بسم الله الرحمن الرحيم

University of Diyala
Collage of Science
Computer Science Department

02/05/2018
Fourth Year
Mobile Computing, 5th & 6th lecture

Mobile Application Architectures 2 and OS

Introduced by
Assist. Lect. Musaab Mohammed Jasim
Email mus.albiaty85@gmail.com

Topics:

i. Smart Client.
ii. Messaging.
iii. Mobile Operating Systems.

i. Smart Client.

Smart client applications are a powerful alternative to wireless Internet applications. Instead of using a microbrowser on the client, custom software is developed. This software typically contains a persistent data storage mechanism as well as business logic. This means that smart client applications can be executed at any time, even when a wireless connection is unavailable. Actually, smart client applications are essentially mobile and do not require wireless communication at all. Integration to the enterprise is typically provided via a synchronization process whereby the client application communicates data to back-end data sources through a synchronization server. This communication can occur over wireless or wired connections such as TCP/IP, serial, or USB. The smart client architecture is depicted in Figure 1.

In order to deploy smart client applications, the client device has to have a mobile operating system that supports custom applications. This usually eliminates mobile phones from the smart client deployment options, leaving smart phones, PDAs, handheld PCs, and laptops. The common approach for developing smart client applications is to use libraries native to the mobile operating system, although alternatives such as Java are also available. Let us look at each component of the smart client architecture, in turn in the next subsection.

❖ Smart Client Architecture Components

The following are key components of the smart client architecture:

1. Smart Client

The smart client application is where client-side business logic is executed. The application itself is either a native executable or Java application that is deployed to the mobile device. To provide offline data access, mobile data store products are incorporated into the application. Many data store options are available. To communicate to the enterprise systems, a synchronization or messaging layer is also part of the smart client application.

2. Synchronization Server

Data is sent from the client application to the synchronization server. This can occur over a wireless or wired connection to the server. From there it is then communicated to the enterprise data sources. The synchronization server, with its

associated logic, is responsible for ensuring that the minimal amount of data is transferred and that any conflicts are detected and resolved. It also provides the communication layer to enterprise systems.

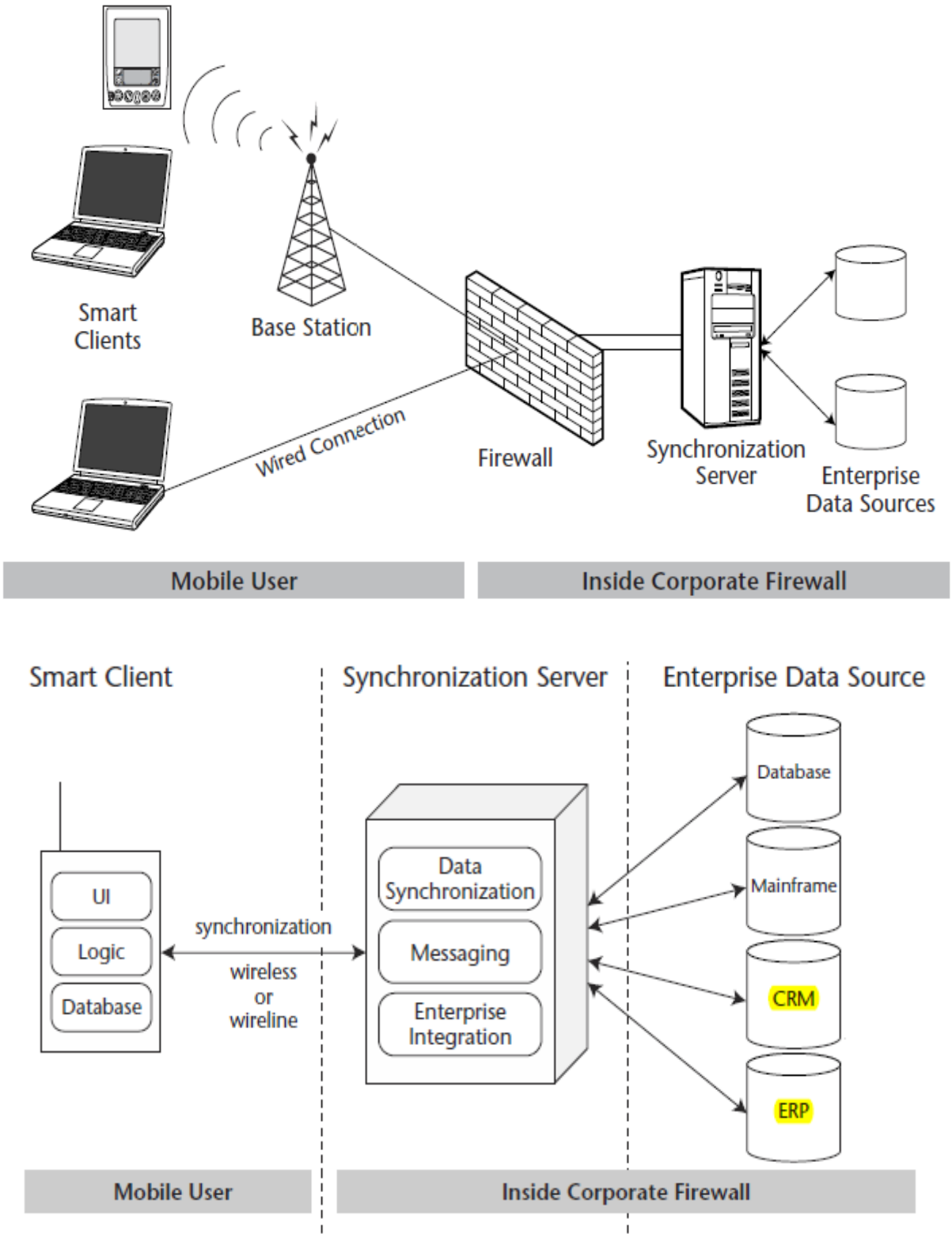

Figure (1) Smart Client Architecture

3. Enterprise Data Source

The synchronization server will access the enterprise data source using the preferred access mechanism. The access to the enterprise source may occur during the synchronization process if it is imperative for the smart client to receive feedback from the synchronization.

This is a simple process and requires that the client connection to the synchronization server remain active until the enterprise is finished processing the data. It is also possible to synchronize data to a corporate data buffer and return pending status for any operations that need to be completed. The enterprise system can poll the data buffer at its convenience for information to process. On the next smart client synchronization, the status of all pending operations are downloaded. This provides for a much shorter connection time to the server, and a greater number of clients can connect, but at the cost of end-user convenience.

One of the most common concerns about smart client applications is **software deployment**. Manufacturers of products that specialize in mobile device management have addressed this in the enterprise space. These software packages allow a central server to manage all aspects of application delivery and maintenance. For the consumer market, **over-the-air (OTA)** provisioning is a key enabler of smart client computing. OTA provisioning enables consumers to purchase and install applications directly over a wireless connection; no companion PC is required. Many wireless operators and device manufacturers are setting up digital malls that users can browse to find a variety of applications. When they find an application they like, they download and install it on the device. A nominal charge [usually between $1 and $5 (U.S.)] will be added to the user's airtime bill. These malls are very attractive for developers who require a distribution channel for their applications. The most common examples are **games**, but utilities, music, and other software applications all are benefiting from this type of distribution.

❖ Advantages of Smart Client Applications

The following are key advantages of smart client applications:

A. **Always-available data**. When no wireless coverage is available, the user is still able to access and modify enterprise data to be synchronized later. This overcomes one of the most significant limitations experienced with wireless Internet applications.
B. **Rich user interface**. Sophisticated displays with graphics and charts can be developed. This type of user interface is more similar to what users expect based on desktop experiences. In the case of Pocket PC devices, a subset of the Windows functionality is available, allowing for rich graphical components.
C. **Performance**. How the interface is "served up" is determined by processor speed and data access, which are known qualities. No performance is lost in having to continually communicate over wireless networks to access data.
D. **Distributed computing**. The execution of business logic and data processing occurs on the client devices, reducing the load on the server machines. This means organizations have to purchase less server-side hardware.
E. **Security**. Implementing a solid end-to-end security scheme is easier than with wireless Internet architectures, as you have the ability to control data on both ends of communication. There is no dependence on third-party systems such as WAP gateways.
F. **Cost**. Wireless airtime fees can be reduced or even eliminated with smart client applications. Only the data pertaining to the specific mobile user has to be transferred between the smart client and the synchronization server. This results in a minimal amount of data being transferred, thereby lowering wireless airtime charges. The synchronization can also occur over wired networks so that wireless fees can be avoided altogether.

❖ Disadvantages of Smart Client Applications

The disadvantages of smart client architectures are:

A. **Enterprise integration**. Because smart client application can provide advanced functionality on the client device, many organizations end up enhancing the client application beyond the original design. While the client-side component may not be complex to develop, the enterprise integration component will often take more time and resources than expected. (In a way, this is both an

advantage and a disadvantage since the result is a more capable client-side application.)

B. **Application deployment**. Remote deployment and management of the smart client applications may seem a trivial or end-user-driven process, but it can become an IT burden when there are a lot of remote users.
C. **Mobile viruses**. Introducing a mobile operating system opens the doors to viruses. This can be addressed by software in the application deployment phase; nevertheless, it does add another item of concern.
D. **Development complexity**. Programming in the native language of the mobile device requires learning a new client-side API. If you are developing for multiple mobile devices, with different operating systems, this can turn into a significant challenge. Java and other cross-platform tools are on the market to help address this issue.
E. **Multiple development cycles**. When multiple devices and mobile operating systems are targeted, separate development and testing phases may be required for each device involved. This can be a drain on resources and result in drawn-out development cycles if not planned for ahead of time.

❖ Application Examples

In general, a smart client architecture deserves strong consideration for any application that requires offline data access. Here are a few example applications that are particularly well suited for a smart client architecture:

A. **Sales Force Automation (SFA)**. One of the most important assets for a Salesforce is current information. The difference between making a sale or not often depends on having immediate access to information while the salesperson is meeting with the customer. The best way to ensure that the sales staff has this information is to deploy a Sales Force Automation (SFA) application with persistent data storage.

   SFA applications give sales representative immediate access to important sales information, such as the customer's contact information and order history, product catalogues, price lists, and inventory lists. With this information on hand, the sales rep can put together quotes with delivery dates on the spot.

Once the order is placed locally on the device, the rep relays the order to the back office through synchronization.

B. **Field Force Automation (FFA).** Field service workers can benefit from persistent data access in many ways. By empowering service workers with the information they need, they can improve all aspects of their jobs, resulting in higher productivity and increased customer satisfaction.

C. **Healthcare Applications**. Medical professionals, such as doctors and nurses, can realize many benefits from smart client applications, in particular, the availability of up-to-date, accurate, patient information at their fingertips. With this information, medical practitioners can diagnose conditions more effectively and accurately.

D. **Data Collection Applications**. An entire breed of applications has been designed for data capture. These applications allow businesses to streamline existing data capture processes. Many of the most recent devices are very well suited for this task. The rich user interfaces and add-on peripherals increase their usability. For **inventory management applications**, having a built-in barcode scanner allows for easy capture of merchandise, asset, and document information. This information is recorded without data entry errors and can be synchronized directly into back-end systems.

ii. Messaging

Messaging applications can take many forms, ranging from email to alerts and notifications to application-to-application messaging. In some cases messaging is used as an enhancement to an existing mobile application; in other situations, it is itself an application architecture. Numerous messaging technologies are available, so it follows that there are many uses for messaging. However, for our purposes here, **three categories of mobile messaging** are of interest to us. The following is a brief description of each:

1. **User-to-user messaging**. Messages can be sent from one user to another using a variety of mechanisms, including email, paging, and wireless text messaging such as the Short Message Service (SMS) or Instant Messaging (IM). Richer messages

that include graphics and formatted text can be sent using the Enhanced Message Service (EMS), while multimedia content can be sent using the Multimedia Message Service (MMS). These forms of messaging can also be generated by server-side processes as a means of information dispersal.

2. **Notifications and alerts**. Messages that are **urgent in nature** can be pushed to mobile users on their wireless devices. This allows corporations to ensure that information is received in a timely fashion. These messages can contain **a URL link** to a wireless Internet site where the user can obtain additional information. These types of messages are called **actionable alerts**, since the recipient performs an action based on the message content. The two leading technologies for notifications and alerts are ***Handheld Device Markup Language (HDML) alerts*** and ***WAP Push***.
3. **Application-to-application messaging**. In many cases, user interaction is not required for the message to be successful. Enterprises can communicate data directly from an enterprise server to a client application without user interaction. This can be useful to enhance smart client applications with server-initiated synchronization.

The first two types of messaging in this list do not require additional software to be deployed to the client device. For user-to-user messaging, the client software is typically provided by the device manufacturer; and for notifications and alerts, this capability is part of the microbrowser on the device. From the users' perspective, when they receive these types of messages, they view the information and react accordingly. This type of messaging is relatively straightforward to implement.

Application-to-application messaging is more involved. It requires software to be deployed on a device that understands the communication stream between the client and the server, as well as how to operate over a wireless or wired network. The client application typically contains a message queue that can store information. Figure (2) illustrates the application-to-application messaging architecture. Each component of the application-to-application messaging architecture is described in the next subsection.

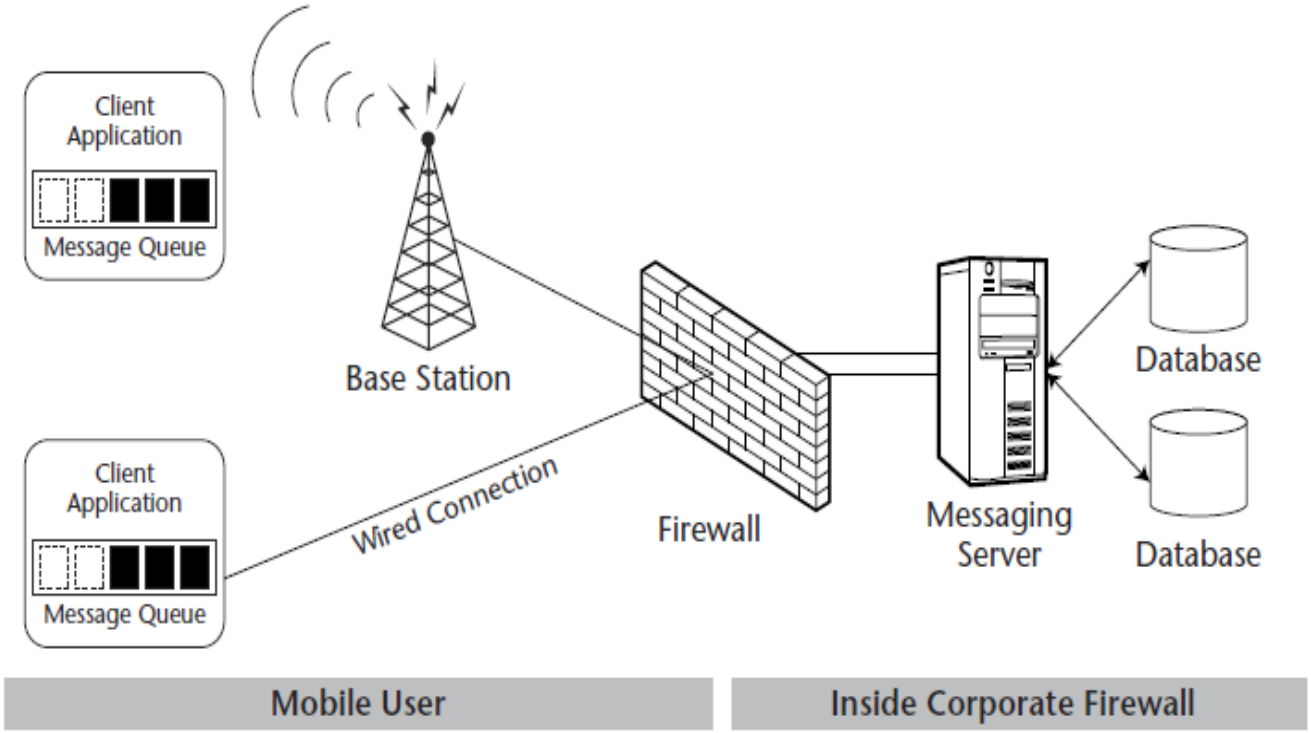

Figure (2) Application-to-Application Messaging Architecture

❖ Application-to-Application Messaging Architecture Components

Here are the main components of an application-to-application messaging system:

1. **Messaging client**. The client application contains message queues as well as client side logic. The message queues can store incoming and outgoing transactions for future access. For example, if an application attempts to send data to the server when a connection is unavailable, it can be stored in an outgoing queue and be sent automatically when the user establishes a connection to the server. This technique is called store-and-forward messaging. The messaging client can be used either on its own or in conjunction with other smart client features such as a persistent data store. The application itself can usually be programmed in languages specific to the operating system or in Java.
2. **Messaging server**. The server component is the part of the system that communicates with the messaging client, as well as the enterprise systems. The industry name for messaging servers is Message Oriented Middleware (MOM). Many of these systems are built on the Java Message Service (JMS).

JMS provides a reliable and scalable base platform with store-and-forward capabilities. This form of messaging is useful for m-business applications because the sender of the message does not have to wait for the recipient to receive the information, allowing him or her to continue working while the message is routed and acted upon.

3. **Enterprise data source**. The messaging server can interact with a variety of back-end systems, including databases, business applications, and other messaging systems. This integration can use the preferred access technique for the enterprise system.

The store-and-forward capability is not unique to application-to-application messaging. The server components of both user-to-user messaging and notifications and alerts can implement store-and-forward as well. This feature is especially important when communicating over a wireless network as it helps address the issues of intermittent connectivity, dropped connections, and coverage.

Many types of applications take advantage of messaging technology. For corporate applications, messaging is often used in conjunction with other technologies to provide access to enterprise data. Application-to-application messaging excels in this arena, often as an enhancement to smart client applications, whereas, HDML alerts and WAP Push are very valuable for enhancing thin client applications. SMS can be a valuable service for many businesses, but it is not typically integrated into corporate applications.

Over 80 percent of all SMS messages are sent from consumer applications. Over time, more business may adopt text messaging, but the initial market focus is clearly on the consumer. Next, we will examine the advantages and disadvantages of messaging architecture.

❖ Advantages of Messaging

Here are the advantages of messaging:

A. **Push capabilities**. Wireless Internet and smart client applications can be enhanced with push messaging functionality. For wireless Internet applications, the message may contain a URL where the user can go to get more information. For smart client applications the push message could notify

the user that he or she should synchronize the client data; or in some cases, the message may actually be able to initiate the synchronization automatically from the server.

B. **Store-and-forward**. When working over wireless networks, an important capability is for messages to be stored in a queue and delivered when connectivity is available. On the client, it means users can work as if a network connection is unavailable, since the information will be forwarded to the server at a later time. On the server, it is beneficial because messages will be sent to users when they connect to the network, offering a form of guaranteed message delivery.

C. **Personalized data delivery**. The leading messaging products give end users the ability to register their messaging-enabled devices and schedule the delivery of messages to a particular device. For example, the schedule might dictate that between 9:00 A.M. to 5:00 P.M., a message should be delivered to an email account, but between 5:00 P.M. and 11:00 P.M., it should be sent as an SMS text message. In this way, users can receive messages at the location that is most convenient for them.

D. **Wired and wireless communication**. Application-to-application messaging can operate over both wired and wireless networks. This gives companies the option of providing wireless access to the application users. It also allows application users to connect through a wired connection when wireless coverage is not available. User-to-user messaging and notifications and alerts typically require wireless networks for communication.

❖ Disadvantages of Messaging

Messaging is an optional component for many applications, so typically it is incorporated into applications only where there is a clear benefit. For this reason, there are not many disadvantages to using messaging, other than when it is simply not required. However, messaging does add complexity to the development of the mobile application. For application-to-application messaging, development is required on both the client and the server. For text messaging and notifications and alerts, most, if not all, of the development occurs only on the server. In either case,

incorporating messaging functionality into an application does require additional development and testing resources.

iii. Mobile Operating Systems

The mobile operating system (OS) market is vastly different from the desktop operating system market. In the mobile world, there is no clear leader in the operating system space. The **Palm OS** has a large amount of market share in the *consumer space*; **Windows CE** has been very successful in the *enterprise market*; and the **Symbian OS** is dominant in the *European market*. In addition, a following is growing for both **Java-based** operating systems and **Linux**.

The operating system is an important aspect of mobile application development. It is one of the main factors that will dictate whether you can develop a smart client application. It also plays a role in the ***development tools***, ***prepackaged software***, and ***device support*** that will be available to you. We will take a look at five main operating environments: Windows CE, Palm OS, Symbian OS, Embedded Linux and Java.

❖ Windows CE

Microsoft has made great strides in the Windows CE operating system, both in terms of functionality and usability, as well as in market share. Windows CE .NET, Microsoft's current operating system, delivers the most complete mobile operating system available. Windows CE **enables multitasking**, allowing the user to work on one application while another is executing in the background. For example, you could be working on PocketWord while synchronizing data in the background. ***PocketPC*** 2002 devices use customized versions of Windows CE, and look as though they will be the market leader for the enterprise market, and possibly the consumer market as well. But the situation was not always so positive for Microsoft.

The Windows CE operating system was designed for Handheld PC devices, which, size-wise, are in between PDAs and laptops. Even though many device manufacturers supported Windows CE, it was given a very cool reception by both the consumer and enterprise markets. It did not look as though Microsoft was going to reproduce its desktop OS success with Windows CE. Indeed, many consumers found the user interface too cluttered and complex. Microsoft found itself in an

unfamiliar position, trailing both Palm OS and Symbian OS (at the time called EPOC) in popularity. It had to come up with something better.

In 2000, Microsoft released a version of Windows CE that garnered some attention. Windows CE 3.0 was released when the PDA market was becoming more attractive to enterprises considering building line-of-business applications. For these applications, corporations were looking for an operating system that would allow them to run existing Windows-based software and to create sophisticated data-driven applications. Windows CE fit that description. Even though the popularity of the Palm OS was at an all-time high, Microsoft made some early penetration into the enterprise market during the year. By 2001, Microsoft found itself in a more familiar position, often being the OS of choice for new business application development.

Why did Windows CE 3.0 succeed? For many reasons. It ***comes with many familiar applications***, including PocketWord, PocketExcel, PocketOutlook, and Pocket Internet Explorer. In addition to its office products, Windows CE also features ***built-in multimedia capabilities for both audio and video***. These capabilities make it easy for existing Windows desktop users to move to the Windows CE operating system. They can maintain the rich level of functionality they are used to, only on a smaller device. The progression of Windows CE does not end there, Microsoft recently released many of Windows CE versions that have a high popularity.

❖ Palm OS

Palm OS has experienced tremendous success in the PDA market. In the early days of PDAs, Palm established itself as the market leader, capturing nearly 75 percent of the worldwide mobile operating system market by early 2000 (IDC, June 2000). Since then, it has continued to be a dominant player in the market. Lately, however, it has faced some difficult times.

Palm has not had the same level of success in the enterprise market for new application development as it did with consumers. In the early days of PDA application development, many organizations chose Palm OS as a deployment platform because the devices were readily available and many employees were already familiar with them. This situation resulted in a wide range of applications being developed for Palm OS. As the requirements for business applications became more complex, however, Palm OS was found to be unsuitable for many of these tasks. For example, prior to the release of version 5.0, Palm OS had single-tasking

capability only; meaning users could perform only one task at a time on the device. Thus, if they wanted to download a file, they had to wait for the download to complete before they could move on to other tasks. Many corporations found this to be a significant limitation. When, with version 5.0, Palm introduced some multi-tasking capabilities to the platform, it improved the usefulness of Palm OS for more complex applications. More enhancements for enterprise applications are also expected in Palm OS 6.0. It's important to point out that Palm hardware devices and Palm OS used to be considered one and the same. This changed when Palm started licensing Palm OS to third party device manufacturers. The first licensee was Handspring; many others followed suit, including IBM, Symbol, Kyocera, and Sony. Palm's goal by separating the hardware and software divisions was to maintain the widespread usage of Palm OS while avoiding the same problems Apple confronted as the sole manufacturer of both the hardware and the operating system software for the Macintosh. To date, it looks as though Palm is on the road to success in this regard. Handspring, for example, in 2001, released a device on the market called Treo, which incorporates a phone with wireless modem, Palm OS, email, wireless Internet access, SMS text messaging, and a full thumb keyboard, all in one device. Samsung and Kyocera have similar offerings. Palm OS 5.0 also includes new features that make it more suitable for enterprise customers. The most significant of these is the move to the ARM series of processors. The result of this move is a significant increase in the overall performance of Palm OS, allowing for the development of more advanced, feature-rich, enterprise-level applications. At the same time, version 5.0 also added the aforementioned multitasking capabilities, additional security for encrypting private data, new expansion slot capabilities, and new wireless capabilities for easier access to the Internet and email systems; it also enhanced the level of color support.

Moreover, the new features of the Palm i705, including wireless support for always on wireless networks, Bluetooth, and clip-on modems, will make the Palm OS a formidable challenger to Windows CE in the enterprise space. The only concern is that these new capabilities may have come too late; Windows CE had two years to become established before Palm OS delivered these new enterprise-level capabilities. To further distinguish their enterprise offerings, in the fall of 2002, Palm introduced two new product families: Tungsten and Zire. The Tungsten family of products is targeted at the enterprise market, providing powerful solutions for mobile professionals and enterprise work forces; the Zire family is focused on the

consumer market, providing affordable options for individuals to organize their schedules and contacts. These new offerings should help Palm regain the momentum it had in 2000/2001. Recently, Palm has a good efforts in the mobile OS market.

❖ Symbian OS

The Symbian gained its popularity in the mobile phone sector; in fact, Symbian is an operating system developed exclusively for mobile devices. As phone capabilities increased, a more functional operating system was required. Proprietary operating systems from the device manufacturers were no longer practical, so a group of the largest cell phone manufacturers, including **Nokia**, **Ericsson**, **Motorola**, and **Matsushita**, together with **Psion**, ***formed a joint venture known as Symbian***. The goal of this partnership was to create a standard operating system for smart phones and PDAs based on **Psion's EPOC operating system**.

Indeed, Symbian OS is the name assigned to the latest versions of the EPOC operating system. Versions prior to 6.0 are called EPOC, while later versions are called Symbian OS. Symbian OS version 6.0, which made some radical improvements over the previous version, was released in 2000. The design goal was to bring together various forms of communication protocols, including TCP/IP, WAP, GSM, Bluetooth, IrDA, as well as serial connections, while allowing applications to be presented using C++, Java, WAP, and other Web protocols. By linking the communications protocols and applications, Symbian hoped to achieve close integration of contact information, messaging, browsing, and wireless telephony. In 2002, Symbian OS version 7.0 was released. This version was designed for the unique requirements of advanced 2G, 2.5G, and 3G mobile phones. Some of the key features in this release included over-the-air data synchronization using SyncML; full-strength security using HTTPS, WTLS, and SSL; enhanced multimedia capabilities; support for advanced messaging, including MMS, EMS, and SMS; and a complete suite of application engines for contacts, schedules, Web browsing, and system control. These features make Symbian OS a leading choice for smart phone implementations.

In order to target the full range of mobile devices, there are a range of different platform models that can be used on mobile devices. The original two are the

reference designs: **Quartz and Crystal**. Quartz is targeted **at PDA-sized devices** (320×240 pixels), **while Crystal is targeted for communicator type devices** (640×200 pixels). More recently with the release of Symbian OS 7.0, two other user interfaces called **UIQ** and **Series 60** were introduced. UIQ provides a customizable pen-based user interface for media-rich mobile phones. It was designed to take advantage of the large, touch sensitive color displays found on 2.5G and 3G handsets.

❖ Embedded Linux

Linux is a free, open-source, UNIX-like operating system. It actually refers to a combination of two portions: Linux kernels and Linux applications. Linux kernels are developed and maintained by a group of people led by Linus Torvalds, creator of the Linux operating system. A vast number of free, open-source applications have been developed and are in use on various Linux systems. Source code of Linux kernels and applications are mostly available under **Gnu Public License** (GPL). The GPL states that anyone making use of the GPL-protected source code for public release must also ensure free and open access to the code under the same terms of the GPL. A number of Linux distributions (i.e., bundles of a Linux kernel and selected applications) have been provided by some companies such as **RedHat** and **SuSe** Linux. Thanks to the support of such companies as IBM, Oracle, and Novell, Linux is generally viewed as a serious server OS alternative to Windows and proprietary UNIX such as SunOS. It also has potential for the desktop OS market.

Linux is making giant strides in the embedded system market as well, powering up a broad range of network equipment, consumer electronics, industrial facilities, and mobile devices. In particular, several Linux distributions for PDAs are free for download on the Internet. Aside from those offerings from Linux enthusiasts, commercial Linux systems for mobile devices such as Monta Vista Linux are available in the market. Some PDAs and cell phone manufactures, especially those in East Asia, have begun to release Linux-based cell phones and smart phones. For example, NTT DoCoMo will use Linux on its 3G cell phones, and Chinese telecom equipment manufacturers have selected Linux as the embedded operating system for 3G TD-SCDMA cell phones. Numerous cell phone manufacturers such as Motorola and NEC are developing Linux smart phones.

Embedded Linux systems could be either hard real time or soft real time. Hard real time means the system must respond in a deterministic way every time a relevant event occurs. Examples of hard real time systems are autonomous control systems that require a real-time operating system (RTOS). In soft real time systems, quick responsiveness is desired but not guaranteed. Mobile devices used by consumers generally fall into the soft real time category. The latest Linux kernels support both hard real time and soft real time applications, due to the preemptive kernel design and the so-called O(1) process scheduler that utilizes a special task queue to achieve constant time scheduling. Below is a list of characteristics of embedded Linux systems:

1. The monolithic kernel supports multitasking and multithreading and can be tailored for different applications scenarios.
2. The open-source community provides support for the latest network technologies, including Bluetooth, wireless LAN, and wireless sensor networks.
3. Applications under GPL licenses are modifiable and extensible.
4. License fee is low or nonexistent.

Because Linux can be customized to fit into different mobile devices with various hardware configurations, it is of great importance to standardize hardware and software interface and interoperability between conforming devices. To this end, some companies have formed the Consumer Electronics Linux Forum (CELF) as an effort to promote advancement of open embedded Linux systems for consumer electronics devices. Another industry organization, the Embedded Linux Consortium (ELC), is focused on general embedded Linux standard development.

Due to the recent development of Linux smart phones and wireless network equipment, the future of Linux in mobile computing seems quite exciting. For Linux to reach its full potential in the mobile sector, it must address “fear, uncertainty, and doubt (FUD)” with regard to how Linux will perform as a reliable, robust, and yet low-cost platform, let alone the recent litigation addressing UNIX intellectual property between SCO Group Inc. and some UNIX and Linux providers.

❖ Java and J2ME

When the Java programming language was initially developed by Sun Microsystems in the early 1990s, it was aimed at the embedded device market. The large number

of devices with different chipsets made programming difficult with Assembler or C. The idea behind Java was to design a language that could be ported to different architectures and operating systems without reprogramming. It seemed like an ideal fit for the diverse device market, but device manufacturers were not interested. The result was a language that could run on multiple chipsets with multiple operating systems but that did not generate any interest in its target market. Fortunately, Sun Microsystems did not give up. With the growing popularity of the World Wide Web, the need for such a language resurfaced. At the time, Internet content consisted of static HTML pages that did not generate much viewer excitement. Missing was dynamic content. But how could it be programmed? What was needed was a language that could be run on a variety of operating systems based on different hardware architectures—seemingly, a perfect fit for Java. Using the Java platform, developers could create little Java applications called applets that could run inside of a Web browser and provide a rich, dynamic user interface for Web applications. The version of the Java platform that provides this interface is the Java 2 Platform Standard Edition (J2SE).

Java applets failed to catch on as many expected. The general feeling was that client-side Java applications were too slow and too large to download. Around that time, Sun Microsystems was also working on a server-side API for Java, now called the Java 2 Platform Enterprise Edition (J2EE). Rather than developing server-side Web applications using proprietary software, developers can use the J2EE specification to develop applications that can be ported to a variety of servers across a variety of operating systems. The Java platform finally found a large audience of companies and developers ready to take advantage of the capabilities offered by the language. As you are probably aware, most of today's application server vendors support the J2EE specification, and many wireless server-side applications are based on J2EE as well.

With the success of the J2EE specification, in 2000, Sun Microsystems decided to give Java another try on mobile devices. This time the company implemented another version of Java called the Java 2 Platform, Micro Edition (J2ME). J2ME targets the embedded and consumer device space, which ranges from Java Smart Cards to set-top boxes and smart appliances. Java smart phones and PDAs are included in that

range. J2ME applications maintain the same core features of J2SE and J2EE: code portability, cross platform support, along with the Java programming language. In addition, J2ME applications are upwardly scalable, to work with J2SE and J2EE platforms.

That said, it's important to point out that the J2ME architecture is somewhat different from J2SE and J2EE as it incorporates a few new concepts. J2ME is based around configurations and profiles. A configuration consists of a Java Virtual Machine (JVM), core libraries, standard classes, and APIs. Two configurations exist for J2ME today: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). Each is designed for a set of devices with varying capabilities. The configuration provides the lowest common denominator set of classes and the building blocks on which profiles are created. A profile sits on top of a configuration and provides a complete set of APIs for a narrower set of devices. The combination of a configuration and a profile results in a complete J2ME platform.

The CLDC is based on the K Virtual Machine (KVM) and associated libraries. It is targeted at highly constrained devices with either 16 or 32-bit CPUs and less than 512 KB of total memory available for applications. The current profiles for CLDC include the Mobile Information Device Profile (MIDP) and the PDA Profile. Devices that fit into this category include smart phones, two-way pagers, and low-end PDAs. For devices with more robust resources, the CDC is the configuration of choice. These devices typically run on 32-bit processors and have over 2 MB of memory available for applications. The base profile for the CDC is the Foundation Profile. On top of the Foundation Profile is the Personal Basis Profile and the Personal Profile. The relationship between these profiles is somewhat unique. The Personal Basis Profile actually includes the Foundation Profile and is a subset of the Personal Profile. The Personal Profile provides a similar level of client support to J2SE, but has a smaller footprint. The relationship between the J2ME configurations and profiles is shown in Figure (3). Java is listed in the operating system section because you can create smart client applications using Java technology and deploy them to any operating system that has a JVM, which includes all the OSes listed previously. This means that rather than developing a native application that will work only on the targeted operating system, you can develop your application using Java and be able to target a variety

of operating systems without having to rewrite the application. It sounds like the ideal solution. However, it is important to keep in mind that J2ME still suffers from some of the same problems experienced over five years ago with the Java on desktop clients; specifically, performance is not always adequate and the portability is not 100 percent. For most applications, extensive testing is required for each platform to which the application will be deployed.

As device manufacturers continue to add more capabilities to handheld devices, thus increasing their performance, it is expected that Java technology will become a viable option for creating sophisticated, data-driven smart client applications.

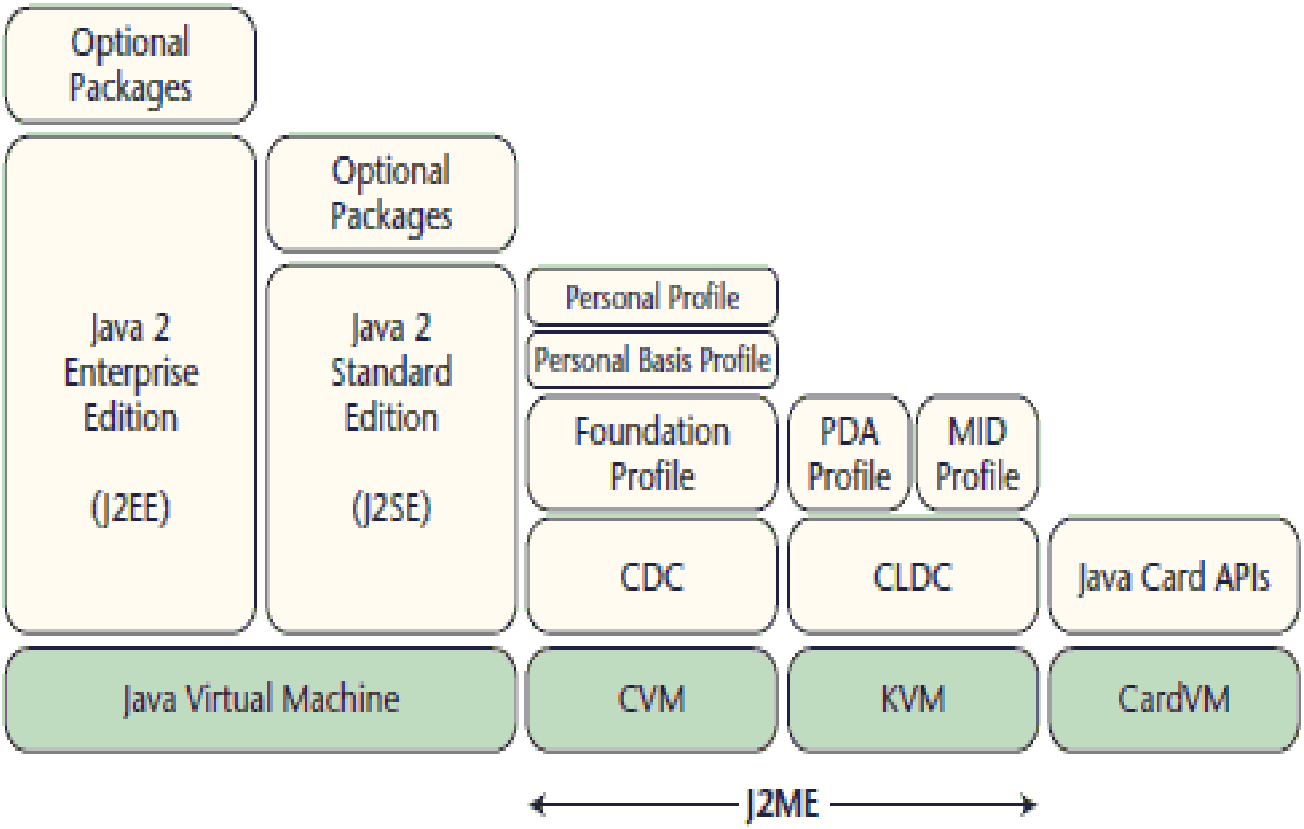

Figure (3) J2ME Architecture

Pocket PC
Start
2:43
Friday, October 18, 2003
Owner: George Jiang
Eva's Birthday
Team Meeting
10:30AM-11:00AM (Conf room)
Lunch w/Sam & Griffin
12:00PM-1:00PM (Cafe)
Quarterly Business Review
ActiveSync: 18 Unread Messages
1 Active task
New
TOSHIBA

symbian
OS