MODULAR APPLICATIONS FOR MOBILE DATA SYSTEM

REFERENCE TO PRIORITY DOCUMENTS

This application claims benefit of priority of: co-pending U.S. Provisional Patent Application Serial No. 60/664,121 entitled "Data Management for Mobile Data System", by Robert O'Farrell et al., filed March 21, 2005; co-pending U.S. Provisional Patent Application Serial No. 60/664,088 entitled "Modular Applications for Mobile Data System", by Robert Loughan, filed March 21, 2005; co-pending U.S. Provisional Patent Application Serial No. 60/664,122 entitled "Adapter Architecture for Mobile Data System", by Robert O'Farrell et al., filed March 21, 2005; and co-pending U.S. Provisional Patent Application Serial No. 60/667,816 entitled "Modular Applications Management for Mobile Data System", by Robert O'Farrell et al., filed April 1, 2005. Priority of the respective filing dates is hereby claimed, and the disclosures of these Provisional Patent Applications are hereby incorporated by reference.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or patent disclosure as it appears in the U.S. Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND

1. Field of the Invention

The present invention relates generally to mobile computing systems and, more particularly, to data management and data deployment in mobile computing systems.

2. Description of the Related Art

Sophisticated customer relationship management (CRM) and enterprise resource planning (ERP) systems are available to improve the automation of back office and front office processes. Although many companies have realized significant savings and efficiencies from deploying such systems, it is also true that many organizations find the systems burdensome to implement and difficult to integrate with existing legacy data systems.

More recently, business organizations and enterprises are deploying CRM and ERP systems to assist mobile employees, primarily to utilize mobile computing devices such as pagers and cell phones and also personal digital assistants (PDAs). One important impediment to greater adoption of CRM and ERP systems that employ such mobile devices involve integration with other data in the enterprise.

Enterprise data integration issues can arise because mobile applications often come in proprietary, closed architectures that impede integration with other data systems of the enterprise. For example, data in the enterprise might be maintained in four or five different sources. Some of the data sources include CRM systems, dispatch systems, ERP systems, and financial records systems. Each of these data sources can utilize a different data architecture, format, and protocol. The data being stored and the configuration of the data and access mechanisms are constantly changing. Many mobile computing systems create an interim datastore in which data from the various sources in the enterprise is collected. In this way, data from the different enterprise data sources, each with a different data architecture and format, can be collected in a single common database. The mobile users can access the enterprise data by accessing the interim datastore, rather than the actual enterprise data sources. The interim store, however, creates data update and conflict issues of its own. Synchronization operations and other safeguards must be performed frequently, to ensure that the data in the interim datastore is a faithful copy of the data in the enterprise data sources.

It is known to provide a data integration solution that can utilize mobile computing devices that interface to enterprise data sources through a network server. Such a system is described in U.S. Patent Application Serial No. 10/746,229 filed December 23, 2003 assigned to Dexterra, Inc. of Bothell, Washington, USA. The contents of this application are incorporated herein by reference.

The Dexterra, Inc. patent application describes a system in which data is utilized between multiple enterprise data sources to mobile clients in a distributed fashion such that requests from a mobile client for enterprise data are received, the appropriate enterprise data sources that contain the requested data are

determined, and the enterprise data is retrieved from the determined enterprise data sources. When the enterprise data is retrieved, it is converted into a relational format, even if the data comes from multiple enterprise data sources of different non-relational types (e.g. File System, email, etc). The converted enterprise data is stored in a relational datastore in the mobile client. In this way, mobile applications can be fully integrated with data from multiple enterprise data sources and data updates and configuration changes can be distributed to and from the mobile clients in real time, without using interim data storage, and thereby avoiding complicated synchronization and asynchronous data issues between the enterprise data sources and the mobile clients. The real time data changes can include deployment of changes to the mobile application itself, as well as data updates. The real time changes are further accommodated with data conflict detection and resolution.

Tools are provided with which developers can define mobile applications that will generate a series of PDA display screen windows such that an application user progresses in an orderly fashion from window to window. In this way, the application prompts the user for data it needs to properly operate, and the application developer is free to fashion a logical sequence of operations that will suit the purposes of at hand. Such design tools are of great assistance to developers, and also to application administrators, who want to modify their system operation as their needs change.

In the Dexterra, Inc. system referred to above, tools are provided for developers to define mobile applications that comprise a series of PDA display screen windows such that an application user progresses in an orderly fashion from window to window. As data is entered and responses are provided in a window, the mobile application will display a succeeding window in the sequence of operations. The sequence of windows comprising the application is referred to as a "force flow". At any single window in the force flow, a user response might result in the display of a dialogue box or data entry window. A user can provide data that is related to the force flow window from which the dialogue box was generated. Multiple dialogue boxes may be displayed, to prompt the application user for the entry of appropriate data and responses, before the sequence of force flow windows is resumed. Such temporary detours from the force flow are said to

comprise a "field flow". After the developer defines a full complement of force flow and field flow displays, they can be linked together to provide the desired mobile application.

The deployment and maintenance of such mobile data integration systems would be made even more convenient if new applications could be developed more quickly and with less effort. The present invention provides these features.

SUMMARY

In accordance with the invention, operating sequences of the mobile data integration system are supported such that such operating sequences comprise operational modules, also called transportable applications ("TransApps"), that are self contained and capable of being linked together with other operating sequences and TransApps. Each operational module can accept input data and can generate output data. The input data can be received from other modules, or from the application user, or from enterprise data sources. The output data can be provided to other modules, or to the application server (for enterprise data sources), or can be provided for display on the mobile computing device itself. In this way, modules can be defined for the purpose of particular problem solutions, and such problem solution modules can be reused during other application design efforts. In this way, earlier solutions to problems can be utilized over and over again, and the knowledge and experience gained from a user community can be exploited for greater leverage, thereby increasing the efficiency of mobile data integration systems.

Other features and advantages of the present invention should be apparent from the following description of the preferred embodiment, which illustrates, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a suitable computer system environment for a mobile enterprise platform constructed in accordance with the present invention.

Figure 2 is a block diagram of the logical architecture of data in the mobile enterprise platform illustrated in Figure 1.

Figure 3 is a block diagram that illustrates the Connector interface between the enterprise data sources and the mobile client of Figure 1.

Figure 4 is a diagrammatic illustration of the force flows and field flows from which a mobile data application in accordance with the present invention can be constructed.

Figure 5 illustrates a screen shot of a graphical user interface display window for the DAD application designer program.

Figure 6 is an example of a collection editor screenshot.

Figure 7 is an exemplary screenshot of a Business Object editor of the DAD application designer program for use with business object collections for TransApps.

Figure 8 shows that the Force Flow Designer interface allows a DAD user to add and edit the operations to be performed by a force flow.

Figure 9 shows a force flow collection editor display that a DAD user may use to view members of force flow collections and select any force flows of interest for viewing and manipulation.

Figure 10 shows a Menu Item Collection Editor display that a DAD user may utilize to add a menu item to a force flow sheet.

Figure 11 shows a Toolbar Button Collection Editor display that a DAD user may utilize to add a toolbar button to a force flow sheet.

Figure 12 shows a Mapping Editor display that a DAD user may utilize to specify data mappings.

Figure 13 is a flow diagram that illustrates operations executed by a computer system in executing a transportable application module in accordance with the present invention.

DETAILED DESCRIPTION

In a mobile data integration system in accordance with the invention, mobile applications can be specified by defining force flow and field flow displays, and also by specifying one or more operational modules, also called transportable applications ("TransApps"). The operational modules are self-contained and are capable of being linked by metadata to other operating modules and to individual force flows and field flows. Details of the operational modules will be provided below in greater detail, after the overall system configuration is described.

I. SYSTEM OVERVIEW

The present invention provides a system in which data is utilized from multiple enterprise data sources to mobile clients executing mobile applications such that the mobile applications are integrated with the multiple enterprise data sources, and data updates and configuration changes can be distributed to and received from the mobile clients in real time, without using interim data storage. The elimination of an interim data storage facility avoids complicated synchronization and asynchronous data issues between the enterprise data sources and the mobile clients. Thus, data updates and system configuration updates for the mobile application can be communicated from the enterprise to the mobile clients, and from the mobile clients to the enterprise, in real time. No special synchronization operation is needed, as changes can be propagated through the system in real time.

II. SYSTEM PLATFORM

Figure 1 is a block diagram of a suitable computer system environment 100 constructed in accordance with the present invention. Figure 1 shows a mobile client device 102, such as a Personal Digital Assistant (PDA) device that operates in conjunction with the Microsoft PocketPC or Palm PDA operating systems. The mobile client device communicates over a network connection 104 with an application server 106 to request data from the server and receive data updates, provide new data, and receive configuration changes. It should be understood that multiple mobile clients 102 can communicate with the server 106. Only a single client device 102 is shown in Figure 1 for the sake of drawing simplicity.

The mobile clients 102 consume the server-side connector web services for real time data retrieval from multiple enterprise data stores. Additionally, the mobile clients consume the server-side data manager web services for the management of real-time client-side data updates, server side data updates and system configuration updates.

The application server 106 communicates with enterprise data sources 108, such as CRM data sources, ERP sources, financial system resources, legacy data stores, and the like. The exemplary enterprise data sources illustrated in Figure 1 include data including "Siebel" software from Siebel Systems, Inc. of San Mateo, California, USA; "Oracle" software from Oracle Corporation of Redwood

Shores, California, USA; "SAP" software from SAP AG of Walldorf, Germany; and legacy software. The administrator application 110 and a developer application 112 communicate with the application server 106, which also stores metadata 114 for the system, as described further below.

The application server 106 provides data manager, configuration, and data connector web services for data interchange and updating, user authentication, security, and logging services. The application server also handles business process management in the form of business information and rules.

The mobile client 102 also includes a datastore 116 that includes a relational data base 118 that stores business data 120 and also a relational database that stores metadata 122 for application execution on the mobile client. An application 124 that is installed at the mobile client 102 includes various software components that perform suitable functions. For example, the application might comprise a field service application that informs field service personnel as to a location at which service has been requested, explains the nature of the service request, and provides for logging the service visit and settling the account. The application 124 may include multiple applications that process the data requested by the mobile client 102.

The administrator application 110 and developer application 112 together comprise a "Studio" component 130. In the illustrated embodiment, the administrator and developer are provided as two separate applications, and provide a means to configure the system, including the metadata data and application interfaces.

The system 100 comprises a mobile enterprise platform that supports the service application 124. The system provides a set of Web services that effectively deploy and manage mobilized software solutions to enhance mobile business processes. Common examples include integrating to CRM or ERP, sales force automation (SFA), and customer support and help desk functions for an enterprise. Such enterprise applications depend on cross-application interaction, in that data from one function or system is often used by a different function or system. When executed on the mobile client, the existing application functionality and enterprise information is utilized among multiple enterprise software applications, legacy data systems, and mobile workers. In this way, a

significant return on investment can be achieved for these applications and for the mobile enterprise platform.

The mobile enterprise platform 100 provides Web services that simplify the use of mobile clients and associated portable devices in the field. These Web services include a data manager function, a configuration function, and a connector function. These will be described in greater detail below. The applications 124 that are installed on the mobile clients 102 can be fully functional in any connected or disconnected state, after they have been properly initiated by the application server 106.

III. LOGICAL ARCHITECTURE

Any client application that makes use of the Mobile Enterprise Platform illustrated in Figure 1 will utilize the system components illustrated in the block diagram of Figure 2. These components include:

Business Objects--programmable objects based on business concepts, combining fields and relating information from different enterprise data sources. (e.g. data sources such as Customer, Contacts, Assets, Tasks, etc.).

Business Rules--custom logic to enforce business processes utilizing business constants with checks applied against business data from the enterprise data sources.

Business Constants--A user-configurable variable for use throughout the client applications, and client and server-side business rules (e.g. Business Rules, Warning Messages, and the like).

Datasource Connectors--data source connectors designed to seamlessly provide access to a wide variety of enterprise data sources (e.g. databases such as those formatted according to Oracle and SQL Server, messaging systems such as MQ Series or MSMQ, CRM applications such as Siebel or Peoplesoft, generic web services, and so forth).

Business Process--metaphors, such as a "Force Flow" process of Dexterra, Inc. of Bothell, Washington, U.S.A., that defines a form-to-form navigation paradigm for modeling business processes.

Forms--a combination of standard visual display screens (e.g., View, Edit, Find, and the like) with event driven logic that are designed to show information, gather information, and direct the user through a given business process, referred to herein as either a "ForceFlow" or a "FieldFlow".

Views--A modifiable representation of the data identified from an enterprise datasource or application that is utilized by one or more Business Objects.

Filters--A Filter that can be applied to a View to modify the data available to a Business Object.

These components can be used to specify the configuration (logical architecture) of any client application that is constructed utilizing a technology framework such as the Microsoft Corporation ".NET" and tools such as Microsoft Corporation's "Visual Studio .NET". Those skilled in the art will be familiar with such programming tools to specify an application and its associated data objects.

The Mobile Enterprise Platform illustrated in Figure 1 is implemented as a metadata driven framework. The framework provides integrated client and server web services, enabling the connection, configuration, and data management services necessary to deploy fail-safe, mission-critical mobile enterprise solutions.

Figure 2 illustrates that, in the mobile enterprise platform of Figure 1, the structure of relational database tables and external application business objects are mapped to views as metadata. One or more views are consumed by Business Objects, also defined in metadata, which are in turn utilized by the mobile application. The mobile application utilizes a client framework, referred to as the "Dexterra Smartclient", which manages the instantiation of the Business Objects, Local Data Access to the underlying physical database that resides on the mobile client device, Device integration, as well as the client-server data communication via the data manager and/or connector web services. Within the platform, specifications for all logical layers (e.g., Business Objects, Views, Filters, and Connectors) are defined and maintained within the metadata.

The mobile enterprise platform is architected as a logical stack, designed to insulate layers in the logical architecture from all but non-adjacent members. At the bottom of the logical stack, the Target layer, is data that resides in back-end,

enterprise data sources. The platform works with the source data in place, and does not require information within the back-end system of record to be replicated to a middle-tier replication database. That is, no interim datastore is needed. This provides flexibility in design, as well as real time data access and can help reduce total cost of ownership of the platform and applications, and assists simplification of data management processes.

The next layer up in the logical stack is the Connector layer. The Connector layer provides a programmatic construct that describes the back-end datastore to the application server in a relational format. The information regarding how to connect to an enterprise data source, as well as the security settings (such as authentication methods and user and group definitions) are stored within metadata, and are maintained using the Administrator component.

The next layer in the stack is the View layer, which comprises objects that provide a one-to-one mapping to an object or table in a back-end, enterprise data source. For example, if a back-end system has a table called CUST_ADDR (customer address), and data from that table is required for use in an application, then a View will be created in the Administrator component. The Administrator View might be called, for example, CUSTOMER_ADDRESS, to represent that data in the environment of the mobile enterprise platform, outside of the enterprise data sources. It should be understood that a View has properties that correspond to the properties or columns of the data object in the back-end system. However, it is not required that all properties in the back end data source are required as properties in the View. Indeed, the properties required are defined in the administrative component and stored as metadata In the example just provided, the properties might include fields such as ID, STREET_ADDR, CITY, STATE, and ZIP_CODE.

Additionally, the user can define the data types of the properties within the View, and these data types can be independent of the data types of the corresponding properties in the enterprise data source. Other options of the view properties that can be identified are unique identifier, read only, indexing, required property and length. All the above information is stored as metadata.

The View layer also provides an indication of data conflicts, and provides a means for resolving such conflicts. Data conflicts can occur, for example,

whenever there are data changes between what is being uploaded from the mobile client and what exists at the server. Resolution of such conflicts can be performed at the View layer, enforcing business rules such as permitting the most recent data change to always take precedence, or permitting data changes from a particular source (e.g., either the mobile client or an enterprise data source) to take precedence depending on the data type (e.g. field data or customer account data). This is described further below, in conjunction with the Data Manager Web Service.

As illustrated in Figure 2, the Views can be defined against multiple objects in multiple datastores, thus providing flexibility in application deployment and in the use of in-place systems, without the burden of data replication. As with the Connectors, the definitions of Views are stored in metadata, and are managed with the Administrator. Those skilled in the art will understand details of data definitions in metadata, without further explanation. As noted above, Filters can be applied to the Views, to modify the data that is passed to the next layer. The Administrator provides View management features, including a Views Wizard that automatically creates Views based upon the object interface or table definition of the back-end datastore objects (from the enterprise data sources).

The next layer up in the Figure 2 diagram includes the Business Objects, which are mapped, or associated with, one or more Views. A Business Object of the platform is the programmatic entity with which a developer will interface with when building customizing mobile applications. The Business Objects include multiple properties, each of which can be of a simple data type, or can be another Business Object. Because the Business Objects of the platform can be mapped to multiple Views, developers can work with a single entity that represents data sourced from multiple, heterogeneous data sources. Thus, a single Business Object defined in accordance with the mobile enterprise platform of the invention can include data from multiple, potentially incompatible enterprise data sources, such as from different proprietary formats.

In creating or modifying applications for the mobile applications and mobile client devices, developers can interact solely with the Business Object layer. This insulates the developers from any requirement to understand or interact directly with the back-end systems (enterprise data sources) for the source data. In this

way, the Business Object layer provides an object-based interface for application developers, abstracting the details of persistence and retrieval of data. There is no need for the developer to directly interact with the local datastore on the mobile device. In addition, due to the nature of disconnected data, the mobile client, through the Business Object interface, automatically manages the processing of data changes, by storing data changes locally in the client that will be passed to the application server during an Update process. This further insulates developers from this rote programming task.

The Business Objects exist on the mobile client device as metadata, and are also managed using the Administrator (Figure 1). The use of metadata throughout the mobile enterprise platform provides an environment in which the attributes and behavior of most data entities can be configured through a graphical user interface rather than coded.

The metadata-driven nature of the mobile enterprise platform enables performing business processes on the mobile client through a stateless server architecture. Through the metadata, the mobile application can be configured and customized. The metadata defines the structure of the business objects referencing the business enterprise data to the mobile device and defines the events that trigger business rules that govern the business processes.

The metadata database contains the reference of the cross-functional, cross-application back-end business information that is exposed through the Connectors to configure a business object. This process is accomplished through the Studio component (Figure 1) to configure and reference the connecting enterprise data source business information with the Business Objects. This provides the path to the specific data for the mobile applications, ensuring that no business data from an enterprise data source is stored in its native data format on the application server or on any other interim datastore of the system for data updates. This non-invasive and real time synchronous approach using the metadata permits the mobile enterprise platform to effectively connect to back-end systems with a minimum amount of disruption while maximizing cross-functional data access, data consistency, and data integrity.

IV. MOBILE ENTERPRISE PLATFORM COMPONENTS

A. MOBILE APPLICATIONS

As noted above, the mobile client 102 (Figure 1) can include installed applications 124 that implement business processes of the enterprise. The application can leverage the mobile enterprise platform described above, and demonstrates how the application instantiates the business objects which drive the business process configured in metadata.

For example, Task or Work Order information would be provided to the mobile application through views and would be accessed via a business object. In retrieval of the business data via the view definition, using the data manager web service, the business object can deliver the business data to the mobile application to describe the tasks. This data is stored on a local relational database on the mobile device. When an update to the task data is committed to the task business object in a request from the application, the Smartclient application will persist the changes to the view defined datastore on the mobile client, then the Smartclient manages the data updates back to the original data source via the data manager web service, ensuring data integrity and consistency.

By utilizing the depth, breadth, and power of web services (e.g., connection, configuration, and data manager services) that are available in the mobile enterprise platform described herein, a large suite of mobile applications can easily be constructed, including applications such as sales force productivity, customer service, and support solutions. Such applications can be integrated with a broad set of vertical applications including oil/gas, healthcare/medical and financial service industry solutions.

B. SERVER COMPONENTS

The application server is a type of metadata-driven platform application and provides information, applications, and business processes to the mobile client, and ensures managed data integrity between the mobile enterprise platform and a host of back-end enterprise data sources. The application server is a process-based, high performance solution built on the ".NET" technology from Microsoft Corporation of Redmond, Washington, U.S.A. Using the ".NET" technology, the mobile enterprise solution is a framework that is Web Services native through the use of XML and SOAP for data exchange and transport. The application server provides three core Web Services, as shown in the functional architecture diagram of Figure 1:

Connector Web Service

The Connector Web Service delivers non-invasive integration of the existing enterprise applications infrastructure while maintaining control of the Data-Integrity Conditions between the mobile clients and the discrete enterprise data sources.

Configuration Web Service

The Configuration Web Service manages the metadata defining the business data, business objects, business rules, business constants, and system configuration such as authentication, logging, security, and roles that encompass the mobile applications that are passed to the mobile client--the component application that is resident on the mobile device.

Data Manager Web Service

The Data Manager Web Service orchestrates the update interactions between the mobile client application, the application server, and the third-party enterprise data sources. Additionally the Data Manager Web Service provides the ability to directly communicate with the connector layer for real-time queries. The Data Manager Web Service delivers flexibility in the manner that manages the various conditions concerning multiple updates by multiple users to the multiple enterprise data sources to enforce the integrity of the data. The Data Manager Web Service can do this via the application server or direct to any API and/or third-party published Web Service.

In this way, the Data Manager Web Service can manage deployment of application updates and data changes throughout the mobile clients of the system.

Each of these components will next be described in greater detail.

1. Connector Web Service

The Connector Web Service is designed to support communication with any ODBC-compliant data source or Web Service API. The Connector Web Service allows a customer to define and build views based on data stored in one or more third-party systems. The Connector Web Service has a published interface that allows for standard bulk updates as well as real-time data access from a mobile client.

The Connector Web Service provides the physical layer connection between the application server meta-application and the specific interface of the enterprise data sources. The connectors support database dispute management and notification services, transaction management, and error handling. In a default customer configuration, the mobile enterprise platform system is deployed to customers with an ODBC or Web Service connector. Those skilled in the art will be able to produce connectors to the most common enterprise systems, such as Siebel, SAP, PeopleSoft, Oracle, SQL Server, and the like.

For example, an "Oracle" applications connector allows a customer to make calls to Oracle support services, either through the closest data constructs the customer has to APIs (such as PL/SQL procedures) or directly to the enterprise database itself via ODBC. As with all of the ODBC connectors the dynamically interrogation of the RDBMS schema is automatically executed, exposing the specific physical design of the database. This gives the customer a hierarchical view of the actual interfaces into that system.

Figure 3 shows an example of how the Connectors interface the enterprise data sources to the mobile enterprise platform. On the left side of Figure 3 are representations of multiple enterprise data sources, including an ERP data source 302, a CRM data source 304, an HR/Finance data source 306, a Legacy/ODBC data source 308, and can include other Web Services or other sources (not shown). In the middle portion of Figure 3 is a representation of the metadata 312 that specifies to the application server 314 how data from the different enterprise

data sources will be stored and related in the mobile client 316, which is represented at the right side of Figure 3.

Thus, in this example, data identified as ORDER_ID exists in the ERP data source. Data identified as F_NAME and L_NAME exists in the CRM data source. Data identified as CRED_LIM exists on the HR/Finance data source, and data identified as WARRANTY is stored in the Legacy/ODBC data source. All of these identified data are stored in enterprise data sources, such as at back-end office systems.

In the metadata 312, the data definition from the enterprise data sources is mapped to views that are used to create the data store on the client and store the relevant business data on the mobile client from the enterprise data sources in a relational database . Access to this business data is performed via a the business object layer defined and stored in metadata on the mobile client. As shown in Figure 3, the ORDER_ID from the ERP data source is mapped to a business object property called OrderID, whos relational definition is stored in metadata 318 on the mobile client 316 and utilized by one or more the mobile applications also defined in metadata. The F_NAME data from the CRM enterprise data source is mapped to (stored into) the FirstName business object property definition stored in the mobile client database, and the L_NAME data is mapped to the LastName business object property. Similarly, the CRED_LIM data from the HR/Finance data source is mapped to the CreditLimit business object property, and the WARRANTY data from the Legacy/ODBC data source is mapped to the Warranty business object propery. Thus, data from the potentially dissimilar and incompatible disparate enterprise data sources 302, 304, 306, 308, 310 are delivered to the mobile client through the Data Manager Web Services to the local data store (represented by the lines from the enterprise data sources to the application server 314) in the proper format for access using one of the business objects on the mobile client (indicated in the mobile client 316 with actual values).

Connector Types

The connectors that are supported by the Connector Web Service include the following three connector types:

1. The Web Services connector is used when the mobile platform is connecting to a third-party system (a) that is either non ODBC-compliant, or

(b) does not allow ODBC/RDBMS connectivity, or (c) whose interface is defined by a standard API and can be wrapped and defined by Web Service Descriptor Language (WSDL).

2. The ODBC/RDBMS connector is used when connecting the mobile platform to a third-party system (a) that is ODBC compliant and (b) allows for direct ODBC/RDBMS access and (c) whose data is located physically within the same LAN environment or accessible via a communication protocol supportive of the transport (such as RPC, TCP, etc.).

3. The API connector is similar to the Web Services Connector but (a) requires the API to be accessible via non internet protocols such as RPC and (b) is used if the Web Services Interface is not available.

Reading schema, via the ODBC/RDBMS connector, information is accomplished through the use of the Studio portion 130 (Figure 1) of the mobile enterprise platform, using the Administrator application. The Studio portion is used to configure the View definition mapping to the backend data source and map the definition of one or more Views to one or more Business Objects. When defining the View definition or mapping the Views to Business Objects, using the administrator, the information is stored as metadata. During an update process with the application server and enterprise data source, the metadata is read to determine how to read, persist and remove the data (select/insert/update/delete functions) while managing and enforcing the data integrity using such functions as conflict detection/resolution, transactions both inherent and compensating where appropriate.

Using the ODBC/RDBMS connector, data is read, persisted and/or removed via ANSI SQL statements and/or stored procedures in the case of Microsoft Corporations SQL Server or Oracle's RDBMS (8i, 9i, etc.). Using the Web Services/API connector, data is read, persisted and/or removed by calling the appropriate API function or method for the transaction.

2. Configuration Web Service

The Configuration Web Service consumed by the Dexterra Studio provides an easy interoperable way for administrators, business analysts and developers to implement, configure, and administer the Dexterra Mobile Enterprise solution.

The Configuration Web Service allows for easy manipulation of the metadata used to configure and customize the data and process definitions of Mobile applications. This service will be better understood with reference to the features of the Administrator component, which is described in greater detail below.

3. Data Manager Web Service

Update Process Model

An update process model is utilized in the system, in which mobile applications update their locally held data (either the application or its business objects) with the backend enterprise database using a set of core Net components that are exposed as Web Services for easy interoperability.

The Data Manager Web Service updates the mobile application and all its associated business objects defined data. The Update process model enables two-way data transfer between the enterprise datasources via the Dexterra application server and the mobile client, allowing updates to be made while the mobile client is connected to the network, merging the updates between clients when they are connected. When in the disconnected state, updates are managed in the client environment, until a time at which a connected state is attained and the update request can be initiated.

The update process model takes the "all or nothing" approach. If a failure occurs before the entire stream is downloaded from the application server onto the mobile client (or before the entire stream is uploaded from the client to the server), then the Data Manager Web Service on the application server does not receive a confirmation on the download transaction (or upload). As a result, the server carries the intelligence to manage the client state as to whether it requires a roll back of data or simply a retry. When the mobile client performs an update process operation the second time, the application server takes into account the original information state and may either deliver the results if the application server has processed or process again in the event all the required information was never received by the application server thus enforcing the reliable deliver of information once and only once between the mobile client and application server. This, in event, enforces the integrity of the data as it moves from mobile client to one or more back end data sources.

Update Process Breakdown

Two types of update processing are supported:

1: Get Latest: In this update type, the mobile client makes a request to get the latest information from the enterprise data sources via the Dexterra application server.. The Dexterra application server process the request and retrieves the business information from the multiple data sources using the Dexterra Connector Web Service and delivers the business information to the mobile client.

2: Update (2-way update): In this update type, records on both the client and server end are interchanged enforcing the integrity of the data on both the mobile client and the back end enterprise data sources using Dexterra Conflict Resolution configured parameters.

Conflict Detection/Resolution

Conflict resolution describes the rules used to arbitrate on data conflicts caused by changes made between a mobile client and one or more back end enterprise data sources. This is performed first by identifying the conflict (Detecting) and then resolving (Resolution) the conflict in one or more various ways.

The Dexterra application server can detect conflicts in one of three ways: Revision, Date/Time Stamp or Manual as well as identify a conflict situation by row or column level.

Revision is a setting where a specific field or property is identified in a single record source as revisioned and the Dexterra application Server will use this to determine whether data has been changed on either the back end data source or the mobile client.

Date/Time Stamp

Date/Time Stamp is a setting where a specific field or property is identified in a single record source as date/time stamp and updated upon any insert/update or delete and the Dexterra application Server will use this to determine whether data has been changed on either the back end data source or the mobile client.

Manual is a setting where there is no specific field or property to identify a conflict situation in a single record source therefore the Dexterra application Server compares all the field or property data to define uniqueness and detect

whether data has been changed on either the back end data source or the mobile client.

Depending on configuration of the Dexterra application Server, Conflicts are resolved in one of four ways: First Update Wins, Last Update Wins, Admin Resolution or Server-side Rule

First Update Wins

Under the First Update model the application server will only accept changes of any record that is the first one to make an update. If a record is first updated by the back end data source and a conflict is detected by the Update Web Service, instead of returning an error, the Data Manager Web Service will drop the version provided by the client and return a copy of the latest version of the record from the back end enterprise data source to the mobile client.

Last Update Wins

Under the Last Update Wins model, the server need not detect conflicts. Instead, it simply persists the changes from the mobile client to the back end enterprise data source overwriting the current record in the back end enterprise data source.

Admin (or Manual) Resolution

When configured for Admin/Manual resolution, the server will treat all conflicts as requiring manual intervention to resolve and will return a copy of the current record from the back end enterprise data source and optionally notify via any notification service (SMS, Email, etc.) that a conflict situation has arisen and allow for resolution via the Dexterra Administrator. Doing so allows for column level conflict resolution since the Administrator determines the values to reapply back to the back end enterprise data source selectively.

Server Side Rules

Customizable Server Side Rules can be created to determine more programmatically and specifically how certain conflict situations should be resolved. For example, a conflict may be resolved based on the values of data in a record. This flexibility allows for complete control over the specific actions surrounding a conflict resolution scenario.

Client Deployment from the Server

The application server contains the definition of one or more mobile field applications that are to be downloaded to the mobile client, including the Forms/screens represented as tasks (referred to as "FormFlows"), data-interactions (referred to as a "FieldFlow"), and groups of FormFlows and FieldFlows constructed into a Business Process/Workflow (called a "ForceFlow"). The FormFlows, FieldFlows, and ForceFlows are described further below. The application definition also includes the configured metadata associated to an application such as View, Business Object, Business Constants definition. Also included in the deployment is the specific business data from one or more back end enterprise data sources required to run the mobile client in an "occasionally" connected state.

The application server provides the foundation on which to deliver and manage applications and to connect to existing enterprise data sources and systems. The mobile enterprise platform applications are distributed and managed to the mobile devices, such as Pocket PC and Tablet PC devices, by the application server, providing a highly manageable administration of all user interfaces in the field.

C. ADMINISTRATOR COMPONENT

As noted above, the Administrator component (Figure 1) allows system administrators to perform changes that are relatively regular or frequent. The Administrator component provides access to decision variables, drop-down list content, and other information in a format appropriate for business analysts or administrators to manage. This approach to administration allows system administrators to extend many functions down to the Administrator level without compromising system integrity.

For example, data comprising business information that is used to define the business processes of the enterprise can be received through a Business Objects definition form. The Configuration Web Service provides access to this aspect of the Administrator component.

D. CLIENT COMPONENT

As noted above, the client 102 (Figure 1) in the enterprise platform architecture provides a framework in which the mobile application allows the use

of role-based business processes using techniques referred to as "ForceFlow", "FieldFlow", and "FormFlow", and using Web Services, thus enabling communications between the mobile client and the Dexterra application Server and the enterprise data sources over a LAN/WAN network, such as the Internet, via wired and wireless connections. The mobile application running on the client devices functions in a manner that is optimized for small form-factor devices providing an exception, easy to learn user experience.

In the illustrated system, the client is an object framework that is built utilizing the ".NET Compact Framework" of Microsoft Corporation that is metadata aware. The client component enables delivery of enterprise-class application functionality on the mobile devices, which preferably operate according to the "PocketPC" operating system or Microsoft Tablet PC operation system from Microsoft Corporation. The client component also integrates with existing "PocketPC" functionality to provide seamless integration with Calendar, Task, and Today screen functionality of the PocketPC interface. It thereby provides a stable, effective environment in which to work.

FormFlows, FieldFlows, ForceFlows

Any business process tasks or steps or operations in the form of display screens are called "FormFlows". The FormFlows are used to initiate process interactions called "FieldFlows" that allow the initiation of business processes, which are referred to as "ForceFlows". The FieldFlows allow launching of "out of band" ForceFlows to bring real-world elasticity to the business processes.

The FormFlows are broken into three categories: (1) Information; (2) Activity; and (3) Update. An Information FormFlow is a screen that shows information needed by a mobile user to fulfill the next logical task in the business process. An Activity FormFlow is a screen that shows something the user may need to do or perform. An Update FormFlow is a screen that is displayed when a mobile user is prompted to enter data that will be returned to the host applications (the enterprise data sources).

A FieldFlow may be required, for example, when a part might have failed and a search of inventory databases might need to be performed to see if any matching parts or similar problems with solutions exist and are available, called a

lookup, or a FieldFlow may be required when a part might need to be ordered or assigned or scheduled for delivery to the client, a FieldFlow called an update.

A ForceFlow is a business process, and therefore is a collection of FormFlows and FieldFlows. An example of a ForceFlow would be time, travel, and expense recording that is associated with a job or dispatch event.

Referring back to Figure 2, this block diagram shows how the relationships between columns and fields in the target application are related to information In the "FormFlows" (steps in the business process represented as 'Forms" in the application) and are then associated into the ForceFlow (the business process). There can be many Business Objects in one FormFlow and potentially more than one FormFlow in any business process.

Filters allow characteristics and conditions to be placed onto the data when referenced in the mobile application. For example, data type (e.g., Date), valid types (e.g., only Monday through Friday), and any conflict conditions may be detected. Other filter characteristics and conditions can be configured.

Views define the data and storage location for use in one or more Business Objects, and the Business Object can be based on one or more Views. This allows additional characteristics to be associated. For example, a Business Object may be referred to as "Customer", which may Include standard customer details; location, contacts, inventory, and also SLA and other attributes that the application would like to classify as Customer but not held in the same Target table or even Target application.

Figure 4 is a diagrammatic illustration of the force flows and field flows from which a mobile data application in accordance with the present invention can be constructed. Figure 4 is a flow diagram that shows an operational sequence 402 that includes a force flow sequence 404 extending from left to right across the drawing. A field flow 406 extends down the page and is adjacent to a second field flow 408 that extends downwardly. The force flow 404 includes multiple display windows, beginning with an initial, left-most display window 410 followed by a second display window 412. The arrows in the drawing indicate the sequence of operations, represented by successive display windows, that will be carried out by the application. The second display window 412 includes two field flow display windows 414, 416 that extend down the drawing sheet. It should be understood

that the operation of the application returns processing to the second force flow window 412 after the processing for the field flow 406 is completed. After the second force flow display 412, the third display window 420 also includes a field flow display window 422 that extends down the drawing sheet, representing the second field flow 408. After operations for that field flow 408 are completed, the application proceeds to the next force flow display 424 and then the next 426.

With previous developer tools, such as described in the Dexterra, Inc. patent application referenced above, it was possible to define the application 402 by specifying the sequence of display windows making up the force flow 404 and the field flows 406, 408. Each of the display windows would be defined by the collection of dialogue boxes, text, and graphics. A suitable development suite that can implement such specifications is the ".NET" Visual Basic suite available from Microsoft Corporation of Redmond, Washington, USA. Each display window 410-426 defined in this way could be linked so that the sequence of operations as illustrated in Figure 4 would be carried out.

E. TRANSPORTABLE APPLICATIONS

In accordance with the present invention, a collection of multiple application windows can be specified as a module, or transportable application (also referred to as a "TransApp" of the system). For example, a collection of operations such as might be specified by the Figure 4 force flow display windows 412, 420, 424 and their associated field flow display windows 414, 416, 422 could comprise a TransApp that performs a time tracking function to provide a timesheet function that enables field workers to enter hours worked and associated data. The collection of operations is performed in the same manner, so that the collection of operations as exemplified by the force flows 412, 420, 424 and the field flows 414, 416, 422 as specified using the TransApp technique described herein will provide the same output on the mobile platform as would the collection of operations as specified without using the TransApp technique. In this regard, the TransApp technique described herein provides a modular alternative to the "brute force" technique of specifying each force flow display page, specifying each associated field flow display page, specifying the data links between them, and storing the collection of operations in an application program for the mobile platform described herein. Thus, the development tools provided in conjunction with the

TransApp technique permit more convenient and efficient development of mobile applications.

The development tools provided in accordance with the present invention enable the application developer to utilize a previously written module by specifying that module from a library of modules when defining the application. Thus, using previous tools, a developer might define the display window dialogue boxes and associated text for the first force flow window 410, and then define the dialogue boxes and text for the second window 412, followed by defining the display windows 414, 416 of the first field flow 406, then defining the next force flow window 420, and so forth. In accordance with the present invention, however, the developer need only specify the first window 410 and then can specify the time tracking functionality of the display windows 412, 420, 424 and their associated field flow display windows 414, 416, 422 by specifying the particular TransApp from a library of TransApps. The development tools also permit a user to create new TransApp collections of operations, and perform typical edit functions with TransApps, such as edit existing TransApps, copy, delete, rename, and the like. In this way, developers can capitalize on previously defined modular operations for reuse. This makes application development for the Figure 1 system much more efficient, and increases flexibility.

V. DEVELOPMENT TOOL COMPONENTS

The development tools with which the operational modules can be specified can include multiple functional components. The functional components will be utilized in accordance with a development application for the mobile platform, with which the TransApps will be constructed and configured. As noted above, the TransApps can be included within a system application for deployment over the mobile platform described herein, so as to provide the desired functionality in which mobile clients will communicate with servers to gain access to data from multiple enterprise data sources. In this discussion, the development application will be referred to as the Data Adaptation Designer (DAD). The DAD comprises a development application program that can be installed on any computer with suitable resources to support operation of the program. For example, the DAD application program can be installed on a computer such as the Figure 1 computer 110 that hosts the Dexterra Administrator program or the

computer 112 that hosts the Dexterra Developer program. The DAD will produce a mobile application program, which can then be installed on the server 314 (Figure 3) of the mobile platform described herein. Alternatively, the server computer can host the DAD application program that produces the application program. Preferably, the DAD will support utilization of functional components that will include:

Business Rule Designer

Business Object Designer

ForceFlow Designer

Field Flow Designer

Component Designer

These components will enable a designer of applications for the mobile platform to efficiently design and specify operational modules, specify previously defined operational modules for inclusion, and thereby efficiently design and implement mobile applications. These tools will be described in further detail below.

A. DATA ADAPTATION DESIGNER (DAD)

The DAD provides the ability to connect to, and construct data components for, any enterprise data source adapter. In the illustrated embodiment, the DAD and adapters utilize a ".NET" plug-in in accordance with the mobile platform described herein. Those skilled in the art will understand the workings of the ".NET" configuration supplied by Microsoft Corporation of Redmond, Washington, USA. The components of the mobile platform include a Connection Object, Command Object, Data Object, and View.

To create or modify a TransApp for the mobile platform, a user will utilize the DAD to create a Connection Object to gain access to a back end enterprise datasource using a datasource Adapter. As noted previously, the enterprise datasource Adapters are configured to interface with disparate data configurations of multiple data sources. The Connection Object will expose any available data interface objects that are available, either using a discovery or intraspection process, or a predetermined description. The data interface objects will be exposed through the Adapter as either a Table, a Stored Procedure, a Script, or an Object. Using the DAD, a user will then create a series of Command Objects that perform specific actions through an Adapter, performing actions such as

Select, Insert, Update, and/or Delete. A DAD user can then define a Data Object in which they will select the appropriate Select Command, Insert Command, Update Command, and/or Delete Command. A View is then bound to the Data Object for its request and respond actions.

In this way, the DAD tool enables a developer to request and persist data from one or more back end enterprise datasources mapped to a single defined data object within the Server, thus providing a layer of abstraction to the physical data structure and interface capabilities. Once the server application has been thus defined and implemented, mobile clients can interface to the enterprise datasources via the mobile platform server, which uses the definition of the Connection, Command, Data, and View platform components to determine how and what data to retrieve or persist to a back end enterprise datasource.

Figure 5 illustrates a screen shot of a graphical user interface display window for the DAD application designer program referred to above. The Figure 5 display permits the DAD user to design a mobile application for use in conjunction with the mobile platform described herein. Figure 5 shows that the display window includes a menu bar for File and View menus to select TransApps and operations to be executed on them through the DAD program. The operations to be executed can include create, open, save, edit, delete, copy, link, and the like. The Figure 5 display window includes a work area (on the left side of the screenshot) in which a user can graphically lay out a TransApp sequence of operations, to specify interconnected force flows laterally across the display page and specify interconnected field flows vertically up and down the page. The display pages of the various force flows and field flows are represented as sheets in Figure 5. Through the interface of the DAD illustrated in Figure 5, the user can create and configure a mobile application, and the user can add TransApps, ForceFlows, Sheets, Actions, Business Objects, Menu Items, and ToolBar Buttons for the mobile application display pages at the client, as well as designate a Splash screen and designate a StartUpObject for the mobile application. As the links or flows between sheets are specified, the system can generate the appropriate code or scripts for ensuring successful operation of the links and flows once the TransApp is executed on the system.

The File menu displays a list of menu items for typical file editing operations to be applied to TransApps, including saving and editing a TransApp, creating, importing, and printing a TransApp. The View menu displays a list of view options for the display window, such as showing a layout view (such as the default layout view illustrated in Figure 5), zoom display control, display of a single item of the TransApp, a hierarchical view that provides a tree structure representation of the TransApp, and the like.

The right side of Figure 5 shows a list of components that a developer designer can select to be included on a window display of the mobile application, to be generated for display at the client device. The components can appear in the mobile application as display buttons for selection of actions or as data windows for entry of data by the user of the mobile application. The components can be stored and retrieved as collections of components, as indicated in Figure 5. That is, the designer will be able to import and edit an existing TransApp or FieldFlow as well as create a new TransApp or FieldFlow by name for use within the mobile application. In either case, the user will be able to add, edit, and remove ForceFlows, Sheets, Actions, Business Objects, Menu Items and ToolBar Buttons using the graphical interface of the DAD, as represented in Figure 5.

The "actions" component allows a designer to define actions that apply at the TransApp level, the "business objects" component allows a designer to define business objects that apply at the TransApp level, the datasource component allows a designer to display a select a datasource and identify a server and adapter to be used, and the force flow component allows a designer to add, edit, and remove force flows from the TransApp. The actions applied at the TransApp level (such as in the Figure 5 window) will be applied to the forms and force flow forms within the TransApp. These actions can over-ride, append, or prepend the actions found on the base sheets of the force flows. Any toolbar buttons and menu items added at the TransApp level will apply corresponding buttons and menu items to all forms (including force flows) within the TransApp. These buttons and menu items are in addition to any buttons and menu items specified at the individual form and force flow form level.

Figure 5 indicates that collections of actions, business objects, force flows, and the like can be utilized in configuring TransApps. To support such

manipulations, the DAD application program also includes a collection editor function that can be launched when a collection is selected from the Figure 5 display.

Figure 6 is an example of a collection editor screenshot. In Figure 6, the editor is adapted for manipulation of TransApp collections. The left side of the Figure 6 display shows names of TransApps that are open by the editor and are available for manipulation and configuration. The right side of the Figure 6 display shows actions (and collections of actions) that can be specified for the TransApp by clicking on the display menu items, and also shows a "Launch Designer" option, specification of TransApps by name, and shows a form option that displays a list of base TransApps that are available for manipulation and configuration, and permits naming of a new TransApp to be created.

The collection editor will propagate changes in the editor window to corresponding sheets and force flows of the TransApp. That is, any actions applied at the TransApp level (such as in the Figure 5 window) will be applied to the sheets and forms and force flow forms within the base TransApp (that is, the TransApp on which the editing functions are being performed). These actions can over-ride, append, or prepend the actions found on the base sheets of the force flows.

B. FUNCTIONAL COMPONENTS

As noted above, the Data Adaptation Designer will incorporate support for five different component tools with which the TransApps can be configured and manipulated. These components include Business Rule Designer, Business Object Designer, ForceFlow Designer, Field Flow Designer, and Component Designer.

1. Business Rule Designer

The Business Rule Designer component provides the ability to create simple business rules in an "IF...Then...Else..." format. Such constructs are to be executed by the SmartClient or Server attached to one or more Events, as described above. The Business Rule can be defined either in a simple interface that can be provided by those skilled in the art or by a launch out to "VS.NET" available from Microsoft Corporation and can be coded in a supported Managed Syntax (e.g. VB.NET, C#, etc.), as known to those skilled in the art.

The business rules can be applied to any of the sheets in the TransApp in response to data conditions or data entered at the mobile platform (client). For example, a mobile platform designer can use the Business Rule Designer of the DAD to specify a rule such that, if a data condition or client user data entry is beyond a predetermined threshold amount or is not within a prescribed condition, then the mobile application program can initiate a display that prompts the user to take correction action or supply further information or the like. In an expense account tracking TransApp for a mobile application, for example, a business rule might check to identify if a client user enters an expenditure amount that is greater than a predetermined limit, in which case the application will initiate a predetermined action comprising asking the client user for an authorization code to accept the amount.

2. Business Object Designer

The Business Object Designer component provides the ability to create and define a Business Object in meta data bound to one or more Views from one or more backend data sources that can be used by one or more Mobile Client Applications utilizing the Dexterra Studio "VS.NET" plug-in. The ability to create relationships to one or more other Business Objects provides for a true Object Oriented application component architecture utilizing the Dexterra Studio VS.NET plug-in.

The Business Object Designer functionality permits a developer to configure the definition of a Business Object including Properties, Default Values, Relationships, Filter Conditions, Permissions, Associated Applications and Business Rules. Using the DAD Business Object Designer, a developer can configure the definition of a Business Object including Properties, Default Values, Relationships, Filter Conditions, Permissions, Associated Applications, and Business Rules.

Figure 7 is an exemplary screenshot of a Business Object editor for use with business object collections for TransApps. The Business Object editor enables a DAD user to add and remove business objects from a mobile application, a TransApp, a force flow, or a sheet, as well as change properties of particular business objects that are open with the editor.

The Figure 7 screenshot shows the program display window of the DAD tool with a Properties sub-window for a particular named collection of data called BusinessObjectSet1, as indicated by the circled area in Figure 7. The sub-window indicates particular editing actions that may be performed with respect to one or more of the Business Objects in the named business object collection.

3. Force Flow Designer

The Force Flow Designer component provides the ability to create one or more mobile client business processes using a designer tool surface for easy construction and generation, again utilizing the Dexterra Studio "VS.NET" plug-in.

The Force Flow Designer functionality permits a developer to lay out (i.e., design surfaces) of a mobile business process and configure the binding to the meta data definition configured using the Business Object Designer.

Figure 8 shows that the Force Flow Designer interface allows a DAD user to add and edit the operations to be performed by a force flow. The DAD user can add actions, business objects, menu items, and toolbar buttons to the sheets of the force flow. As with the other editing windows of the DAD, any actions applied at this (force flow) level will be applied to the corresponding TransApp (force flow) form itself or within the operations of the force flow itself, and the actions can over-ride, append, or prepend the actions in the base forms. In addition, any toolbar buttons or menu items added with the Force Flow Designer editor will be applied to corresponding toolbars and menus within all operations of the force flow forms.

In Figure 8, a force flow called LinkForceFlow is selected, as indicated by its name in the work area of the display window and the dashed line encircling the sheets called Overview, Details, and Finalize. Clicking on the ForceFlows name box on the right side can also cause the DAD to automatically show available force flows (that is, collections of force flows). These force flows can be added to, copied, and otherwise edited with respect to the current force flow (LinkForceFlow) through the DAD tool user interface. Figure 9 shows a force flow collection editor display that a DAD user may use to view members of force flow collections and select any force flows of interest for viewing and manipulation.

4. Field Flow Designer

The Field Flow Designer component provides the ability to create simple screens and work flow such as View, ViewMany, Edit and/or Find processes using the Dexterra Studio VS.NET plug-in.

A developer using the DAD can load a FieldFlow template (i.e., View, ViewMany, Edit or Find) and launch the designer tool. Then the developer can fill out the Property Sheet for the Template by selecting the Business Object(s) and configuring the screen functions. The Field Flow Designer component then generates the screen including the UI elements and .NET code that binds to the meta data definition of the Business Objects.

The Force Flow Designer (see discussion above) can be used to specify field flows that comprise operations initiated from particular sheets of a force flow. The Force Flow Designer user interface screenshot is illustrated in Figure 8.

5. Component Designer

The Component Designer provides the ability to create mobile screens/forms using one or more custom controls or compound controls that are bound to the metadata definition of one or more Business Objects utilizing the Dexterra Studio "VS.NET" plug-in. Custom Controls are the most basic building blocks such as Text Boxes, Grids, Lists, etc. that are metadata aware. Compound Controls are combinations of Custom Controls that are meta data aware and grouped together to perform certain operations such as 1-1 (one-to-one), 1-M, M-M data functions. Modules are a predefined set of Custom and Compound Controls and/or Screens that perform a basic business function such as Signature Capture, Login, etc.

A developer can create a UI screen/form using the DAD by selecting a base form and dragging Components (custom controls or compound controls) onto the form and configuring the property sheet to bind to one or more meta data configured Business Objects. The DAD includes editor windows of the TransApp Designer function with which sheets, forms, menu items, toolbar buttons, and data mapping relationships can be specified. For example, Figure 10 shows a Menu Item Collection Editor display that a DAD user may utilize to add a menu item to a force flow sheet. Figure 10 shows that a mobile application called SmartApplication is being designed (and/or edited) and that a Properties sub-

window shows properties of the application, with "SubMenuItems" selected (identified by the windows encircled with the oval on the right side) and also shows the Collection Editor sub-window that is generated in response, with a list of properties. When the DAD tool user makes selections in the Collection Editor, appropriate data bindings will be specified for the corresponding Business Objects, ensuring appropriate data transfer operations in the executing mobile platform application.

Figure 11 shows a Toolbar Button Collection Editor display that a DAD user may utilize to add a toolbar button to a force flow sheet. The display window is circled in Figure 11 to identify the display window in front of the underlying DAD application window. Figure 12 shows a Mapping display that a DAD user may utilize to specify data mappings, with the Mappings selection in the Editor window and source business objects and destination business objects specified in the Mapping Collection Editor window and in the Business Object Mapping dialog box. The editor window, identified by the circled area, indicates that objects can be specified for mapping by name from a drop-down list, providing a convenient user interface.

C. MODULAR CONSTRUCTION

As noted above, designer tools are provided to deploy, configure, and adapt operating sequences of the mobile data system with the operational modules called transportable applications ("TransApps"). The TransApps are data objects that are self contained and capable of being linked together with other operating sequences and with other TransApps. As illustrated in the drawings and described above, the user tools permit deployment, configuration, and adaptation of the TransApps through a convenient user interface that does not require knowledge of programming code.

Each TransApp operational module can accept input data and can generate output data. The input data can be received from other modules, or from the application user, or from enterprise data sources. The output data can be provided to other TransApp modules, or to the application server (for enterprise data sources), or can be provided for display on the mobile computing device itself. In this way, modules can be defined for the purpose of particular problem solutions, and such problem solution modules can be reused during other

application design efforts. As described herein, the TransApp operational modules can be managed through a convenient user interface without specialized programming expertise.

Figure 13 is a flow diagram that illustrates operations performed by a computer system in executing a TransApp application module at a mobile application server for processing of data that is shared between multiple enterprise datasources and a mobile client that communicates with an application server. The first operation, at the flow diagram box numbered 1302, is to specify data to be requested from one or more of the multiple enterprise datasources and to specify a mapping of the specified data to a single Defined Data Object at the application server. Next, at 1304, a Connection Object is created (instantiated) that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources, wherein the data interface objects comprise one or more Adapter Objects at the application server and comprise data that is either a data table, a stored procedure, a script, or a data object. At 1306 the system creates one or more Command Objects that perform data actions on the specified data, wherein the Command Objects permit selection of one or more commands on the specified data that include commands including select data, insert data, update data, and delete data. At 1308 a Data Object is defined that specifies one or more data actions in the Command Objects. And at 1310 a View Object is bound to the Defined Data Object such that the View object interfaces with the specified data. Thus, a transportable application is produced that includes the request for the specified data and mapping such that the transportable application comprises a modular computer program application that can be installed at the application server for communication of the specified data with the mobile client, such that the application server will automatically link the transportable application with other transportable applications of the application server in accordance with their respective specified data and mappings.

In accordance with the invention, a computer program tool referred to as "DAD" for use by designers of mobile applications is provided to deploy, configure, and adapt operating sequences of the mobile data integration system with the TransApps. The DAD application program tool provides these features through

the user interface illustrated in the drawings. Thus, the DAD application program tool provides a means for specifying data to be requested from one or more of the multiple enterprise datasources and mapping the specified data to a single Defined Data Object at the Dexterra application server by way of creating and manipulating TransApps. In this way, the DAD application program also provides a means for producing a TransApp transportable application that includes the request for the specified data and mapping such that the transportable application comprises a modular computer program application that can be installed at the application server for communication of the specified data with the mobile client, and such that the application server will automatically link the transportable application with other transportable applications of the application server in accordance with their respective specified data and mappings. Those skilled in the art will appreciate that "producing" a TransApp may involve checking for proper specifications and parameters in the module code, such as verifying data parameters, ensuring proper data object availability, verifying data object bindings, and the like for proper module performance in accordance with the data object configuration of the mobile platform system.

The computer program comprising the DAD tool can be installed on a computer apparatus or system, such as a desktop computer, notebook computer, or the like, so long as the DAD tool program can receive user input to carry out the TransApp configuring process and can verify datasources, bindings, and the like. The configured TransApp can be included within a mobile platform and installed at an application server of the mobile platform such as described above, so that the operational features of the TransApp can be utilized at the mobile clients for operations with the enterprise datasources.

Thus, the TransApps are self contained and capable of being linked together with other operating sequences and with other TransApps. The DAD tool permits deployment, configuration, and adaptation of applications through a convenient user interface that does not require knowledge of programming code. In this way, modular TransApps can be defined for the purpose of particular problem solutions, and such problem solution modules can be reused during other application design efforts. The DAD tool permits management of TransApp

operational modules through a convenient user interface without specialized programming expertise.

The present invention has been described above in terms of a presently preferred embodiment so that an understanding of the present invention can be conveyed. There are, however, many configurations for mobile enterprise data systems not specifically described herein but with which the present invention is applicable. The present invention should therefore not be seen as limited to the particular embodiments described herein, but rather, it should be understood that the present invention has wide applicability with respect to mobile enterprise data systems generally. All modifications, variations, or equivalent arrangements and implementations that are within the scope of the attached claims should therefore be considered within the scope of the invention.

CLAIMS

WE CLAIM:

1. A computer program system for constructing a modular computer program application for processing of data that is shared between multiple enterprise datasources and a mobile client that communicates with an application server, the system comprising:

designer means for specifying data to be requested from one or more of the multiple enterprise datasources and mapping the specified data to a single Defined Data Object at the application server; and

producer means for producing a transportable application that includes the request for the specified data and mapping such that the transportable application comprises a modular computer program application that can be installed at the application server for communication of the specified data with the mobile client, such that the application server will automatically link the transportable application with other transportable applications of the application server in accordance with their respective specified data and mappings.

2. A system as defined in claim 1, wherein the designer means further comprises:

means for creating a Connection Object that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources;

means for creating one or more Command Objects that perform data actions on the specified data;

means for defining a Defined Data Object that specifies one or more data actions in the Command Objects; and

means for binding a View Object to the Defined Data Object such that the View object interfaces with the specified data.

3. A system as defined in claim 2, wherein the data interface objects comprise one or more Adapter Objects at the application server.

4. A system as defined in claim 3, wherein the Adapter Objects interface with the Connection Object to expose the data interface objects as data comprising a data table, a stored procedure, a script, or a data object.

5. A system as defined in claim 2, wherein the means for creating one or more Command Objects permits selection of one or more commands on the specified data that include select data, insert data, update data, and delete data.

6. A method of operating a modular computer program application for processing of data that is shared between multiple enterprise datasources and a mobile client that communicates with an application server, the method comprising:

specifying data to be requested from one or more of the multiple enterprise datasources and mapping the specified data to a single Defined Data Object at the application server; and

producing a transportable application that includes the request for the specified data and mapping such that the transportable application comprises a modular computer program application for operation at the application server for communication of the specified data with the mobile client, such that the application server will automatically link the transportable application with other transportable applications of the application server in accordance with their respective specified data and mappings.

7. A method as defined in claim 6, further including:

creating a Connection Object that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources;

creating one or more Command Objects that perform data actions on the specified data;

defining a Defined Data Object that specifies one or more data actions in the Command Objects; and

binding a View Object to the Defined Data Object such that the View Object interfaces with the specified data.

8. A method as defined in claim 7, wherein the data interface objects comprise one or more Adapter Objects at the application server.

9. A method as defined in claim 8, further including interfacing the Adapter Objects with the Connection Object to expose the data interface objects as data comprising a data table, a stored procedure, a script, or a data object.

10. A method as defined in claim 7, further including creating one or more Command Objects that permit selection of one or more commands on the specified data that include select data, insert data, update data, and delete data.

11. A method of method of operating a modular computer program application for processing of data that is shared between multiple enterprise datasources and a mobile client that communicates with an application server, the method comprising:

specifying data to be requested from one or more of the multiple enterprise datasources and mapping the specified data to a single Defined Data Object at the application server;

creating a Connection Object that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources, wherein the data interface objects comprise one or more Adapter Objects at the application server and comprise data that is either a data table, a stored procedure, a script, or a data object;

creating one or more Command Objects that perform data actions on the specified data, wherein the Command Objects permit selection of one or more commands on the specified data that include commands including select data, insert data, update data, and delete data;

defining a Defined Data Object that specifies one or more data actions in the Command Objects;

binding a View Object to the Defined Data Object such that the View object interfaces with the specified data; and

producing a transportable application that includes the request for the specified data and mapping such that the transportable application comprises a modular computer program application that can be installed at the application server for communication of the specified data with the mobile client, such that the application server will automatically link the transportable application with other transportable applications of the application server in accordance with their respective specified data and mappings.

12. A computer program system for processing of data that is shared between multiple enterprise datasources and a mobile client, the system comprising:

a processor that communicates with the mobile client and interfaces with the enterprise datasources through an application server;

a transportable application module that specifies data to be requested from one or more of the multiple enterprise datasources and specifies a mapping from the specified data to a single Defined Data Object, such that the transportable application module produces output and receives input for communication with the application server and mobile client, such that the processor automatically links the transportable application module with other transportable application modules of the computer program system in accordance with respective specified data and mappings of the transportable application module.

13. A system as defined in claim 12, wherein the transportable application module further specifies:

a Connection Object that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources;

one or more Command Objects that perform data actions on the specified data;

a Defined Data Object that specifies one or more data actions in the Command Objects; and

a View Object that is bound to the Defined Data Object such that the View object interfaces with the specified data.

14. A system as defined in claim 13, wherein the data interface objects comprise one or more Adapter Objects at the application server.

15. A system as defined in claim 14, wherein the Adapter Objects interface with the Connection Object to expose the data interface objects as data comprising a data table, a stored procedure, a script, or a data object.

16. A system as defined in claim 13, wherein the Command Objects permit selection of one or more commands on the specified data that include select data, insert data, update data, and delete data.

17. A computer program system for processing of data that is shared between multiple enterprise datasources and a mobile client, the system comprising:

a processor that communicates with the mobile client and interfaces with the enterprise datasources through an application server;

a transportable application module data object stored in the system, wherein the transportable application module data object specifies data to be requested from one or more of the multiple enterprise datasources and specifies a mapping from the specified data to a single Defined Data Object, such that the transportable application module produces output and receives input for communication with the application server and mobile client, and such that the processor automatically links the transportable application module with other transportable application modules of the computer program system in accordance with respective specified data and mappings of the transportable application module, and wherein the transportable application module further specifies

a Connection Object that exposes data interface objects at the application server that provide access to the specified data at the enterprise datasources, wherein the data interface objects comprise one or more Adapter Objects at the application server and comprise data that is either a data table, a stored procedure, a script, or a data object,

one or more Command Objects that perform data actions on the specified data, wherein the Command Objects permit selection of one or

more commands on the specified data that include commands including select data, insert data, update data, and delete data,

identifies a Defined Data Object that specifies one or more data actions in the Command Objects,

identifies a View Object that is bound to the Defined Data Object such that the View object interfaces with the specified data.