

Development Considerations for Mobile Enterprise Applications

Abstract

The rapidly evolving need for enterprise mobile applications today presents a number of new challenges to the systems integrator, independent software developer and corporate IT department seeking to deploy products in this space. This paper highlights key issues that developers should consider before embarking on development of mobile enterprise applications.

The Need for the Mobile Enterprise

Enterprise mobility applications are now utilized across most industries and functions, including transportation, health care and pharmaceuticals, public safety and security and military operations. Mobile workers utilize them for sales force automation, field force automation and distributed telemetrics, as well as healthcare delivery, pharmaceuticals testing, project accounting, expense reporting, inventory management and many other critical tasks.

In order to optimize these increasingly prevalent mobile applications, field professionals require access to enterprise data while removed from a network connection. Complex, distributed applications permit manipulating local data residing on the mobile platform, while occasionally interacting and downloading information from back-end systems such as CRM, SFA, and ERP solutions. They automate workflows, enhance research and data collection, increase mobile employee productivity and give the enterprise faster access to information from the field.

Such activities are typically carried out using notebooks, laptops and memory-constrained devices such as PDAs and Tablet PCs. Applications must also be miniaturized for a mobile environment and any embedded or related technologies, such as a database, must be exceptionally small as well. Network connectivity may occur either asynchronously, where information is cached until connectivity is available and synchronization can occur, or synchronously with continuous access to a wireless data network.

Evaluating the Architectures for Mobile Applications

Computing and information management are constantly evolving, with legacy architectures co-existing along side newer ones in today's large corporations. Different architectures are developed to meet new business needs and to exploit more efficient and less costly resources. The chart in Figure 1 characterizes the differences in the predominant data architectures that have emerged in corporate computing.

Figure 1: Data Computing Architectures

Computing	Process Distribution	Network Dependence
Server-Centric (Mainframe)	All User Interface (UI), application logic, and data management resident at server.	Critical
Network-centric (Thin-client)	UI at the client, application logic distributed between client and server. Data management resident at server(s).	Critical
Peer-to-peer	Client-resident UI. Application logic and data management distributed across connected peer clients.	Critical
Client-Centric	All UI and application logic at the client. Data management distributed between client and servers.	Minimal

The selection of an appropriate architecture for enterprise mobility applications requires deliberation between the competing constraints of this environment: network performance/reliability and device performance. Mobile memory constraints directly impact the performance of mobile application logic executed on the device.

Each architecture has advantages and disadvantages, but all serve present-day business needs. Network-centric computing has dominated recent rapid growth in technology because it optimizes distributed resources while balancing efficiency of management and control. Peer-to-peer can be viewed as a variation on network-centric computing across heterogeneous connected clients.

These architectures also differ in their relative dependence on a network. Less common now is an implementation of server-centric computing that necessitates a network to relay information to "dumb-terminals," as in a central reservation system. Network-centric and peer-to-peer architectures can only leverage distributed resources when a network is present. For any of these architectures, the failure of any part of a network results in failure of the system.

A thin-client, network-centric architecture predicates an "always connected" network. Its advantages include centralized application management, a common data store and minimal use of mobile device resources. Disadvantages include connectivity cost, network speed, network reliability and multiple potential points of failure. Reliability, speed and cost of the thin client mobile architecture are therefore directly linked to the speed and cost of the wireless network.

Client-centric architectures, on the other hand, place minimal reliance on the network, as the client holds all application logic and temporarily caches local data. Known as "occasional connectivity," this networking structure supports periodic synchronization of local data to server databases, as well as facilitating activities such as distribution of application upgrades to field and mobile workers. Enterprise applications now more commonly rely on such architectures, as they are more robust and reliable. The occasionally connected architecture allows enterprises to provide availability, reliability and speed across a broad geographic area.

Development Considerations

In many ways, the considerations for development of mobile applications are congruent with those for web or PC application development. This section introduces specific areas the mobile application developer should evaluate when selecting technology.

Platform

The mobile application developer currently has a significant range of choices when selecting an application environment. These include a wide variety of physical forms, user interfaces, memory, processor types and speeds and operating systems.

While these choices provide the developer with more flexibility in selecting function-specific combinations, the development of general-purpose applications across devices becomes more complex. Adoption of standards such as Java and SQL, the most widely accepted standards in the industry, allows the developer to conform to corporate infrastructure and lower development costs.

Hardware

As discussed earlier, there is a wide range of mobile devices available in the mobile marketplace. The physical characteristics of these devices are governed by their function and cost. In general, however, their common attributes include:

- Limited I/O, particularly in contrast to a PC
- Limited battery life
- Limited processing speed
- Limited memory
- Limited wireless bandwidth

Based on these key points, serious consideration of the performance characteristics of the target platform must be undertaken before deploying an application in this environment. Trends in system performance are expected to improve for these devices as it did during the evolution of the personal computer. However, due to battery limitations, the rate of improvement is not likely to duplicate that of the PC. Selection of the appropriate hardware platform should be evaluated across cost, suitability to requirements and performance expectations.

Operating environment

Today's major operating systems for mobile environments include Microsoft Windows, Windows CE, Palm, Psion, Linux and many other smaller providers. Many factors in this multi-platform environment can be mitigated by the use of Java. Without the use of Java, the following factors must be considered:

- Market share of hardware device. This impacts the range of choices for physical devices and the availability of skilled developers for the platform.
- Strength of integration on target hardware platform, including performance, footprint and robustness.
- OS level features for event handling, I/O and other common tasks.
- Development, testing and simulation tools availability.
- Partnership strength between OS vendor and device manufacturer. This impacts product integration, support and other factors.

The risks of selecting the correct vendor can be offset by the adoption of a standard operating platform such as the Java Virtual Machine (JVM). Through abstraction, applications can be insulated from a specific OS environment and hardware configuration and will operate on any platform that supports Java.

Java Virtual Machine

The Java Virtual Machine (JVM) allows software applications to be developed independently from the nuances of a specific operating system. Programs written in Java can be executed on any platform supporting a JVM, lowering development costs by avoiding the need to recompile for a variety of platforms.

The available libraries for the original Java specification have diversified into three related editions – Java 2 Enterprise Edition (J2EE™) for server-side application code, Java 2 Standard Edition (J2SE™) for client-side application code, and Java 2 Mobile Edition (J2ME™) for the resource-constrained mobile platforms.

Within the J2ME environment, the developer will find several "configurations" or "profiles" of a specific set of Java libraries for two categories of mobile devices. The Connected Device Configuration (CDC) is generally for systems with sufficient memory to support an approximate 2MB Java footprint. The Connected Limited Device Configuration (CLDC) is generally for systems with sufficient memory to support an approximate 200k footprint. These environments can be contrasted with an approximate 8MB footprint for J2SE.

Profiles such as the Mobile Information Device Profile (MIDP) within the CLDC configuration, are tailored specifically to smaller emerging mobile devices. All support the Java runtime environment and applications developed in this language.

The Issues of Data Management

Development of information-rich mobile applications requires consideration of the mechanisms for data management and synchronization with external data sources, such as enterprise databases.

As discussed earlier, the selection of architecture directly impacts the importance of data management in the mobile environment. For the network-dependent thin-client architecture, data management at the client-side is reduced to cache management. Enterprise data management techniques suffice to support this architecture.

For occasionally connected, client-centric architectures, the storage and management of data on the mobile device becomes more sophisticated and more crucial. Local data is stored for ready access when the network is not connected. When the network is once again available, changes to the local data are reconciled with server data during a process called synchronization.

Data management for this architecture generally takes three forms: native storage (memory registers, etc.), flat file systems and relational databases.

Native storage is coded by developers to manage operations such as insertion, deletion, and modification of data as well as to ensure transaction integrity, adherence to business rules, ability to recover data and a range of other operations. For simple applications, this may cost less. However, for more complex applications, the disadvantages include an increased cost of development, time-to-market and maintenance through the necessity to create and re-create common data management functions.

Flat file systems are still used by some developers, but less frequently as mobile applications become more functional. These systems require proprietary development of such features as automatic data recovery to prevent data corruption, concurrent update capability, fast access to high volumes of data, data security such as password protection or encryption and management of complex data relationships.

By selecting relational database technology to support mobile applications, on the other hand, the developer removes the need to code and maintain these standardized functions, lowering costs and reducing development time and speeding time to market. Today's local mobile databases provide a range of features, reliability, security and storage capacity to enable complex enterprise applications.

Mobile Databases

A variety of databases are available that are designed for memory-constrained mobile platforms in client-centric environments. These are becoming increasingly powerful to enable the capabilities of the occasionally connected environment and support mission-critical applications. The size and performance of database engine should be considered when selecting this technology. It should also be taken into account that most corporations are adding mobility to existing enterprise systems. By definition, therefore, mobile databases must be able to function in heterogeneous environments, and thus should also be able to port easily to a variety of platforms.

Database scalability at the enterprise server is a well-understood factor in system development. Scalability in this sense tends to refer to management storage capacity (in terabytes), complex query performance, and volume of simultaneous users. On the other hand, although it varies by the platform, scalability for the mobile environment tends to refer to a database's ability to manage the operations of a single user in a small, resource-limited environment.

Abstraction through standards such as Java or SQL offers the advantage of allowing the software developer to focus on only supporting functions unique to the mobile application. The developer need not be aware of implementation details for common functions, which are handled by the JVM or database as appropriate. This allows application software to be less complex but rich in functionality, and reduces cost and development time.

On the other hand, coding and compiling application logic specifically for the native environment, and conducting data management tasks without a database can produce performance-optimized software at significantly increased development cost and time. As processor speeds for mobility devices increase, the performance gain for proprietary environment-specific logic will be reduced in its relative performance gain.

Developers should look for a mobile database that:

- 1) Provides a footprint that is small enough to achieve scalability and low impact on application performance
- 2) Is portable to avoid recompiling for various platforms in heterogeneous environments
- 3) Is based on the major mobility standards, especially SQL, which is used across every industry and application
- 4) Provides a strong subset of SQL queries, to avoid problems when running queries based on information from corporate databases
- 5) Is capable of multiple, simultaneous connections for more efficient upload/download
- 6) Provides good security and transaction integrity to keep enterprise information safe
- 7) Integrates easily with synchronization technologies to achieve bi-directional synchronization with corporate databases

Data Synchronization

Selection of a client-centric architecture necessitates also considering the available mechanisms for synchronization.

Developers should consider the form and storage of consolidated data at the server. Synchronization with enterprise relational systems such as Oracle®, Microsoft® SQL Server™, and types of databases is easier when local storage is also a relational database. Removing the need for data transformation reduces the complexity of the synchronization operation and the cost for implementation.

Synchronization also impacts system integration with other enterprise-level systems including message brokers, transaction brokers and application servers. Developers should evaluate the enterprise's integration requirements when selecting the appropriate synchronization technology.

Developers should look for a synchronization technology that:

- 1) Provides platform-independent, bi-directional synchronization between mobile and corporate databases to facilitate the flow of information
- 2) Offers scalable connectivity to accommodate growing numbers of users
- 3) Provides a range of tools that can monitor data flow throughout the enterprise
- 4) Is based on a publish-and-subscribe model to help users download only the information that is specifically needed, as opposed to the entire corporate database
- 5) Provides capable conflict detection and resolution tools to resolve data conflicts among concurrent multiple users
- 6) Provides good security and transaction integrity to keep enterprise information safe
- 7) Provides an API that integrates easily with databases to expedite synchronization development

Creating the Development Environment

Tools

Selection of a platform with productive development tools can influence the cost and time-to-market for applications development. Consider assessing availability of documentation, maturity of tools, efficiency of compilers and usability.

Many proprietary OS vendors either provide development tools directly or have a community of partners that do. Tools for such OS environments are nearly always specific to that platform, a characteristic that influences training costs and skills availability for that toolset.

Java development tool suites have an abundance of suppliers and many are emerging that specifically support the J2ME environment. Selecting Java as the mobile platform decreases the risk for available tools and skills, Java tools can be effectively used to develop applications for several mobile platforms.

Skills

Before embarking on any software implementation project of significance, it is prudent to assess the development skills market for the technologies selected. The availability of appropriate software engineers can be a deciding factor in the selection of products and standards for implementation.

Cost, training and experience are also important characteristics to assess when establishing a development team. The adoption of technology with a large base of experience developers and support personnel reduces risk.

The large and growing international pool of Java programmers presents a favorable market for applications developed on this platform. Best practices for the J2ME mobile environment are emerging. Mobile Java developers should first familiarize themselves with the library differences between the J2ME platform (CLDC/MIDP) and the J2SE and J2EE platforms before beginning software design.

Availability of training and third party support for the selected platform technologies plays an important role in continued maintenance of mobile software applications. For both Java and SQL, using these well-accepted technologies makes it easier to find skilled developers, and reduces cost of training and project startup.

Costs

An assessment of the financial cost for mobile project implementation can be attained through a comprehensive study of the preceding factors affecting development. Cost assessment should include weighing the cost of third party utilities, tools, databases and operating systems against the cost of in-house development. Additional factors such as training, support and availability of skills through consultancies or hires also impact project budgets.

PointBase Mobile Database Technologies

As we have seen, it is important when developing enterprise-level mobile applications to select a client-centric database solution that is Java-based, provides robust local data storage and synchronizes easily with a variety of enterprise databases. Such a mobility solution is available from PointBase, a division of DataMirror, the leader in Java data management and synchronization solutions.

The PointBase Micro database is a platform-independent, relational database written in Java and designed specifically for mobile environments. It can be embedded directly into mobile applications and is transparent to the end user. It meets the requirements for enterprise mobility applications:

- 1) It is small enough (<45KB for J2ME CLDC/MIDP and <90KB for J2SE and J2ME CDC) to run efficiently on most of today's mobile devices.
- 2) Its Java-based portability eliminates recompiling for multiple platforms and reduces the complexity of application upgrades.
- 3) PointBase Micro has a strong subset of standard SQL 92 functionality, including transactional support.
- 4) It offers multiple connections performing simultaneous tasks.
- 5) It complies with J2SE and J2ME (CDC and CLDC/MIDP) standards, and supports the JDBC programming interface.
- 6) PointBase Micro offers enhanced Java-based security for encryption and role privileges, and leverages the Extended Tiny Encryption and PointBase encryption algorithms.
- 7) PointBase utilizes its companion product, PointBase UniSync, to provide periodic synchronization with Oracle, Microsoft SQL Server, and other JDBC-compliant corporate databases.

For more scalable mobile applications that run primarily on laptops or notebooks, developers may also want to consider PointBase Embedded, a full-featured relational database written in Java. PointBase Embedded is completely portable and includes comprehensive SQL compliance, transactional integrity and zero administration.

For synchronization of both these databases, PointBase UniSync provides a Java API that allows developers to easily synchronize data between local databases and corporate back-end databases. It meets the needs of enterprise information gathering, including:

- 1) It provides platform-independent, bi-directional synchronization with corporate databases.
- 2) It supports multiple connections to both the UniSync engine and the corporate database, with the number of users limited only by the hardware being used.
- 3) PointBase UniSync offers automatic data type mapping and tracking capabilities to monitor data flows.
- 4) It is based on a publish-and-subscribe model that allows client applications to have subscriptions to published data on the server side, which can consist of single or multiple tables.
- 5) It provides sophisticated conflict detection and resolution to resolve conflicts which may occur if two different users make changes to the same row or table and then try to synchronize them.
- 6) UniSync supports data encryption across the network with standard encryption algorithms and transactional integrity to assure database consistency.
- 7) It offers a fully documented API with custom business logic that integrates with all PointBase database products as well as JDBC-compliant third-party databases.

Conclusion

As mobile applications play an increasingly vital role in corporate enterprises, the demands placed upon these technologies and their developers will only increase. Development considerations for a mobile environment are similar in many ways to current enterprise software practices. Specific issues such as resource limitations, platform variety, network unreliability and security characterize the decisions to make when beginning a mobile project. A careful assessment and prioritization of business requirements is essential to selecting appropriate technology and achieving project success.

Corporate Headquarters
3910 Freedom Circle, Suite 104
Santa Clara, CA 95054

Main. 408.961.1100
1.877.238.8798 (US and Canada)
information@pointbase.com
www.pointbase.com