September 2003

A Multi-tier Architecture for Mobile Enterprise Resource Planning

Karl Kurbel
European University Viadrina Frankfurt (Oder)

Andrzej Dabkowski
European University Viadrina Frankfurt (Oder)

Anna Maria Jankowska
European University Viadrina Frankfurt (Oder), jankowska@euv-frankfurt-o.de

In: Uhr, Wolfgang, Esswein, Werner & Schoop, Eric (Hg.) 2003. *Wirtschaftsinformatik 2003: Medien - Märkte - Mobilität*, 2 Bde. Heidelberg: Physica-Verlag

ISBN: 3-7908-0111-9 (Band 1)
ISBN: 3-7908-0116-X (Band 2)

A Multi-tier Architecture for Mobile Enterprise Resource Planning

Karl Kurbel, Andrzej Dabkowski, Anna Maria Jankowska
European University Viadrina Frankfurt (Oder)

Abstract: Mobile computing is changing the behavior of individuals and also of organizations. Instant access to information is beneficial in many business situations. Consequently, core information systems like ERP (enterprise resource planning) systems that today's organizations rely on have to support the mobile behavior of their users. We discuss some architectural considerations for mobile applications and introduce a multi-tier architecture for a mobile ERP system. Two key questions to answer are how to access content of an ERP database from a mobile device, and how to make that content available to a mobile user no matter which device he or she is using. A prototypical implementation based on a real ERP system is described. Open questions and issues for further research are discussed in the concluding section.

Keywords: Enterprise resource planning, mobile ERP, mobile computing, multi-tier architecture

1 Motivation and Background of the Work

Mobility is a general societal phenomenon today, affecting not only people's personal lives but also, increasingly, the behavior of organizations and the requirements and expectations by business partners that organizations have to meet. In many situations timely access to information makes life easier or lets businesses act more effectively. A manager in a critical business meeting will be happier to receive the information that his flight is delayed by two hours on his mobile device than to learn about this mishap at the check-in counter. A sales representative visiting an important customer will certainly appreciate up-to-date order-tracking information when being asked by the customer about the delivery date of an urgently awaited order. For a logistics manager it is not only crucial to have en-route information about unexpected delays and events that might endanger just-in-time delivery but also to be able to react immediately and perhaps re-schedule part of the supply chain according to dynamic priorities that are stored in the company's information systems - no matter whether he is in his office or on

the way to a warehouse in a remote location. Many more examples like these could be given.

As a consequence, an essential requirement for the core information systems that today's organizations rely on is that these systems support the mobile behavior of their users. This trend goes hand in hand with ubiquitous computing [Wei91], i.e. providing access to information and computing power independent of locations and devices.

Not surprisingly "mobile business" has become a major area of information systems research and a growing field of ongoing developments. The market for mobile-business solutions is taking on shape. For Germany alone, a recent study by Berlecon Research predicts an annual market volume of 2 billion Euro up to the year 2005 [Ber01].

Technologies for mobile business are maturing, becoming more and more powerful. The range of mobile devices includes not only cellular phones with WAP (Wireless Application Protocol) and SMS (Short Message Service) or MMS (Multimedia Messaging Service) but also PDA's (Personal Digital Assistants), Smartphones, Palmtops, and more.

Some severe restrictions set by the GSM (Global System for Mobile Communication) standard, in particular slow transmission rates (max. 9,600 bits per second), have already been overcome by the HSCSD (High Speed Circuit Switched Data) and GPRS (General Packet Radio Service) technologies. Those technologies provide theoretical transmission rates of 43.2 kbps and 171.2 kbp/s, respectively. Third generation networks like UMTS (Universal Mobile Telecommunication System) with up to 2 mbp/s will be available soon and further relax the limitations for mobile business imposed by current bandwidths.

Emerging multimedia technologies are enhancing mobile devices to such a level that they can serve as frontend systems in the same way as personal computers. Whereas in Europe the majority of cellular phones still have text and line based displays, in other parts of the world the dissemination of third generation multimedia-based devices has reached a significant level. In Japan, for example, NTT DoCoMo's i-mode service has gained more than 34 million subscribers within 30 months since its launch in 1999, and camera-equiped mobile handsets reached a market share of 10 % in August 2002 already, just a few months after their introduction into the market [Yam02].

In the long run it can be expected that mobile devices will provide similar user interfaces as desktop monitors. With this enhanced frontend functionality, they can serve as mobile clients just like today's laptop, notebook, and personal computers, fostering the convergence of "conventional" data processing and telecommunications.

This trend raises new challenges for business information systems in general and for enterprise resource planning (ERP) in particular. ERP systems are the

information-systems backbone of most enterprises today. Mobile frontends can help to make them more effective in a dynamic environment. In the above mentioned examples, information and functionality needed by the sales representative or the logistics manager are probably stored in the company's ERP system. Mobile access to that system is a prerequisite for flexible actions and reactions.

Taking into account the general trend of data processing and telecommunication growing together, a long-term vision for enterprise resource planning is to make all ERP functionality available independent of particular frontend devices – on mobile high-resolution multimedia phones as well as on traditional desktop clients.

While this vision might still sound futuristic, the first step is to make ERP data available for mobile users. Such data are normally stored in the ERP system's database and managed by a database management system (DBMS). The core questions are thus:

- How to transmit queries by the mobile user from the mobile device to the DBMS used by the enterprise resource planning system, i.e. how to access the data in the database?
- How to transmit and convert such data from the database tables so that they can be displayed on the screen of a mobile device (or more precisely - on screens of a variety of mobile as well as stationary devices)?

The solution to these problems has two major aspects: (1) Accessing the content of the database, and (2) extracting information retrieved from the database and preparing it in a device-dependent manner. These two tasks are solved in our approach by a Content Access Engine and a Content Extraction Engine.

In the subsequent sections, an architecture around those two engines is presented. The Content Access Engine is in charge of retrieving data from a relational database with the help of JavaBeans components, and representing them in an XML format. The responsibility of the Content Extraction Engine is to detect the type of the user's device (e.g. cellphone, PDA, Pocket PC) and to generate interface-specific forms of the XML data in the respective markup language (WML, XHTML, HTML, etc.) for the user's display.

This paper is organized as follows. In the next section, the general architecture for mobile ERP, the underlying concepts, and the technologies used are presented. Section 3 illustrates by means of a specific ERP system how this architecture was implemented in a particular case. Some observations and open questions for further research are discussed in the final section.

2 Architecture for Mobile ERP

2.1 Architectural Considerations for Mobile Applications

The choice of an architecture can affect all aspects of software design, implementation, and maintenance. Important factors to be considered include the business objectives, the complexity of applications based on that architecture, the desired level of integration, the number of users or transactions, and various technical constraints. If the design of the architecture is not appropriate, then the consequences may be that application system development takes longer, maintenance is difficult, and response times in the operation of the system are inacceptable.

Many application systems today have three major layers: presentation layer, business or application logic layer, and services layer [Bri00, pp. 91-106]. The presentation layer provides the user interface and is responsible for the interaction between the user and the device. The application or business logic layer contains the business rules that drive the given enterprise. The services layer provides general services needed by the other layers, usually including database services, file services, print services, and communication services.

The functionalities of these three layers can be assigned to logical entities called tiers. Conventional client-server systems had two tiers, one for presentation and one for the business logic including the database. Mobile and wireless applications are typically deployed with three-tier or multi-tier architectures. The three-tier architecture attempts to overcome some of the limitations of the two-tier scheme by separating presentation, processing (business logic), and data into distinct tiers. It allows for parallel development of individual tiers by application specialists and provides flexible resource allocation. This architecture requires more planning but reduces development and maintenance costs over the long term by leveraging code re-use and elasticity in product migration [Mye02].

In a three-tier architecture the business logic can be entirely or partially contained in a middleware component called application server. If the designer moves most or all of the business logic to the application server and clients are exclusively responsible for the presentation layer, then such clients are called "thin". If significant portions of the business logic are on the client, then the client is considered a "fat" client. Browsers in mobile applications are examples of thin clients – they can connect to different applications and provide a suitable user interface without knowing about the application logic.

The term multi-tier architecture usually refers to the fact that more servers than just one application server are included. An immediate example is a Web server as part of an application that is accessed over the Internet. Generally speaking an

application server can request services from many other servers, i.e. the services themselves use other services to respond properly to the client's original request.

In the project underlying our work the need to develop an architecture arose from the fact that we had to find an effective way to make ERP system data and functionality available on mobile devices. Instead of re-implementing ERP from scratch with the help of tools for mobile computing, we investigated data and functional areas of enterprise resource planning with regard to mobile access. For this purpose, a real-world ERP system for small and medium-size enterprises – *infor:COM* by infor Business Solutions [Inf03] – was analyzed. Indeed, plenty of information appeared worthy to be mobilized. For example, sales orders, quotes, order status, customers' requests, delivery dates, quantities on stock, and many more were found to be quite helpful for sales representatives visiting customers.

The major technical requirement for mobile access to an ERP system is presentation of information in multiple formats. Mobile and wireless devices are equipped with different browsers that support various media formats. It is therefore necessary to deliver the content in different markup languages such as WML, XHTML or HTML [W3C99, WAP02, W3C02]. An architecture should make it easy to add new formats, without changing the existing structure.

Imagine a user with a mobile device browsing a mobile ERP application. He or she navigates through numerous menus and generates requests for the ERP system by choosing from menus displayed on the screen by a mobile browser. In response to such a request the user expects to receive information from the ERP system. This information has to fit the browser; in other words, the content needs to be generated in the specific markup language supported by that browser and well-tailored for the display of the device.

Obviously the presentation of the data retrieved from the ERP systems is device dependent. It depends on the device's manufacturer, on the display size, on the built-in browser, etc. On the other hand, the data as such are not device dependent, so they can be represented in a metadata format. In our approach, these data – on the way from the ERP system to the user's device – are stored in XML format. XML (eXtensible Markup Language) [W3C00] was chosen because it facilitates data exchange between different systems. XML data can be easily processed, transformed to other formats, reordered, filtered, modified, and extended.

Our architecture for mobile applications is browser based and designed for thin client applications. In this architecture, ERP system functionality can be accessed through mobile and wireless devices. The ERP system as such remains unchanged. Our discussion of architectural considerations does not include the architecture of the ERP system but focuses on the additional features that are needed to mobilize such a system.

This architecture is divided into four tiers. The first tier, the data tier, is represented by the *ERP system's database*. The second tier has the application

logic of the "mobilization" task encapsulated in the *Content Access Engine*. Application logic is defined as the processes which “do the work” such as requesting data, returning data, formating data, etc., for example building queries from a mobile user's request for information and preparing the results for processing. The Content Access Engine transforms the data retrieved into XML format. Special data formats were developed to simplify the process of XML generation.

The third tier has the challenging task of device-context aware content delivery to the user. This tier incorporates the presentation logic in the *Content Extraction Engine*. This engine determines the type of the browser and the most important device characteristics, and then tailors the content to significant features of the device. The Content Extraction Engine implements part of the presentation layer. It is, however, not responsible for the graphical user interface (GUI), but rather for the presentation logic [Sun02b].

The fourth tier, on the presentation layer, consists of different mobile and wireless devices like WAP-enabled cellular phones, PDAs, Palmtops, and Pocket PCs with their respective browsers and GUIs.

The constituents of the architecture for mobile applications, the design decisions, and the underlying ideas and the technologies are discussed in the subsequent sections.

2.2 Content Access Engine

Data of an ERP system are usually stored and maintained by a database management system (DBMS). In most cases the business logic resides on special ERP application servers and/or on database servers. Our approach is applicable to both cases. We treat the DBMS as a repository of information where data are stored in tables. The ERP system operates directly on those data, updating, inserting or deleting them. Sometimes ERP systems apply another approach to data manipulation. They store the most frequently requested or the last requested operational data in memory. If data needed are not in the cache they are retrieved from the database upon the first request. This approach of keeping data structures in the cache speeds up data access and improves the overall performance of the ERP system. The cached data are periodically exchanged with data in the database tables.

The *Content Access Engine* described in this section operates directly on the database tables. This approach is more general in the sense that it is applicable to any ERP system using a relational database. If specific cache structures are used the implementation of the Content Access Engine could be modified to operate directly on those data structures.

The general architecture developed for mobile enterprise resource planning is outlined in figure 1.

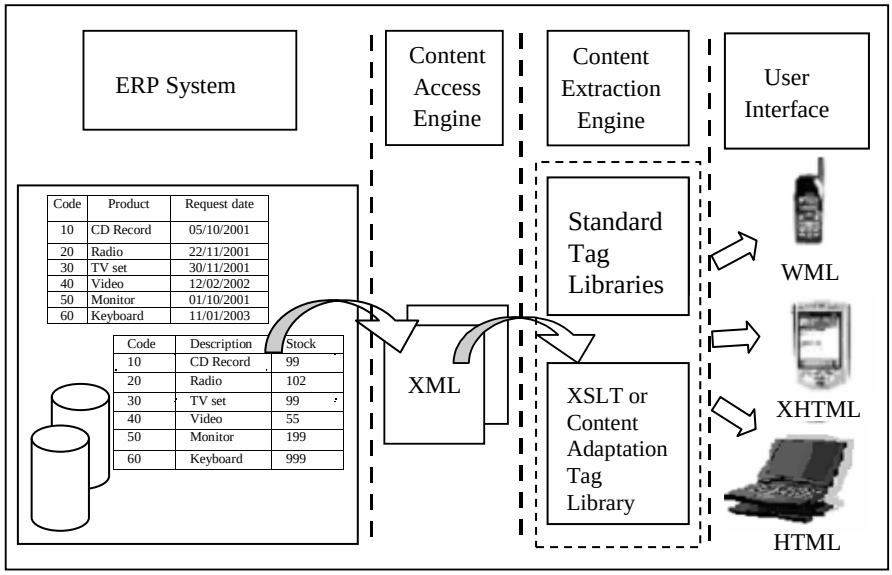

Figure 1: Architecture for mobile ERP

The purpose of the *Content Access Engine* is to transmit data from the ERP database and subsequently convert them into XML-based documents. This procedure consists of several stages. First of all, data gathered from database tables have to be transmitted to the Content Access Engine as shown in figure 2. This task is accomplished by special components implemented as JavaBeans [Sun97]. JavaBeans encapsulate all SQL queries that reflect the physical structure of the requested data in a database [Ger00]. With regard to maintainability and portability of the Content Access Engine, the JavaBeans are the components that have to be modified if the data organization, tables, or dependencies in the database change or if another ERP system wishes to communicate directly with the cache structures mentioned above.

A reasonable and convenient way to store information retrieved from the database for further processing are dynamically built lists (in Java, vectors of vectors). The first row of such a list contains the column names from the database tables. The column names are used as tag names later, in the next step – the generation of XML documents on-the-fly based on the data in the list. However, when the XML documents are created, it is possible to replace the original column names with user-defined names. In this way a developer can provide more meaningful tag names.

Figure 2 illustrates this step: The column name “ID” in the example is changed to “Product Code” in the XML document In case the underlying database uses a country-specific language, e.g. German, Polish or French, the column names retrieved and written to the vector will also be in that language. The generated XML tag names, however, could be changed to language-independent names during the mapping process. The XML documents created in this way serve as input for the Content Extraction Engine.

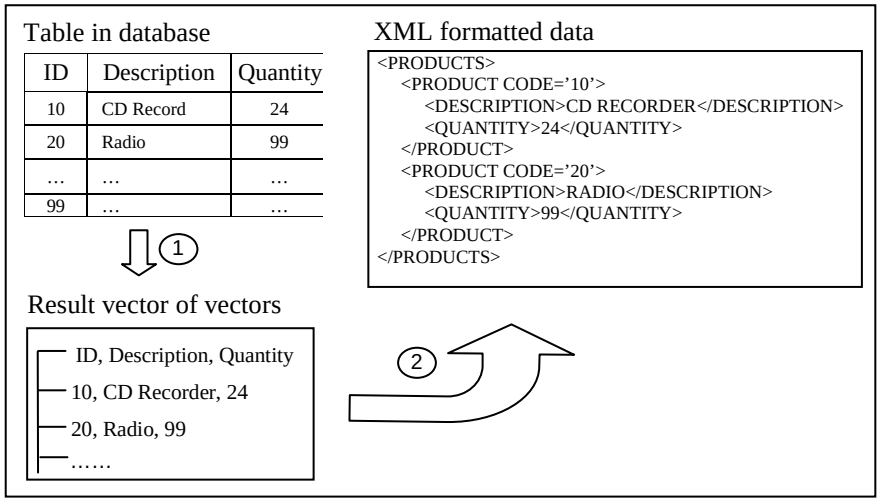

Figure 2: Phases of data transfer from ERP system to XML

2.3 Content Extraction Engine

In the world of mobile computing users can access information systems through a variety of wireless devices with many different characteristics. The devices support different markup languages, have different displays and different features for data input, etc. A key requirement for mobile applications is therefore to adapt content according to the characteristics of the devices. In our approach this functionality is provided by the Content Extraction Engine. The Content Extraction Engine retrieves information about the devices' features from HTTP headers and performs appropriate metadata transformations.

The most popular way to obtain delivery context is to use the HTTP standard Accept headers [W3C02a]. These headers include the supported media types (MIME types), character sets, content encoding, and languages. Additionally, the User-Agent header contains information about the device manufacturer, the version number, the device hardware and the browser used. Information delivered by HTTP headers is, however, subject to different limitations. Since there is no

standard specifing the structure of the User-Agent header, delivery context is up to the device manufacturer.

In our approach content adaptation is based on information about device capabilities retrieved from HTTP headers. This information can be obtained with the help of the `getHeader` method of a request object and is then available during the user's session. The `getHeader` method is a JavaServlet method. It returns the value of the specified header field as a String [Sun02a, p. 58]. All request header names can be obtained as an `Enumeration` object from the method `getHeaderNames()`.

Subsequently the Content Extraction Engine determines the form of presentation depending on the relevant features of a device – supported markup language, graphical formats, size of the display area, browser type, and colors displayed.

In the next step, the metadata generated by the Content Access Engine are transformed according to device-specific characteristics. Two components are available for this transformation: tag libraries and transformation objects with XSLT stylesheets [W3C99a].

Since the Content Extraction Engine is invoked in Java Server Pages (JSP), we were able to benefit from *tag libraries* introduced as a new feature in JSP 1.1. Tag libraries are reusable modules that can build and access programming language objects and influence the output stream. They usually encapsulate frequent tasks and can be used across applications, increasing the speed and quality of development. Tag libraries have access to all objects available to Java Server Pages, they can communicate with each other and can be nested, allowing for complex interactions within a page.

Custom JSP tags consist of three main parts: the JSP page, the tag handler, and the tag library descriptor. The most important part of a tag – the tag handler – is a Java class that processes the tag. The tag library descriptor is an XML file that groups tags into a library and describes the specifics of each tag, such as its attributes, the tag handler's name, short name, and so on [Sun02, pp. 568-602]. The most popular tags were developed as part of the Apache Jakarta Project[1] and are included in the JavaServer Pages Standard Tag Library (JSTL) [Sun02, pp. 607-632]. They provide support for tasks such as database access, XML document manipulation, formatting and internationalization.

Although some simple JSP tag libraries for mobile applications have already been provided by vendors, libraries supporting the development of applications for different devices are not yet available[2]. Therefore we developed a special tag library – the *Content Adaptation Tag Library* – for the generation of appropriate markup elements depending on the device context. This library helps to separate

1 See http://jakarta.apache.org/taglibs/index.html

2 See http://jsptags.com/index.jsp for a list of available tag libraries

the presentation format from the presentation logic. It encapsulates most of the functionalities used in HTML, WML, and XHTML pages. The most important tags are as follows:

- SiteTag – creates the root elements for HTML, WML or XHTML and the title of the page, and optionally specifies a link to an appropriate CSS stylesheet [W3C98] in XHTML and HTML, depending on the detected browser type.
- WMLOnTag (and WMLOffTag) – indicates that the enclosed elements have to be included in (excluded from, resp.) the output for browsers supporting WML.
- CardTag – produces a WML card or a deck of cards.
- TableTag – draws a table with optional attributes like background color or border size.
- TableRowTag and TableCellTag – draw rows and cells of a table and specify their look-and-feel.
- LinkTag – produces link elements in HTML, WML and XHTML.

With these tags a frontend developer can control how data are processed in backend Java components without writing any Java code in the JSP page. The tags communicate with other tags on the same page. The SiteTag, for example, detects the browser type, and all other tags use this information for content adaptation. The tags support the generation of completely different presentation layers depending on the device context. For example, the same data can be displayed on one page in a WWW browser or split into many separate cards for a WML browser.

Content transformed by the Content Adaptation Tag Library can be in the form of regular text or XML data. In the second case, the XML source has to be transformed with the help of XSLT (eXtensible Stylesheet Language Transformations) in order to extract specific data or modify them. A developer needs a stylesheet for this transformation and an XSLT processor invoked from JSP or from a custom tag. The Jakarta Taglibs Project provides two libraries for transforming XML into various formats: JSTL and XTags library [Apa02]. The XTags library was designed for working with XSLT and XPath [W3C99b]. It supports navigation, processing and styling of XML documents directly in JSP. Since the XTags library offers more possibilities to manipulate XML data, this library was chosen for the transformation of metadata produced by the Content Access Engine. The tag library component of the Content Extraction Engine works in the way illustrated in figure 3.

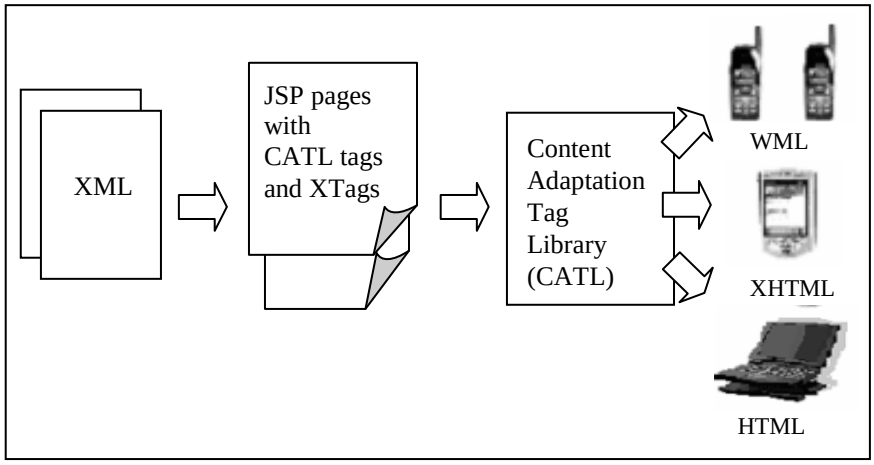

Figure 3: Custom tag library component of the Content Extraction Engine

The second component of the Content Extraction Engine are *transformation objects* with XSLT stylesheets. XSLT enables the transformation of XML data into virtually any format. The most popular formats are WML, HTML, XHTML, and SVG (Scalable Vector Graphics) [W3C01]. For the transformations, it is necessary to prepare a number of stylesheets. An XML document then can be parsed and modified according to the respective stylesheet. In our approach the Java Transformation API for XML (TrAX) [Sun02, p. 120] was chosen to invoke XSL stylesheets from Java programs. TrAX is capable of compiling stylesheets and holding them in memory, thus improving the performance significantly.

Usually TrAX takes an XML file and a stylesheet as input to create an output representing the result of the transformation. For better performance our stylesheets are compiled when they are used for the first time. Then they can be re-used in multiple transformations. Compiled stylesheets are gathered in a Map[3] composed of Templates objects. These objects are provided by TrAX and stored in memory. The Map of Templates objects plays the role of a cache.

Up to now some sample stylesheet templates with typical elements of HTML, WML or XHTML pages have been provided for the transformations. Developers can make use of these templates directly or import them and override variables or templates included. In the next step we intend to build a library of reusable stylesheets for generating HTML, WML and XHTML.

[3] See http://java.sun.com/j2se/1.4.1/docs/api/java/util/Map.html

3 An Example of Mobilizing a Real-world ERP System

3.1 Restrictions and Design Considerations

Technologies for mobile business are offering more and more powerful features, particularly in the field of multimedia. Multimedia-enhanced mobile devices are gaining more market share, yet the majority of existing devices in Europe are still equiped with small screens that can display only few lines of text and simple graphics. Designers of mobile frontends have to take those limitations into account and eventually prepare additional layout versions for devices with enhanced multimedia features.

The type of device from which a particular user sends a request can be detected automatically and likewise the characteristics of the device can be determined. Therefore it is possible to generate an appropriate layout version of the response and send that version to the device. The frontend of a PC-based client will be different from a text and line oriented frontend of a phone-based client, so a transformation is necessary.

A significant problem in mobilizing today's ERP systems is the low bandwidth of telecommunication networks. It is time and cost ineffective to send larger portions of data to mobile devices. Instead, all processing should be done on an application server and only the results should be transmitted in a compact form to the mobile device. This type of distribution of work is likely to change with third-generation networks like UMTS but currently it is the most reasonable way.

Another shortcoming to be considered is the low processing power of mobile devices. As a consequence, only simple things like validating input can be done on the device's side while the mobile application logic must remain on an application server. It should also be noted that sometimes it is difficult or not profitable from a business point of view to mobilize certain ERP functionality. We took these aspects into account when developing a prototypical solution for a real-world ERP system, *infor:COM* by infor Business Solutions. This system aims at small and medium-size enterprises and has a good market penetration in Central Europe.

Developers of mobile solutions can make use of simulators for mobile devices provided by manufacturers and other companies. Market leaders like Nokia, Ericsson and Phone.com offer toolkits for the development of mobile applications free of charge. These toolkits contain simulators that emulate real mobile devices. The simulators have display areas and buttons like the real devices. They are very well suited for prototyping of mobile applications, making development, testing, and debugging a lot easier [Kur+02].

Before selecting those ERP application domains which appeared worth to be enhanced with mobile access, an empirical study of the state-of-the-art in this field was done (for detail results see [Kur+03]). Then the infor:COM system was analyzed with respect to application areas which are both interesting from a business point of view and feasible taking the technical limitations into account. infor:COM like all ERP systems covers a wide range of solutions for many sectors of an enterprise. It provides modules for finances, sales, stock, manufacturing, planning, purchasing, etc. As a result of our studies several application areas were chosen, starting with sales and distribution, in particular with quotes and customer orders.

3.2 Content Access and Content Extraction

Standard *infor:COM* applications have a forms oriented user interface, spreading information all over a conventional screen of a large monitors. Screen content can be quite complex, showing many data and subforms at the same time.

Considering the small display sizes of mobile devices, it is obviously not possible to "translate" an existing ERP frontend design to a mobile frontend one-to-one. The developer has to concentrate on important information instead and adopt only the really essential data from the standard screens of the desktop-based application. When all data are significant, or when it is hard to decide which ones are the most important, then the data have to be distributed across several screens of a mobile device ("cards" in the terminology of mobile browsers). These design principles were applied in the translation of conventional infor:COM forms into mobile cards.

As an example, processing quotes in the infor:COM sales module is described. Figure 4 shows the corresponding user interface on a desktop client. A standard menu is displayed in the frame on the left part of the screen. Each menu item has a label with the module name like sales, planning, etc. The main menu items can be expanded into submenus etc. which are then visible at the same time as the main menu.

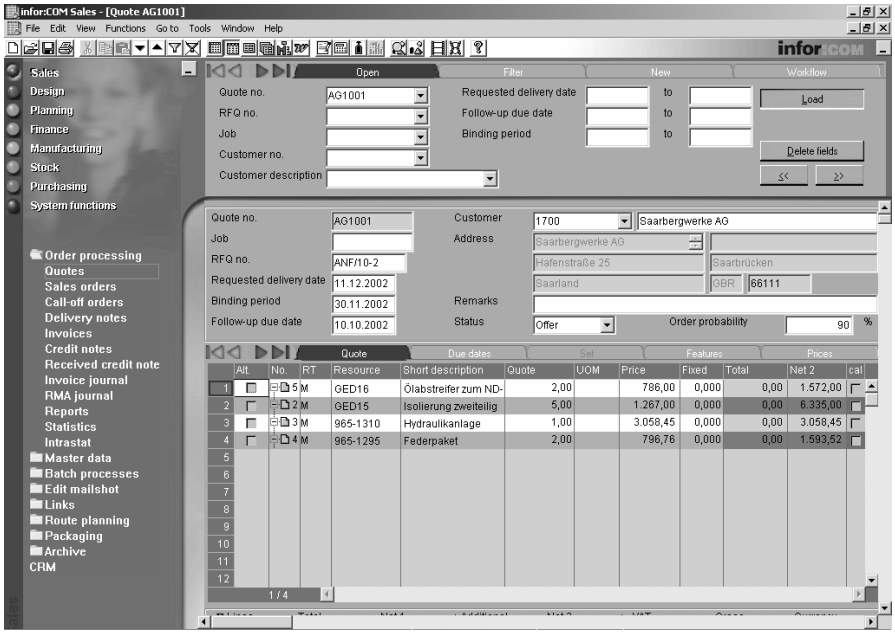

Figure 4: Selecting quotes from desktop-oriented ERP user interface

It is quite obvious that this type of menu has to be re-designed because it cannot be displayed in the same form on the screen of a mobile device. Therefore the menus in our mobile ERP application are displayed level by level. When the user starts navigating on the top level and clicks on a menu item, he or she is redirected to menus detailing the functions of the chosen module. Finally the user reaches a card with the desired functionality.

Figure 4 shows a search for quotes that match some condition specified by the user. An analogous process was implemented for a user with a *mobile* device. Assuming that the user looks for a particular quote (the same one as in figure 4, with quote no. "AG1001"), he or she navigates through a sequence of menus to reach the desired card. The functionality drafted in figure 5 is the same as for the desktop-based screen of figure 4. For example, the user can select a certain way of filtering quotes (by quote number, RFQ number, customer no., etc.). From the list of quotes displayed afterwards he or she can select the desired one. If the list of results is long, only a few items are displayed at a time, i.e. the list is stretched over several cards. Likewise the details about quotes are also divided into a number of cards.

Figure 5 illustrates this scenario. Obviously the user notices only the frontend displaying ERP data as requested, but the requests and responses are transferred across the overall architecture as described in section 2. In order to look for information the user fills input fields on a mobile screen. In this way he or she

determines the criteria for the search. The user's input is treated as parameters and passed to the Content Access Engine where it is further processed behind the curtain.

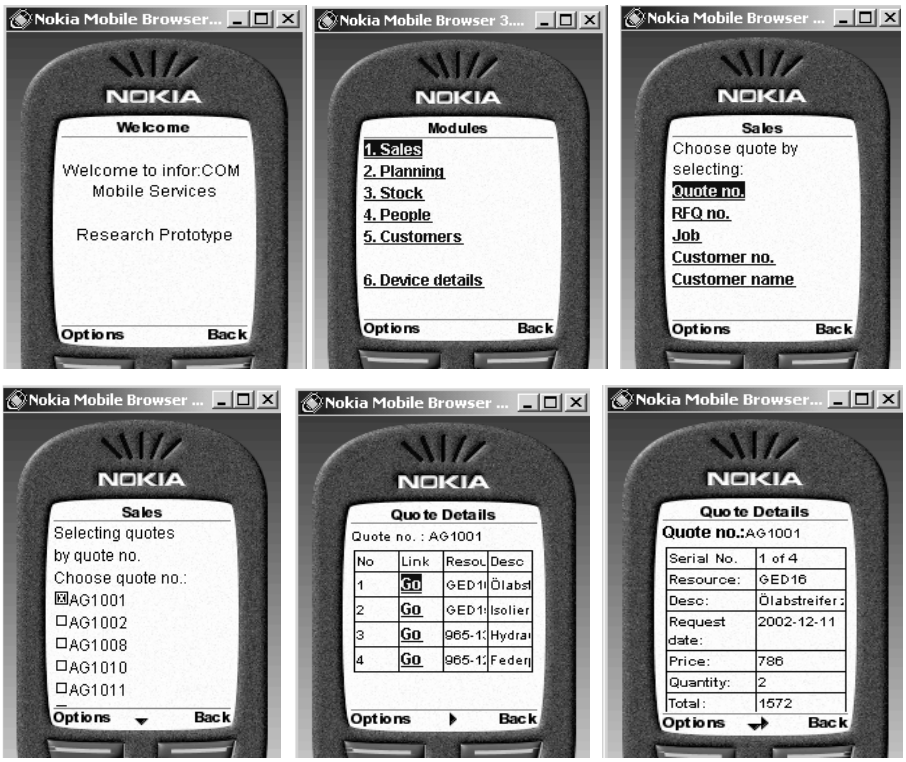

Figure 5: Selecting quotes from ERP database

Within the Content Access Engine, JavaBeans are responsible for communicating with the tables of the ERP system's database. To be more precise, appropriate JavaBean methods are executed with parameters from the user in order to obtain data that match the user's request. The data retrieved are stored in a vector of vectors and then transformed into XML-formatted data. The Content Access Engine forwards the XML data to the Content Extraction Engine.

Employing information about actual device capabilities, the Content Extraction Engine transforms the data represented in XML to an appropriate markup language with the use of either special tag libraries or XSLT stylesheets. XSLT stylesheets are recommended for more a sophisticated design of mobile screens. If the result set of a user request contains too many element to be displayed on one screen, the Content Extraction Engine automatically spreads these data over as

many screens as needed. No additional processing and communication with the database is necessary to view the data on multiple cards. For each user request, e.g. when the user provides new search criteria or starts looking for different information, the Content Access Engine and the Content Extraction Engine repeat the steps outlined above.

4 Open Issues and Further Work

The architecture presented in this paper is suitable for the current and near-future state-of-the-art of mobile computing available to the average ERP user. As the pace of development is rapid in the field of mobile applications and technologies, more powerful features will be available in the near future. With the introduction and dissemination of third-generation networks and devices, more advanced functionalities and user interfaces will be possible. For example, an order-entry form on a UMTS-enabled PDA might come quite close to the one shown in figure 4.

The architecture discussed in the previous sections has been successfully applied in a prototypical implementation for a particular ERP system, infor:COM. Some starting points for improving the basic components of that architecture – the *Content Access Engine* and the *Content Extraction Engine* – have become evident in our work.

Currently each user request initiates a process of generating XML-formatted data. If the user needs the same data more than once then the entire process of content access and extraction has to be repeated. Or consider two different users requesting the same data. Two separate but basically identical XML documents will be generated by the *Content Access Engine*. A more efficient solution would be to store the XML documents generated last on an application server in a similar way as in a cache. This raises the question whether segments in the cache should belong to a single user only or can be shared by different users. Further research in that direction is necessary.

Another restriction is that the context framework of the Content Extraction Engine comprises only basic information transported in the HTTP headers. One future task will be the integration of comprehensive context models. By adding new contextual information to the Content Extraction Engine it will be possible to deliver personalized information, e.g. individual presentation styles according to user preferences.

This enhancement can be achieved by using new standards for contextual information like the CC/PP model developed by W3C [W3C00a] or UAProf invented by the WAP Forum [WAP01]. *CC/PP (Composite Capabilities/ Preference Profiles)* specifies a structure for device profiles based on the XML

serialized Resource Description Format (RDF) [W3C99c]. CC/PP has interesting properties for mobile applications, helping to reduce the amount of information to be sent via low-bandwidth wireless networks. *UAProf (User Agent Profile)* is an implementation of CC/PP for WAP-enabled mobile terminals, supporting the convergence of mobile web technologies with conventional web technologies. Vendors of WAP browsers are beginning to offer commercial implementations of UAProf. For constructing device profiles an open-source library, DELI (DElivery Context LIbrary for CC/PP and UAProf), can be used [But02].

Finally, the *Content Adaptation Tag Library* we developed will be enriched in future work with additional functionality. New tags will be developed, e.g. tags for generating forms, input fields, etc. and tags to support other formats (e.g. cHTML, SVG).

In this paper the focus was essentially on accessing the database of an enterprise resource planning system from a mobile device. The next step towards the vision of making ERP functionality available independent of particular frontend devices will be to invoke genuine ERP functions from a mobile client. As an example, re-consider the sales representative visiting an important customer mentioned in the introduction. The customer's satisfaction would certainly rise significantly if the salesperson could promise (and guarantee) an earlier delivery date. To make a reliable statement the salesperson would need to trigger a scheduling function in the ERP system's production or logistics module. While the frontend for this task is quite simple to develop, accessing and/or invoking internal functions of an ERP system is nontrivial unless an API (application programming interface) is available. Since only few ERP systems truly encapsulate their functionalities in an object-oriented manner and provide such interfaces, a lot of reengineering work is necessary. In our project this work is currently going on.

References

[Apa02] Apache Jakarta: XTags library: Tags for working with XML, XPath and XSLT, Version 1.0, 2002. http://jakarta.apache.org/taglibs/doc/xtags-doc/index.html. Download 2002-12-29.

[Ber01] Berlecon Research (Eds.): Mobile Business Lösungen für Unternehmen. Studie der Berlecon Research GmbH Berlin, 2001. http://www.berlecon.de/studien/index.html. Download 2003-01-30.

[Bri00] Britton, Ch.: IT Architectures and Middleware: Strategies for Building Large, Integrated Systems. Addison-Wesley: Boston, 2000.

[But02] Butler, M.H.: DELI: A delivery context library for CC/PP and UAProf, External Technical Report HPL-2001-260, 2002. http://www-uk.hpl.hp.com/people/ marbut/ DeliUserGuideWEB.htm. Download 2003-01-03.

[Ger00] Gertz, M.: Oracle/SQL Tutorial, 2000. http://www.db.cs.ucdavis.edu/teaching/sqltutorial. Download 2003-01-18.

[Inf03] http://www.infor.de. Download 2003-01-18.

[Kur+02] Kurbel, K., Dabkowski, A., Zajac, P.: Software Technology for WAP Based M-commerce - A Comparative Study of Toolkits for the Development of Mobile Applications; in: Proceedings of the International Conference WWW/Internet 2002 (IADIS); Lisbon, Portugal; November 2002.

[Kur+03] Kurbel, K., Teuteberg, F., Hilker, J.: Mobile Business-Anwendungen im Enterprise Resource Planning: Mobilitätspotentiale entlang der ERP-Funktionskreise; Industrie Management 19 (2003) 1, pp. 72-75.

[Mye02] Myerson J.M.: The Complete Book of Middleware, Auerbach Publ., Philadelphia 2002.

[Nok02] Nokia: WAP June Overview, 2002. http://www.forum.nokia.com. Download 2003-01-07.

[Pfe01] Pfeifer, C.: XML Processing with TraX, 2001. http://www.onjava.com/pub/a/onjava/2001/07/02/trax.html. Download 2003-01-20.

[Sun02] Sun Microsystems: The Java Web Services Tutorial, 2002, http://java.sun.com/webservices/download.html. Download 2003-01-12.

[Sun02a] Sun Microsystems: JavaServer Pages™ Specification, Version 2.0, 2002. http://jcp.org/aboutJava/communityprocess/first/jsr152/. Download 2003-01-15.

[Sun02b] Sun Microsystems: Designing Enterprise Applications with the J2EE Platform, 2002. http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/DEA2eTOC.html. Download: 2003-01-24.

[Sun97] Sun Microsystems: JavaBeans, 1997. http://java.sun.com/products/javabeans/docs/spec .html. Download 2003-01-22.

[W3C00] W3C: Extensible Markup Language (XML) 1.0 (Second Edition), 2000, http://www.w3.org/TR/REC-xml. Download 2003-01-23.

[W3C00a] Composite Capabilities/Preference Profiles: Terminology and Abbreviations, Working Draft, 2000. http://www.w3.org/TR/2000/WD-CCPP-ta-20000721/, Download 2002-12-20.

[W3C01] W3C: Scalable Vector Graphics (SVG) 1.0 Specification, 2001. http://www.w3.org /TR /SVG/. Download 2003-01-06.

[W3C02] W3C: XHTML™ 1.0 The Extensible HyperText Markup Language Specification, 2002. http://www.w3.org/TR/xhtml1/, Download 2003-01-23.

[W3C02a] W3C: Delivery Context Overview for Device Independence, 2002. http://www.w3c.org/2001/di/public/dco. Download 2003-01-17.

[W3C98] W3C: Cascading Style Sheets, Level 2 CSS2 Specification, 1998. http://www.w3.org/TR/REC-CSS2. Download 2003-01-03.

[W3C99] W3C: HTML 4.01 Specification, 1999. http://www.w3.org/TR/html4/. Download 2003-01-23.

[W3C99a] W3C: XSL Transformations (XSLT) Version 1.0, 1999. http://www.w3.org/TR/xslt. Download 2003-01-02.

[W3C99b] W3C: XML Path Language (XPath) Version 1.0, 1999. http://www.w3.org/TR/xpath. Download 2003-01-05.

[W3C99c] RDF Model and Syntax Specification, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Download 2003-01-12.

[WAP01] WAP Forum: UAProf, 2001, http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20010530-p.pdf. Download 2002-12-29.

[WAP02] WAP Forum: Wireless Application Protocol WAP 2.0, Technical White Paper, 2002. http://www.wapforum.org/what/WAPWhite_Paper1.pdf. Download 2003-01-23.

[Wei91] Weiser, M.: The Computer for the 21st Century; Scientific American 265 (1991) 3 (Sept.), pp. 94–104.

[Yam02] Yamakami T.: Leveraging Information Appliances: A Browser Architecture Perspective in the Mobile Multimedia Age; in: Chen, Yung-Chang et al. (Eds.): Proceedings of 3rd IEEE Pacific Rim Conference on Multimedia (PCM 2002), Taiwan, December 2002, pp. 1-8.