Best Practices: Extending Enterprise Applications to Mobile Devices

THE ARCHITECTURE JOURNAL Input for Better Outcomes

by Kulathumani Hariharan

Summary: Extending enterprise applications to mobile devices is increasingly becoming a priority for organizations optimizing their work force. To achieve the desired result of a robust, scalable, secure, and responsive mobile solution with multiple device platform support, many components need to work together. The challenge is to seamlessly extend various flavors of enterprise applications, many based on a variety of technologies and platforms, on to mobile devices. This article outlines the components required to extend a generic enterprise application on to mobile devices, covers some best practices and recommendations, and describes a case study based on a real-world implementation.

Contents

- Mobile Solution Overview
- Extending a Product-based SFA/FFAApplication on to a Mobile Device
- Technical Case Study: Extending a CRMApplication on to Mobile Devices for a Territory Management System
- About the Author

Mobile Solution Overview

When extending enterprise applications to mobile devices, many solutions require a three-tier approach: the enterprise application itself, mobile middleware, and the mobile client application.

Enterprise Application. There are, of course, many flavors of enterprise applications that can be extended on to mobile devices, such as Customer Relation Management (CRM), Enterprise Resource Planning (ERP), and Business Intelligence (BI).

Mobile Middleware. As most enterprise applications don't have a direct way of working with devices, mobile middleware (as it will be called in this article) plays a crucial role. Some of the important features of this tier include security, data synchronization, device management, and the necessary support for multiple devices.

Mobile Client Application. The mobile client application is, of course, the software that will run on the device. There are many considerations at this tier, including data availability, communication with middleware, local resource utilization, and local data storage. In addition, many business factors need to be considered. For example, who are the target users? How critical is it to have the latest data? Are there restrictions for storing data on the device? What provisions are there in case of no network connectivity?

When selecting the platform for the device, we see three main options:

• *Online Applications (also known as a thin client).* This is client software, normally a browser, used when connectivity can be guaranteed. Without a connection, the mobile application does not work.
• *Offline Applications (also known as a thick client).* This is client software installed locally to the device that holds all required data for the duration of most operations, and synchronizes at the end of each day or a preconfigured period of time.
• *Occasionally Connected Applications (also known as a smart client).* This is client software installed locally, similar to the offline model, but where the application can update and refresh data at any point in time. The frequency of the data refresh depends on the criticality of the application.

Using the above three tiers as reference, let's now explore what this means for a product-based Sales Force Automation / Field Force Automation (SFA/FFA).

Extending a Product-based SFA/FFAApplication on to a Mobile Device

A product-based SFA/FFA is typically part of a CRM or ERP application. It's common that this type of application does not have an existing solution for mobile devices. The application's server-side front end is typically a Web-based or a rich client application, supported by a relational database with a large data store catering to the whole organization. There tend to be certain restrictions on access to this database, including the following challenges:

- Changes to schema to support mobile extension — there is often little that can be done (or should be done) to change the schema to support mobile applications.
- Data access directly from the database, and update into the tables from the mobile device — often there are several layers of communication to go through and it is not possible to access the database directly from the device.
- Understanding the schema of the data store — schemas for these types of applications are designed to be extended, and as a result can be large and unwieldy.
- Designing a staging area with a schema structure similar to the back end for data to flow to the mobile devices — creating a replica environment for development and staging can be a challenge.

Solution Introduction

When extending a product-based SFA/FFA application on to mobile devices, the challenges mentioned in the previous section need to be effectively addressed. The architecture needs to consider components that work in tandem to address these challenges.

Figure 1 shows a proposed model for the smart client application; Table 1 lists the components for both the middleware and client application with a brief description of its role as part of the solution. Note that the component list is an optional superset and specific implementations may not have all the components.

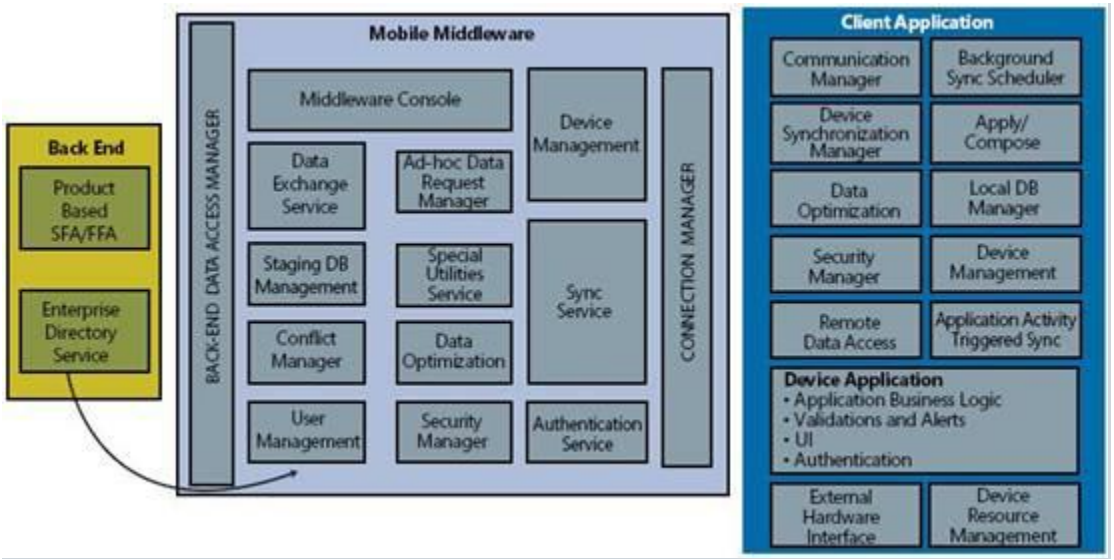

Figure 1.Components for extending a product-based SFA/FFA on to mobile device—Smart Client Approach

Best Practices

From experience, we have found the following to be best practice when creating applications based on the model outlined by Figure 1 and Table 1.

1. Use Database Stored Procedures to write wrapper code for faster data access.
2. For ad-hoc data, the data should be populated using database views for faster output to the device.
3. The staging database infrastructure could be part of the main database server for faster response to mobile devices (the benefit is dependent on the number of users and the server load at any point of time).
4. While extending data from the back end to the staging database, include only those columns and fields that are necessary on the mobile device as the same is to be extended on to the device. This will help in adhering to size constraints on the device.

5. The staging database should only have data for a limited period (two months, for example) with regular scheduled archives; constraining the size of the database will reduce seek time.
6. Use the record version number to easily track records for delta updates during synchronization.
7. Use mapping tables in the staging database to track record version to facilitate conflict resolution; for example, to impose a conflict rule, overriding a transaction record with a server-side change even when multiple changes are done on the client end. (A mapping table is a table in the staging database which contains the primary key of the back-end database table and the primary key of the record on the device database.)
8. The Data Exchange Service should be a recurring process and should be configurable in the middleware console to handle continuous changes on the back-end system and staging database (triggered from client), creating an asynchronous method of working.
9. Maintain only necessary user details on the middleware and link to the enterprise directory service for authentication and other user data. This will reduce out-of-sync issues for user information between the enterprise directory and the middleware.
10. Do not store passwords in the staging database; instead, query the enterprise directory service during authentication. This eliminates out-of-sync issues caused due to non-update of the server-side password in the middleware.
11. During synchronization, the client application should first check for application updates by sending its current version and downloading the latest version if applicable; this is an optimized mechanism for application version management.
12. Store the user device profile (device platforms and OS versions) in the user database and push version updates to the device accordingly, sending different builds to different users.
13. Maintain three tables: in-queue, out-queue, and user-wise out queue for synchronization management, simplifying queue management and optimizing the synchronization process.
14. The communication manager can be made to try alternative types of connectivity when the primary method is not available, so as to use the most efficient available network connectivity option. For example, when wireless LAN is not available, the application tries General Packet Radio Service (GPRS) network; if GPRS is not available, the client does not synchronize.
15. The background sync interval should take into consideration the number of users and the number of concurrent users the server can support. These considerations will assist in reducing the load on the server supporting the maximum number of mobile users.
16. The Device Synchronization Manager just needs to send the username, device application version, sync interval time, and the delta updates during synchronization. To reduce the number of concurrent synchronizations, the middleware should return whether an update is available and the next time for synchronization if no update is required.
17. The record state column should be maintained at record level for faster composing of data during synchronization.
18. The applications should be designed in such a way that when the battery power is low, background thread priority should be set to low, reducing CPU usage and extending battery life.
19. Develop the emulator (if not readily available) for the device type. This reduces efforts during development and testing phases. For example, use Microsoft platform builder to develop a Win CE emulator that can support features required by the application.
20. During application development, also develop simulation application(s) to test the input data and output data through peripherals attached to the device.
21. Dummy data obtained from the device manufacturer or created using device manuals will be crucial for simulation application.
22. When designing the modules for the peripherals, place hardware-specific and application-specific functions in their respective libraries so that changes to the peripherals can be made without affecting the application library.
23. Where the application consists of multiple screens having common UI parts and functionality, design a base form that contains the common elements.
24. Use frames instead of multiple forms wherever possible for faster user interface response.
25. Messages (such as error messages and alerts) should be configured in the middleware and should flow to the devices. Other configuration files should also be configurable on middleware and then pushed to the device application for use.
26. Database-specific queries should not be hard-coded; instead, the queries should be fetched from the middleware via a configuration file.

Table 1. Middleware and Client Application Components

Middleware

Component	Description
Back-end Data Access Manager	• Consists of wrapper code for calling the back-end APIs to insert, update, and delete data from the back end.
Ad-hoc Data Request Manager	• Preconfigured methods that return ad-hoc real-time data to the mobile device.
Staging Database Management	• Manages data in the staging database • Stores the replica of all transaction tables with mobile users' specific columns and data. • Handles data archiving • Handles in-queue and out-queue • Cleans up database space.
Conflict Manager	• Manages data conflicts while synchronizing with the back-end database • Monitors for conflicts like Server Wins vs. Client Wins, sharing of records by multiple users, updating a record on device when the record has been deleted on the back end and so on.
Data Exchange Service	• Composes changes from the back end for a particular user to be sent to the mobile device • Applies changes from the mobile device to the back end using the Back-end Data Access Manager • Runs as a recursive service, running regularly after a period of time (configurable) • Sends composed data to the out-queue • Picks the data from the in-queue for applying to the back end.
Middleware Console	• The user interface for configuring middleware • Used to configure modules such as user management, data subsetting, synchronization management, device management, and so on.
Special Utilities Service	• Contains business logic to do specific activities that are not a core part of the middleware, but part of the mobile solution—for example, a location-based service that gets location updates captured from the device and uses a geographic information system to map latitude and longitude and display as reports • Optional, driven by business requirements.
User Management	• Manages the mobile users using the mobile devices • User list can be linked with the enterprise directory services for using the same authentication on mobile devices.
Device Management	• Manages the devices from the server • Sends new application updates to the mobile devices • Views application logs • Explores device-related issues • Manages enforcing enterprise security policies like erasing of data on devices that have not synchronized for certain number of days.
Data Optimization	• Optimizes the way data is sent to the mobile device • Compresses data and chooses the best method for sending data based on the connection speed or type of connectivity • Uncompresses data coming from the mobile device.

Security Manager	• Encrypts data being sent to the mobile device • Decrypts the data coming from the mobile device.
Synchronization Service	• Core component of the middleware • Incoming data from the mobile device is received into the in-queue and the outgoing data is pushed to the mobile device from out-queue • Data exchange service composes changes for a particular user into the out-queue and applies the in-queue changes from the mobile device onto the back end • Background synchronization from the device; the service maintains a user-wise queue and checks for new records to be sent to the mobile device.
Authentication Service	• Authenticates the mobile user during login process and synchronization.
Connection Manager	• Handles multiple connections at the same time from mobile users up to a maximum number of concurrent users.

Client Application

Component	**Description**
Communication Manager	• Establishes connection to the network.
Device Synchronization Manager	• Authenticates with the authentication service • Sends and receives data from the synchronization service in the middleware • Downloads application updates and device management commands.
Data Optimization	• Optimizes the way data is sent to the middleware • Compresses data and chooses the best method for sending based on the connection speed and type of connectivity • Uncompresses incoming data from the middleware.
Security Manager	• Encrypts data being sent to the middleware • Decrypts the data coming from the middleware.
Apply/Compose	• Applies the incoming data • Composes changes to be sent to the back end.
Scheduler for Background Synchronization	• Configurable component schedules background synchronization from mobile device • Sends data to the server at a pre-configured interval without user intervention (automatically).
Remote Data Access	• Calls the methods defined in the 'Ad-hoc Data Request Manager' to have real-time data on the mobile device; if connectivity is not available then data existing in the device is used.
Local Database Manager	• Manages data in the device database • Applies and composes data • Cleans up temporary data from the database • Manages record state • Manages device configuration details.

Device Management	• Executes commands from middleware • Applies application updates • Locks out application if the user enters wrong password for a specific number of times • Send logs to middleware.
Application Activity Triggered Synch	• Triggers synchronization on completion of a complete business flow, ensuring that the mobile client back end are in sync.
Application Business Logic	• Lynchpin of the whole mobile solution (actual application used by the mobile users).
Validations and Alerts	• Validates the user input with some business rules • In case of non-compliance, displays alerts on the user interface accordingly.
External Hardware Interface	• Interacts with external hardware interfaces attached with the device • Necessary for data flow to external applications, such as Barcode Scanner.
Device Resource Management	• Accesses the resource state and displays alerts if user attention is required.
User Interface	• Face of the mobile solution. • Information entered by the user through the UI is validated and processed by the business logic.
Authentication	• Authenticates user with the middleware if there is network connectivity, otherwise authenticates with credentials available locally.

Technical Case Study: Extending a CRMApplication on to Mobile Devices for a Territory Management System

A top pharmaceutical company with pan-India presence wanted to improve the efficiency of its Sales Representatives working in the field. The sales representatives of the company meet physicians in hospitals and clinics to promote the company drugs, distribute samples and promotional materials and, at the end of the day, record all details through a Web-based CRM application.

The proposed handheld-based solution had the following goals:

- Improve the efficiency and effectiveness of sales representatives
- Improve productivity by 10 percent
- Reduce manual process expenses like stationary and telephone
- Provide a quick and easy user interface for successful user adoption
- Upload latest data from device on to the server so that managers can track the sales representative's work
- Download latest product inventory on the device for getting orders from chemists (only if there is connectivity).

This handheld-based solution was designed, developed, and implemented throughout India where the pharmaceutical company is located.

Problem Definition

Sales representatives of the pharmaceutical company make about 10 field calls a day to meet physicians to promote drugs, take orders, and distribute samples. A sales call typically lasts 2-10 minutes since the physicians have full schedules. In this limited time, the sales representative has to discuss the drugs, get feedback from the physician, and distribute samples, journals, and promotional materials. The information collected is captured on paper by the sales representative

who, at end of the day, enters all call-related information on to the Web-based CRM application. The sales representative's supervisor views the data and approves the day's work; the management team can also analyze the data and view reports.

This manual process of capturing data has several problems. Data has to be entered by sales representatives on paper and then re-entered at end-of-day into the CRM system via the Web. This process leads to data errors and discrepancies.

The **key drawbacks** of the existing manual process were:

- Inefficient data collection process
- Capturing information on paper is time-consuming
- Delays in getting information from the field
- Consolidation and decision support delayed
- Delay in sending the latest information to the field
- Expenses incurred for stationary, phone, and so forth
- Insufficient data to talk to physicians and chemists
- No history of calls available with sales representative
- Error factor due to late data entry
- No knowledge of current inventory status of an item in the field.

Table 2. Overview of Middleware and Client Application Components in the Solution Architecture for the Territory Management System Case Study

Middleware

Component	Description
CRM Data Access Manager	• Performs updates on CRM database (data captured by sales representative while on the field)
Data Exchange Service	• Composes the changes (out–queue) from CRM back-end system based on the data sub-setting for every user (like doctor/chemist appointment schedule) • Runs in an interval of two minutes after the completion of the previous process • Applies data from the staging database (in-queue) to CRM back-end system using the CRM Data • Access Manager (data captured by sales representative in the field).
User Management	• Contains list of users linked with enterprise directory service • Contains user specific information like device information, last synchronization date time, device lockout status and so on.
Middleware Console	• User interface for the middleware • User interface for user management, device management, viewing synchronization logs and data exchange service logs.
Device Management	• Place new application builds in the pre-defined shared location in the middleware • Create builds to cater to different types of devices. (Note the corresponding component in the client application is taken care by Device Synchronization Manager.)
Sync Service	• Place the incoming records in the in-queue • Push the outgoing records from the out-queue • Manage user wise queue like an index table to check if a particular user has any records for download.
Authentication Service	• Authenticates the user by connecting to the enterprise directory service.

Client Application

Component	Description
Communication Manager	• Connects to the middleware for authentication and data synchronization • Tries to connect with middleware first via Microsoft ActiveSync, if not available then connect using GPRS/CDMA • If no connection is available then return with message saying so.
Device Synchronization Manager	• Composes the changes on the client database in XML format • Applies incoming data to local device database • Picks up client application update file from the pre-defined location in the middleware if the file is not available in the device.
Remote Data Access	• Connects to middleware to get updated appointment and campaign lists.
Device Application (Alerts/UI/ Authentication)	• Displays alerts on campaign details, missed appointments, and so forth • Authenticates with middleware if connectivity is available or authenticates locally if connectivity is not available.

Solution

Since the CRM solution had already been implemented, an extension on to mobile devices was required with the same set of features. The handheld application had to integrate with the CRM back end seamlessly. This handheld base solution enables the sales representative to capture and transfer information from the field efficiently. The mobile application runs on the handheld, which is carried by the sales representative when they are on the field and has information such as customer data, product, sample, call history, appointment schedules, and product inventory. Figure 2 shows the deployed mobile solution. The solution has four major components: handheld application, handheld database, middleware, and CRM application.

Handheld application. This application runs on the Windows Mobile devices and is used to capture the data from the field. The application also has a synchronization component to synchronize the handheld data with the server database at office.

Handheld database. This is the database that resides on the handheld. This database has the data specific to the individual sales representative to enable his work in the field.

Middleware. This component, residing on the enterprise end, is used to synchronize the data between the CRM database and handheld device. The middleware uses a staging database that acts as the server for the handheld device. The staged data is then synchronized with the CRM database using native APIs, providing seamless integration.

CRM application. This is the back-end database, which stores the enterprise information. The data specific to each sales representative is downloaded to the staging database and then to the representative's device. The updated data from handheld database is also uploaded first to the staging database and then updated to the CRM database.

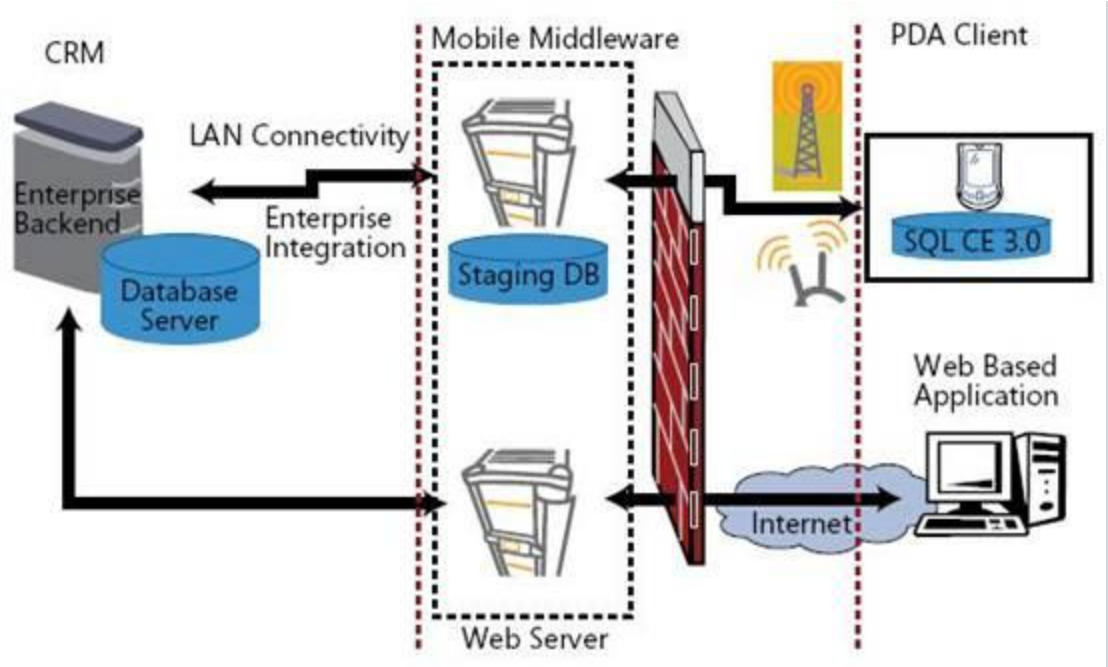

Figure 2. Deployment diagram of territory management system on handheld device

The solution architecture, based on the components from our generic smart client model from Figure 1, is shown in Figure 3. Table 2 lists the components for both the middleware and client application with a brief description of its role as part of extending CRM application on to mobile.

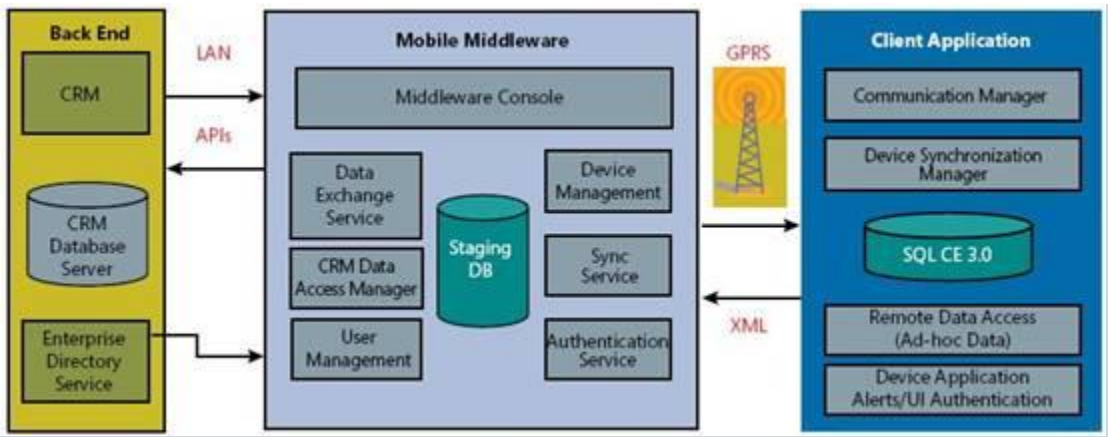

Figure 3. Solution architecture for territory management system on handheld device

The architecture and process flow can be summarized as follows:

- The sales representatives have handheld devices with mobile application installed.
- Sales representative creates a weekly (can be daily or monthly also) schedule for meeting the doctors, approved by supervisor.
- Representative connects to the enterprise network through GPRS and downloads the data specific to the sales representative in the handheld database.
- The representative meets the physicians and captures the sales call information using the mobile application. Representative also captures which (if any) samples or promotional materials were given to the physician.
- Data is uploaded and downloaded automatically without user intervention; latest data is available to the user via background synchronization.
- Sales representative adds/updates the physician information if a new physician has been targeted or information of existing physician has been modified.
- Sales representative books new orders from hospitals or chemists, aided with a real-time view of inventory status.
- The expenses incurred in meeting the physician and chemists are also captured using the mobile application.

• Managers can view the activities of the sales representative through the reports component. They can also create business plan and strategy after analyzing the data.
• Throughout the day, the manager can track the sales representative working pattern and the data entered.

Addressing Key Challenges

During development, the design team came across many challenges.

The challenges and the way they were addressed is described below:

• Cannot make any modification schema to support mobile extension: This challenge was addressed by creating a staging area having a schema structure similar to CRM back-end database.
• Cannot update directly into the tables from the mobile device: This challenge was addressed by creating mapping tables and using wrapper code to call CRM back-end system API's.
• Understanding the schema structure in which data is stored: This challenge was addressed by going through the technical documentation of CRM and going through the table structure in the CRM system database to understand the each field and its use.
• Designing a staging area with the similar schema structure of the back-end CRM system for data to flow to the mobile devices: This challenge was addressed by first copying the structure of CRM backend database on to staging database, then removing the fields that need not be on the device and last creating data exchange service for efficient integration with CRM back-end system.

"Sales representatives of the pharmaceutical company make about 10 field calls a day to meet physicians to promote drugs, take orders, and distribute samples. Information collected is captured on paper by the sales representative who, at end of the day, enters all call-related information on to the Web-based CRM application. This manual process of capturing data leads to data errors and discrepancies."

Pay-Offs

The sales force automation system has been well-received by the pharmaceutical company, especially by the 2000 sales representatives who are the target users of this application.

The implementation of the system has led to an efficient and effective field data collection process and improved communication between the sales representatives and management. Sales representatives in the field and managers in the office each have access to the information they need—data that is up-to-date and relevant to them—whether making field calls or planning marketing strategies.

About the Author

Kulathumani Hariharan is a senior solution architect working with the Wireless & Pervasive Technologies Practice of TATA Consultancy Services. He specializes in architecting enterprise mobile solutions and defining in the mobile middleware adoption strategy for the enterprise mobile customers.