before beginning to research the device, operating system, and development platform you are going to use. It is very easy to focus on the device and related operating system before determining the actual needs of the end user. If you fall into this trap, be careful not to let the device determine the features of the application. A complete application analysis will lead you to a solution that will be well received by the end user.

## Things to Consider

As we consider the issues inherent to gathering the application requirements, we are going to focus on those related to smart client applications. (In Chapter 12, "Thin Client Development," we will look explicitly at the requirements relevant to wireless Internet applications. In Chapter 4, "Mobile Application Architectures," we gave an overview of smart client and thin client applications and the benefits—and drawbacks—related to each architecture type.) Once we gather information about the following items, we can make an educated decision on the solution that is most appropriate, including the operating system, device, wireless network provider, and development platform.

### Application Goals

Across the globe, companies are implementing mobile and wireless applications for a variety of reasons. In some cases, the goal is to increase productivity; in others, it is to reduce costs; while in still others, the reason may be simply that it seemed like the next logical thing to do, regardless of the return on the investment. Whatever the reason, it is important that the final solution satisfy the original purpose of development.

During the needs analysis stage, it is important to determine why the project is being undertaken and to produce a result that is in line with expectations. If, say, the project is being implemented to automate order entry, thereby reducing paperwork, the final application should be well suited for rapid data entry, with integration back into the enterprise system. The technical features being implemented should correspond to a business problem that it will solve. By creating a user interface that allows for rapid data entry, we are solving the order entry problem. By having enterprise integration, we are ensuring that the data entered on the mobile device will flow into the back-end system without requiring additional paperwork. These are just a few examples of how technology can help a corporation meet its business goals as it implements a mobile solution.

### End User

The end users of the application play a key role in the application design and rollout. Their level of technical ability determines some of the intricacies that may be required. If the user is not comfortable setting up wireless network connection parameters, such details will need to be automated within the application. If the user does not have experience with pen-based input devices, using a device with an alphanumeric keyboard may be more appropriate. Many mobile applications fail to reach their potential because they were not developed with the end user in mind.

### Data Access

Extending corporate data to mobile users is essential to many mobile solutions. If this is the goal of the application being developed, then data integration and client storage options will have to be considered. Other factors to consider are how fresh the data must be on the client, how much data should reside on the client, and how often inputted data should be sent back to the server. All of these items will factor into the decision on the data storage and synchronization systems being used.

### Wireless Access

Not all mobile applications require wireless access. In fact, many applications are better off without it since real time data access is rarely a requirement. Updates can be made over wireline connections, forgoing the need for wireless access. Other reasons for avoiding wireless access include: complexity of implementation, unreliability of the networks, cost of deployment, or lack of performance.

That said, many compelling reasons do exist for incorporating wireless access into a mobile solution. For field services applications, access to updated work orders, inventory analysis, or routing information can be critical to the success of the solution. Determining the level of wireless connectivity will affect both the implementation criteria and the cost of any given solution.

### Usage Scenarios

Predicting how an application will be used is not easy. Nevertheless, it should be attempted, because having some idea of possible use can prove helpful for the application design phase. If, for example, an application is going to be used constantly and

## Design Phase

After performing the needs analysis, you will have a good idea of the characteristics the solution requires. You will have limited the number of operating systems, devices, and networks to just a handful that will meet your requirements. From these you will need to make a decision as to which solution best meets your needs and addresses your business constraints. When doing this, review your options with an open mind. Try not to gravitate toward a particular solution immediately, as it is difficult to recognize at this point what will be a successful implementation unless you have an unsuccessful solution for comparison.

One way to accomplish this is to review all the solutions that could possibly meet your requirements, without giving any weight to an individual category. So, for example, if an operating system will meet the requirements, but not perhaps as well as another OS, keep it on the list as a potential candidate. The same goes for devices, wireless networks, and development platforms. After you have gone through this process, ideally you will have two or three possible solutions on which you can concentrate.

Now you can start adding weight to the individual categories to help you choose the option that best meets your criteria. Remember to factor in some "soft" factors as well, such as costs, ease of use, and aesthetics, in addition to the core technical requirements. Once you have chosen what appears to be the ideal solution, keep the other options on file, as you may need to come back to them in the future. (Refer back to Chapter 2, "Mobile Devices," if you require a summary of the device market.)

At this point, you can start to focus on the application design, looking at such issues as what data is required on the device, how you are going to integrate to the enterprise systems, and how the user interface should be designed for maximum productivity.

### Client Data Access

Enterprise data sources will contain much more data than you can hope to store on the mobile device. This is true for enterprise databases such as Oracle, Sybase, or IBM; for ERP systems such as SAP or PeopleSoft; and for CRM systems such as Siebel. The amount of data stored on the device will depend on the mobile data store solution being used, device performance characteristics, and physical limitations of the device. You may only be able to store between a few hundred kilobytes to a couple megabytes of data on the client.

When designing a smart client application, start by examining the subset of data that is required for the mobile user. This subset can be determined by looking at many factors, often depending on the type of application being developed. The subset of data that is required can have various levels of granularity. You can partition the data by its structure in the enterprise source—taking only specific tables from an enterprise database. Another approach is to base the partition on specific data, such as a userid, geography, or price range. The more layers of partitioning you add, the more specific the data on the device becomes.

Let's look at an example sales force automation (SFA) application. For such a system, the enterprise database may contain customer contact information, order history, product information, inventory levels, and supplier information. We do not need all of this data on the mobile device, so we can take a subset of it. In this case, only the contact information, order history, and inventory level tables provide a benefit on the device, so we can limit the data to those tables. To go even further, we can say that we only want data for a particular sales representative, perhaps based on a userid, to be sent down to the device. The result is that we will have a subset of the enterprise data that contains everything relevant to the mobile user, and nothing that is not.

The data access design goes beyond just evaluating the subset of data that is required. Before you can go much further, you will need to establish how you want to implement the data store on the client. If you are implementing your own solution, you will want to start laying out your data storage and management logic. If you are going to use a commercial solution, you will want to start examining how storage is implemented in it, and work on the related design issues, such as the database schema. (For more information on persistent data storage on the client device, see Chapter 9, "Persistent Data on the Client.")

## Enterprise Integration

Enterprise integration is a term used to describe any communication to systems not on the device. It encapsulates integration with enterprise databases, business applications, XML data, Web content, and legacy data, among other things. For the purpose of designing your mobile solution, you will need to determine to which enterprise systems you require access. This should be based upon the data access requirements that you have already set forth in the client data access stage of the design process. There are two levels of integration that you may require: basic integration and complex integration.

Basic levels of enterprise integration include the ability to access enterprise databases using defined communication protocols. These capabilities may include the following:

- Device communication using the standard synchronization software such as HotSync for Palm devices or ActiveSync for Pocket PC devices
- Communication over IP-based networks
- Direct integration with a relational database or flat-file system
- Limited support for transactions

Usually, you can follow a well-documented API for these types of applications. This integration layer either can be developed in-house, or you can use a commercial synchronization solution. In either case, the enterprise integration is much more straightforward than it is for larger and more complex systems.

When you start incorporating other forms of enterprise data and other communication protocols to the solution, a more sophisticated enterprise integration layer is required. The capabilities for this type of system may include the following:

- Support for a variety of mobile clients including laptops, handheld PCs, and PDAs
- Communication over networks that may not be IP-based
- Support for synchronizing multiple users simultaneously to a central back-end data store
- Communicating with systems that do not have well-defined interfaces, often requiring custom adapters
- Synchronization of complex data models
- Support for very large amounts of data with many transactions
- Conflict detection and resolution
- Administration tools to manage the entire process

If your system requires these advanced features, you should look to a commercial solution so you do not consume too many internal resources on one aspect of the overall solution. Purchasing a solution outright will result in significant cost savings, compared to the costs involved with writing, testing, debugging, and maintaining the conduit that is required to support the features listed. Even with the use of commercial software, the design requirements are much more complex than for a basic system. You will still need to worry about security, data partitioning, enterprise system access, and scalability, in addition to the usual network coverage and bandwidth issues. (Data synchronization is covered in depth in Chapter 10, "Enterprise Integration through Synchronization.")

## User Interface

The user interface can account for as much as 80 percent of the total code in a mobile solution. When you have one part of the application accounting for such a large portion of the development effort, it has to be designed correctly to avoid costly changes later in the development cycle.

Be consistent with the user interface (UI). Get a feel for how standard applications that come installed on the device work and stick very closely with their UI styles. Most of these applications provide a very basic entry screen that meets the needs of most users. Typically, these UIs make more advanced functionality accessible via user selection of a button or menu item.

A couple of key items to keep in mind when designing on the user interface include screen size and human interaction. Paying attention to these items will make the application effective for the application users and help you avoid changes later in the development cycle. They are explained in the following paragraphs.

### Screen Size

One of the most dramatic differences between desktop applications and those developed for mobile devices is the screen "real estate." When targeting mobile applications, you will have a one-half VGA, one-quarter VGA, or even smaller screen to work with. Using the screen to its maximum benefit is crucial for successful applications. There are many different ways to accomplish this goal, depending on the operating system you are using. Windows CE, for example, supports a tab-based interface, allowing easy navigation to multiple forms with the click of a button. Taking advantage of the menu capabilities in Palm OS applications allows more information to be displayed on a small screen.

### Human Interaction

Studying human interaction with your application will prove to be invaluable in determining its overall usability. History has shown that people always interact with the system in unexpected ways. There is no way that these ways can be predicted, so testing is critical.

Before you get to the stage where you can study human interaction, there are application-specific requirements that can lead the direction of the user interface. If the application focuses on data input, then the main input screens should be easy to navigate using the input properties of the device. For example, if the device offers keyboard support, make it possible for the user to quickly tab between entry fields, rather than having to use a scroll-wheel to get there. If the only means of input is a stylus with character recognition, then including radio buttons and drop-down lists are effective ways to improve the efficiency of data input.

By focusing on the screen design and the navigation practices of the general user, you will be able to design effective user interfaces on devices of all sizes. Do not forget that spending some extra time testing the application early in the development cycle can save you much more time later on.

- *Wireless networks operate over a variety of protocols*. Some of these are not IP-based, so if your application requires IP for data communication, you may have to add an additional IP layer for connectivity. Many software vendors can provide this layer for you, if required.
- *Limit the frequency of wireless data transfer*. Because there are potential problems with the reliability of wireless networks, this consideration will make the application more effective. Having suitable persistent data storage within the application will make it possible to limit the frequency of network connectivity within the application. Infrequent wireless data transfer also has a positive effect on battery life, as wireless communications require more battery power than local access.
- *Limit the amount of data transferred*. This limitation is urged, once again because of the nature of the wireless networks.

## WAP Protocol Stack

The WAP protocol stack has undergone significant change from WAP 1.x to WAP 2.x. The basis for the change is the support for Internet Protocols (IPs) when IP connectivity is supported by the mobile device and network. As with other parts of WAP, the WAP 2.x protocol stack is backward-compatible. Support for the legacy WAP 1.x stack has been maintained for non-IP and low-bandwidth IP networks that can benefit from the optimizations in the WAP 1.x protocol stack.

We will take a look at both WAP 1.x and WAP 2.x, with a focus on the technologies used in each version of the specification.

### WAP 1.x

The protocols in the WAP 1.x protocol stack have been optimized for low-bandwidth, high-latency networks, which are prevalent in pre-3G wireless networks. The protocols are as follows:

**Wireless Session Protocol (WSP).** WSP provides capabilities similar to HTTP/1.1 while incorporating features designed for low-bandwidth, high-latency wireless networks such as long-lived sessions and session suspend/resume. This is particularly important, as it makes it possible to suspend a session while not in use, to free up network resources or preserve battery power. The communication from a WAP gateway to the microbrowser client is over WSP.

**Wireless Transaction Protocol (WTP).** WTP provides a reliable transport mechanism for the WAP datagram service. It offers similar reliability as Transmission Control Protocol/Internet Protocol (TCP/IP), but it removes characteristics that make TCP/IP unsuitable for wireless communication, such as the extra handshakes and additional information for handling out-of-order packets. Since the communication is directly from a handset to a server, this information is not required. The result is that WTP requires less than half of the number of packets of a standard HTTP-TCP/IP request. In addition, using WTP means that a TCP stack is not required on the wireless device, reducing the processing power and memory required.

**Wireless Transport Layer Security (WTLS).** WTLS is the wireless version of the Transport Security Layer (TLS), which was formerly known as Secure Sockets Layer (SSL). It provides privacy, data integrity, and authentication between the client and the wireless server. Using WTLS, WAP gateways can automatically provide wireless security for Web applications that use TLS. In addition, like the other wireless protocols, WTLS incorporates features designed for wireless networks, such as datagram support, optimized handshakes, and dynamic key refreshing.

**Wireless Datagram Protocol (WDP).** WDP is a datagram service that brings a common interface to wireless transportation bearers. It can provide this consistent layer by using a set of adapters designed for specific features of these bearers. It supports CDPD, GSM, CDMA, TDMA, SMS, FLEX (a wireless technology developed by Motorola), and Integrated Digital Enhanced Network (iDEN) protocols.

### WAP 2.x

One of the main new features in WAP 2.x is the use of Internet protocols in the WAP protocol stack. This change was precipitated by the rollout of 2.5G and 3G networks that provide IP support directly to wireless devices. To accommodate this change, WAP 2.x has the following new protocol layers:

**Wireless Profiled HTTP (WP-HTTP).** WP-HTTP is a profile of HTTP designed for the wireless environment. It is fully interoperable with HTTP/1.1 and allows the usage of the HTTP request/response model for interaction between the wireless device and the wireless server.

**Transport Layer Security (TLS).** WAP 2.0 includes a wireless profile of TLS, which allows secure transactions. The TLS profile includes cipher suites, certificate formats, signing algorithms, and the use of session resume, providing robust wireless security. There is also support for TLS tunneling, providing end-to-end security at the transport level. The support for TLS removes the WAP security gap that was present in WAP 1.x.

**Wireless Profiled TCP (WP-TCP).** WP-TCP is fully interoperable with standard Internet-based TCP implementations, while being optimized for wireless environments. These optimizations result in lower overhead for the communication stream.