ARCHITECTURE FOR MULTI-CHANNEL ENTERPRISE RESOURCE PLANNING SYSTEM

Karl Kurbel, Anna Maria Jankowska, and Andrzej Dabkowski
European University Viadrina, Chair of Business Informatics
POB 17 86, 15207 Frankfurt (Oder), Germany

Abstract: Mobile computing is changing the behavior of individuals and organizations. Instant, multimodal access to information is beneficial in many business situations. Consequently, core information systems like Enterprise Resource Planning systems that today's organizations rely on have to support the mobile behavior of their users. In this paper we discuss some architectural considerations for multi-channel applications and introduce a four-tier architecture for a mobile ERP system. Two key questions to answer are how to access content of an ERP database from heterogeneous mobile devices, and how to make that content available in different formats to a mobile user. 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: mobile computing, ERP system, multi-tier architecture, multi-channel applications, graphical and vocal user interfaces.

1. INTRODUCTION

An increasingly important requirement for the core information systems in enterprises is to provide support for the mobile behavior of their users. This trend goes hand in hand with ubiquitous computing (Weiser, 1991), i.e. guaranteeing access to information and computing power independent of locations and devices.

Network technologies for mobile business are maturing, becoming more and more powerful. With the introduction of third generation networks like UMTS (Universal Mobile Telecommunication System) with transfer rates up to 2 Mbps the limitations imposed by narrow bandwidths are relaxed. In

Japan, NTT DoCoMo's third generation i-mode service was launched in 1999 already (Yamakami, 2002).

In the long run, it can be expected that mobile devices will provide similar user interfaces as desktop monitors. This trend raises new challenges for business information systems in general and for enterprise resource planning (ERP) in particular. A long-term vision for ERP systems is to make all functionality available independent of particular front-end devices – on mobile high-resolution multimedia phones as well as on traditional desktop clients.

Our first step towards this vision 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 of the mobile user from the mobile device to the DBMS used by the Enterprise Resource Planning system, and how to transmit and convert such data from the database tables so that they can be displayed on the screen of a mobile device. The two major aspects of a solution are:

1. Accessing the content of the database.
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 with Cache Storage Structures* and a *Content Extraction Engine*. In the subsequent sections, an architecture around these two engines is presented. The Content Access Engine (CAE) is in charge of retrieving data from a relational database and representing them in an XML (W3C, 2004) format. The responsibility of the Content Extraction Engine is to detect the type of the user's device and to generate device-specific forms of the XML data in the respective markup language 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. ARCHITECTURAL DESIGN OF MOBILE ERP

2.1 General Considerations for Mobile Applications

Typical application systems today have three major layers: presentation layer, business or application logic layer, and services layer (Britton, 2000,

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*. Mobile applications are typically deployed with three-tier or multi-tier architectures. Such architectures allow for parallel development of tiers by application specialists and provides flexible resource allocation. They require more planning but reduce development and maintenance costs over the long term by leveraging code re-use and elasticity in product migration (Myerson, 2002).

In 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. The major technical requirement for mobile access to an ERP system is presentation of information in multiple formats. 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 (WAP Forum, 2002), XHTML (W3C, 2003a) or HTML (W3C, 1999). An appropriate architecture should make it easy to add new formats, without changing the existing structure. In addition, many mobile devices are not only equipped with a browser but also support J2ME (Java 2 Platform, Micro Edition). This technology offers better graphical user interfaces than WML or XHTML (Hemphill & White, 2002).

Due to recent advances in digital speech processing technologies and the emergence of new, non-proprietary standards such as Voice Extensible Markup Language (VoiceXML) (W3C, 2003) and Speech Application Language Tags (SALT) (SALT Forum, 2002), it is now possible to enhance mobile applications with voice user interfaces (VUIs) based on speech recognition and synthesized voice output. Although ERP systems are not traditional telephony-based services they can also benefit from additional voice-enabled interfaces. VUIs can be deployed for retrieving information from a database or manipulating data in the database. Voice input could be applied for entering new data or editing existing data. Voice-enabled interfaces can enhance alternative access methods and allow users to interact with an application in a variety of ways, using speech, keyboard, stylus, etc. (so-called multimodal access). Each of these modes can be used independently or concurrently.

Our architecture for mobile applications is designed for thin-client (browser-based) and fat-client (J2ME) applications. In this architecture, ERP

system functionality can be accessed through mobile and wireless devices. The ERP system as such remains unchanged.

The 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 specific application logic of the "mobilization" task encapsulated in the Content Access Engine with Cache Storage and RFC Server. Application logic is defined as the processes which "do the work" such as requesting data, returning data, formatting 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. It takes into account the device's characteristics and manages the dispatching of information retrieved in portions. The entire result set is kept and managed in user-specific cache structures on the application server. The amount of data that can be served in one portion depends on the display of a particular device. Special data formats were developed to simplify the process of XML generation.

A Remote Function Call (RFC) Server is used so that ERP functions can be invoked by remote clients. If a mobile user wants to add or modify ERP data, appropriate functions of the ERP system are activated.

The third tier has the challenging task of device-context aware content delivery to the user, incorporating 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 the presentation logic (Paragon, 2003).

The Content Access Engine and the Content Extraction Engine are placed on the Tomcat 5 application server (Apache, 2005a). Tomcat offers clustering and load balancing capabilities that are essential for deploying scalable and highly available Web applications. To guarantee such application features we applied the vertical scaling type of clustering with four instances of Tomcat running on a single machine. In the future we plan to extend our solution with horizontal scaling clustering capability (Apache, 2005).

The fourth tier consists of different mobile and wireless devices like WAP-enabled cellular phones, PDAs, Palmtops, and Pocket PCs with their respective browsers and GUIs. J2ME applications and vocal applications serve as additional user interfaces.

2.2 Content Access Engine and Caching Mechanism

The general architecture underlying our discussion is outlined in Figure 1. The Content Access Engine operates on database tables, as data of an ERP system are usually stored and maintained by a database management system (DBMS). For the interaction between an ERP system and a DBMS two solutions are commonly used. The ERP system can operate directly on data maintained by the DBMS, updating, inserting or deleting them. Alternatively, the most frequently requested or the last requested operational data are stored in memory and accessed there. If the needed data is not in the cache it is retrieved from the database upon the first request. The cached data are periodically exchanged with data in the database tables.

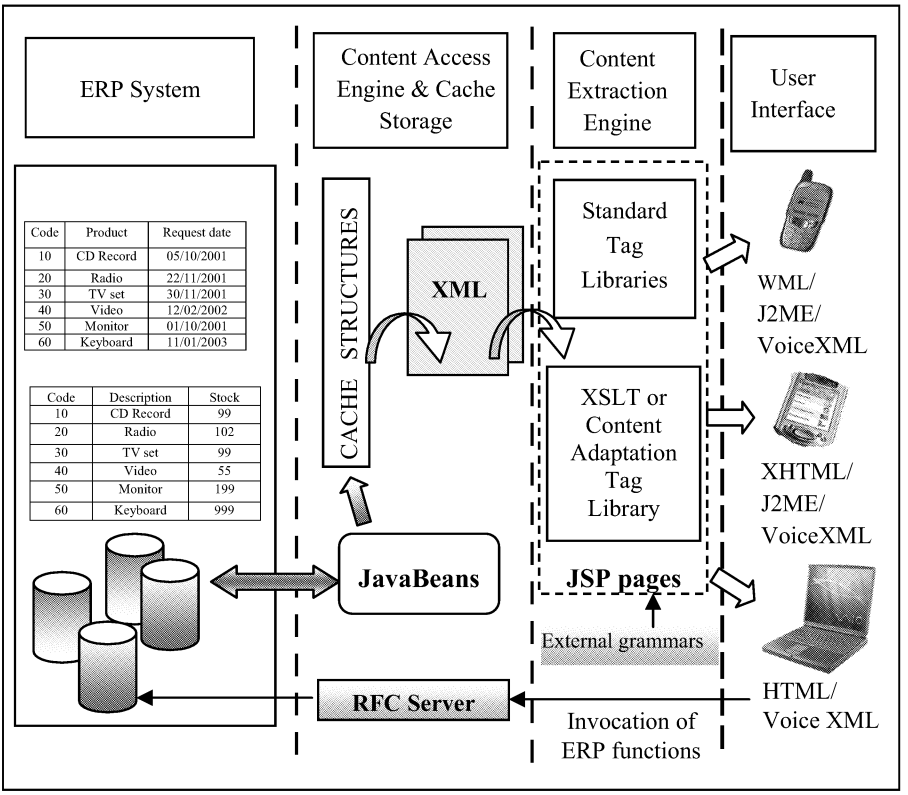

Figure 1. Architecture for mobile ERP

A cache structure is an integrated component of the Content Access Engine (CAE). It speeds up data access and improves the overall performance. The cache is designed to support and maintain the representation of result data. It is initiated with a list of results whenever a

user sends a request for new data. The cache structure belongs to a single user and it is accessible only to him/her. The cache is assigned to the user's session. It is freed when the user disconnects from the mobile ERP system or when the session expires.

Portions of data are served from the cache. Separate sets of parameters describing the devices' displays are assigned to different browsers and device types in the Content Extraction Engine. Subsequently, the parameters are passed to the cache when a browser communicates with the ERP system requesting data.

The task of retrieving data from a DBMS is accomplished by special components implemented as JavaBeans (Sun Microsystems, 1997). JavaBeans encapsulate all SQL queries that reflect the physical structure of the requested data in a database (Gertz, 2000). With regard to maintainability and portability of the CAE, JavaBeans are the components that have to be modified if the data organization, tables, or dependencies in the database change.

A reasonable way to store information 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 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.

2.3 Content Extraction Engine

A key requirement for mobile applications is to adapt content received from the ERP system to the characteristics of a specific device. In our approach this functionality is provided by the Content Extraction Engine. The Content Extraction Engine retrieves information about the devices' features and performs appropriate metadata transformations.

Some mobile operators use special proxies to adapt the content to the properties of mobile devices. Content transformation may also be performed directly on a client with the help of a mobile browser. Opera provides both solutions – it offers a Mobile Accelerator proxy and a special browser for handsets (Opera, 2005). The browser uses a set of style sheets to squeeze the content to the size of a mobile screen and scales images appropriately. The Mobile Accelerator reduces the page size, compresses the images and eliminates unnecessary content (e.g. banners). Both solutions yield pages tailored to a mobile screen but pagination or transformation of the site structure is not possible. Other known transcoding proxies (e.g.

WebAlchemist or Web Intermediaries) can perform advanced transformations based on transcoding heuristics and annotations, such as modifications of the site structure and of navigation modalities (Hwang, Seo and Kim, 2003; Hori et al., 2000). Special browsers and transcoding proxies, however, can provide only limited support in our solution since we already make content available tailored to the specific mobile device. Transcoding approaches or browsers may modify the content but only minor changes are left.

The most popular way to obtain delivery context is to use the HTTP standard Accept headers (W3C, 2002). 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 hardware, and the browser used. Integration of comprehensive context models can be achieved by using standards for contextual information like the CC/PP model developed by W3C (W3C, 2000) or UAProf introduced by the WAP Forum (WAP Forum, 2001).

In our approach content adaptation is based on information about device capabilities retrieved from HTTP headers or CC/PP profiles, if they are available. 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 (W3C, 2003b).

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 (Sun Microsystems, 2002).

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. 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 the presentation format from the presentation logic. It encapsulates most of the functionalities used in HTML, WML, and XHTML pages.

The second component of the Content Extraction Engine are transformation objects with XSLT stylesheets. XML data can be transformed with XSLT into virtually any format. The most popular formats are WML, HTML and XHTML. 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) (Pfeifer, 2001) was chosen to invoke XSLT stylesheets from Java programs. TrAX is capable of compiling stylesheets and holding them in memory, thus improving the performance significantly. Up to now we have built a library of reusable stylesheets for generating HTML, WML, XHTML, VoiceXML and formatting XML to a special format for J2ME applications.

2.4 J2ME Front-ends

Presenting content in markup languages such as WML or HTML reduces the functionality of the handheld device to not more than a simple browser. The user interface is limited by the available markup elements, and when the user is not connected the browser is useless. Powerful mobile applications need user interfaces similar to those of desktop computers and functionalities beyond the scope of mobile browsers.

The Java 2 Platform, Micro Edition (J2ME) meets these requirements at least to some extent. J2ME was designed to accommodate a variety of embedded and hand-held devices. It is composed of both a configuration and a profile. A configuration consists of a virtual machine, core libraries, classes, and APIs. The configuration layer defines the minimum set of Java virtual machine features and Java class libraries available on a particular category of devices. The profile defines the minimum set of APIs available for a particular family of devices representing a given vertical market segment. Profiles are implemented on a particular configuration, and applications are written for a particular profile.

The Mobile Information Device Profile (MIDP) provides a set of Java APIs specific to a particular category of devices (cell phones, PDAs, etc.). In J2ME data can be cached on the client with the help of the MIDP Record Management Store (RMS) API and sent to a database when a connection is established (Riggs, Taivalsaari and VandenBrink, 2001). This solution is very helpful when the connection is suddenly interrupted. It can be applied for incoming and outgoing connections. Instead of temporarily caching data on the server, the RMS stores them on the mobile device and can work independently of the network connection. The same is true for entered data – they are first saved into the RMS and then sent to the database if a connection is available.

J2ME-based applications cannot directly connect to a database using the JDBC mechanism – therefore a special approach for the communication with a data source taking existing tiers into account is needed. Since J2ME technology does not provide any APIs for data connectivity, midlets have to be used to connect to a web server and get the data. Some kind of middleware (e.g. in form of JSP) is necessary for the communication with a database. In our architecture the Content Access Engine is responsible for data delivery in browser-based and J2ME applications. Since the data are represented in XML format, they can be used in J2ME applications, too. The entire structure of the CAE remains unchanged.

A mobile application based on the Java programming language organizes its graphical user interface as several forms in one file (midlet). Therefore the data in XML format cannot be processed in the same way as in an application based on a mobile browser. J2ME applications still need to manage operational memory and storage efficiently. From this point of view it is better to parse XML-based data somewhere else than on a mobile client. Keeping in mind these constraints, the Content Extraction Engine (CEE) can still be applied because JavaServer Pages can be used as a middleware between a MIDP front-end and XML data.

The Content Extraction Engine detects if a response is for a J2ME application. Then it transforms the data from XML format as created by the Content Access Engine to textual information, using either special XSLT stylesheets or tag libraries. Each line of text contains several fields with information items separated by spaces. A single line can be treated as one record in a database. The fields can be simply divided into tokens and displayed in a J2ME-based application on a form. If the set of results includes many records the textual sheet is made up of more than one line. In such a case all records, or a certain part of the records, are stored on the mobile device with the help of RMS.

2.5 Vocal Interfaces

Currently customer services can be provided via the Web and the telephone. In order to obtain telephone services the user needs a device that has audio interaction capabilities. The customer has to call an interactive voice response (IVR) platform that possesses audio input, output, telephony functions and its own service logic as well as a transaction server interface. Companies working on IVR systems developed their own markup languages for their applications, but customers were reluctant to invest in a proprietary technology. In 1999, AT&T, IBM, Lucent Technologies, and Motorola formed the VoiceXML Forum to establish and promote a new, non-proprietary standard based on the eXtensible Markup Language (XML) -

Voice eXtensible Markup Language (VoiceXML). VoiceXML is a language that has features to control audio output, audio input, presentation logic, call flow, telephony connections, and event handling for errors. It serves as a standard for the development of powerful speech-driven interactive applications accessible from any phone. With VoiceXML application developers do not have to care about issues such as concurrent threads of control, resource provisioning, and platform-specific APIs. The language also accommodates platform differences for services that place less importance on portability than on utilizing a new speech technology provided by an individual vendor.

Aural interfaces to the mobile ERP system are based on the VoiceXML standard. For static content (e.g. help information) digitized audio output (recorded voice) in form of mono 8KHz 8-bit u-Law .au-files was used. This format was chosen for performance reasons – files do not need as much storage and download time as the 16-bit linear files (IBM 2003, pp. 57-58). Synthesized speech was applied for the content retrieved dynamically from a database. The data extracted from a database are in the same XML-format as in the previous cases. They were transformed to the VoiceXML format using predefined XSLT style sheets. The Content Adaptation Tag Library was not used for the transformation due to the high complexity of the output structures produced. This degree of complexity was not encountered in the cases of the other formats. Grammars are generated dynamically from a database. For contextual help simple external grammars are utilized. For the design and test phases of the aural user interfaces, the IBM Voice Server Development Kit (IBM, 2002) which fully supports the VoiceXML standard was applied.

2.6 RFC Server

Reading ERP data and displaying them on a mobile appliance is important but not enough. Even more important is to make ERP functionality available to the mobile user. Complete integration of mobile devices with back-office systems is one of the targets for successful implementation of mobile solutions (Vaidyanathan, 2001).

Making ERP application functions available on a mobile device is a challenging task. It is more difficult than just accessing data in relational database tables. A function like re-scheduling a production job, for example, is much more complex than reading or updating values in the database. It may require not only modifications of data items but invoking other functions as well (Dreibelbis, Lacy-Thompson, 2000).

A *Remote Function Call (RFC) Server* is deployed in our architecture to trigger functions of the ERP system. This RFC Server is a real-time link

between an ERP system and a mobile device. It plays the role of a high level environment in which functions of the ERP system can be invoked remotely in real time (Narayanan, 2002). An RFC Server is tied to a particular ERP system. It encapsulates all specific details and communicates with the low-level API (Application Programming Interface) of that system.

When an RFC Server is provided developers of mobile applications do not need to bother with these low-level APIs to call an ERP function nor with specific implementation languages used by different vendors (ABAP for SAP R/3, PL/SQL for Oracle Financials, Lj4 for infor:COM, etc.). Data entered by a mobile user are taken as parameters for the business function to be called. For example, the user will say that he/she wishes to re-schedule a particular job, and enter the job number, the new delivery date, etc. on his device. These values are given as parameters to the remote function provided by the RFC Server. More precisely, a request with those parameters is generated and sent to a web server. The web server communicates directly with the RFC Server and invokes the appropriate business function. While it is not very difficult to develop suitable front-ends for such tasks, implementation of the remote functions invoking internal functions of the ERP system can be quite complicated.

Another problem is posed by the needed support for distributed ACID (Atomicity, Consistency, Isolation, and Durability) transactions. In distributed multi-user applications sharing objects in real time can lead to resource sharing conflicts (e.g. many users may want to update the same objects at the same time). To avoid such conflicts various locking strategies to preserve the integrity of changes are used. In our framework, so-called optimistic locking (write locking) is applied. Optimistic locking allows unlimited read access to an object but a client can only write an object to the database if the object has not changed since the client last read it (Adya, 1994). If many users have the same data open, only the first update to commit succeeds while the others obtain error messages.

If the mobile device sends a request to an application server to update some data in the database, the application server invokes an appropriate ERP function via the RFC server. If that function cannot be successfully executed, an exception (`OptimisticLockException`) is thrown, propagated further to the RFC server and subsequently to the application server. The application server sends a response to the mobile device informing about the error and the user is asked to refresh the current data and re-enter values. Write-write conflicts in our solution are detected using timestamps indicating the last commit time.

3. AN EXAMPLE OF MOBILIZING A REAL-WORLD ERP SYSTEM

3.1 Restrictions and Design Considerations

Significant problems in mobilizing today's ERP systems are caused by low bandwidths of telecommunication networks. It is time and cost ineffective to send larger portions of data to mobile devices. Processing should better be done on an application server and only the results should be transmitted in a compact form to the mobile device.

Another shortcoming is the low processing power of mobile devices. As a consequence, only simple things like validating input can be done directly on the device while the mobile application logic must remain on an application server. We took these aspects into account when developing a prototypical solution for a real-world ERP system, *infor:COM* by Infor Global Solutions (Infor, 2005). This system aims at small and medium-size enterprises and has a good market penetration in Central Europe.

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 (Kurbel, Teuteberg, Hilker, 2003). 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 technical limitations into account.

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 monitor. Screen content can be quite complex, showing many data and sub-forms at the same time.

Considering the small display sizes of mobile devices, it is obviously not possible to "translate" an existing ERP front-end into a mobile front-end one-to-one. The front-end designer has to concentrate on important information instead and adopt only the really essential data from the standard screens of the desktop-based application. Usually it is necessary to divide the data up into several screens of a mobile device ("cards" in the terminology of mobile browsers).

As an example, processing of quotes in the sales module of infor:COM is described. On a standard desktop client, the user generally sees a large form with many menus, submenus, drop-down-boxes, text fields, etc. It is quite obvious that this type of user interface 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 re-directed to menus detailing the functions of the chosen module. Finally the user reaches a card with the desired functionality.

Figure 2 shows a search for quotes that match some conditions specified by the user as a stepwise process on a mobile device. The front-end was implemented with the help of a Nokia 7210 simulator for mobile devices. Such simulators are available in toolkits provided by devices manufacturers and other vendors (Kurbel, Dabkowski, Zajac, 2002).

Figure 2. Selecting quotes from ERP database

Assuming that the user looks for a particular quote (quote no. "AG1001"), he or she navigates through a sequence of menus to reach the desired card. The functionality outlined in Figure 2 is the same as for the desktop-based infor:COM system. For example, the user can select a certain way of filtering quotes (by quote number, RFQ number, customer number, 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 distributed over several cards. Likewise the details of a quote are also divided into a number of cards.

While the user sees only the front-end displaying ERP data as requested, the requests and responses are transferred across the overall architecture as described in the previous section. Looking 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.

4. OPEN ISSUES AND FURTHER WORK

Industry experts forecast that the market for wireless applications will continuously grow over the next few years. With increasing needs of business users to access information from anywhere at any time, organizations have to find new, more effective system architectures.

This paper focused on building a multi-tier framework for mobilizing an existing ERP system. An alternative approach for a mobile ERP solution could be based on a Web Services architecture (Chappell and Jewell, 2002). Web Services are considered as a major step forward in inter-enterprise cooperation and integration of different types of systems. They can play the role of a universal Application Programming Interface which does not require any other protocol than the Internet protocols. Web Services standardize the calling, exchange and organization of application services. In terms of software development, the increasing complexity of business information systems and the need for rapid adaptation to different devices and for integration of business concepts are good reasons for using a high-level approach which implements techniques of modeling and programming with business objects. In our project reengineering and development work is currently going on. The goal is to implement an effective, Web Services-based architecture for a part of the mobile ERP system.

References

Adya, A., 1994, Transaction Management for Mobile Objects using Optimistic Concurrency Control; http://research.microsoft.com/~adya/pubs/tr.pdf.

Apache, 2005, Clustering/Session Replication How-To; http://jakarta.apache.org/tomcat/tomcat-5.0-doc/cluster-howto.html.

Apache, 2005a, Tomcat Server; http://jakarta.apache.org/tomcat/.

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

Chappell, D. and Jewell, T., 2002, *Java Web Services*, O'Reilly & Associates, Sebastopol.

Dreibelbis, D. and Lacy-Thompson, T., 2000, Interfacing with SAP R3; http://www.eaijournal.com/Article.asp? ArticleID=144&DepartmentId=4.

Gertz, M., 2000, Oracle/SQL Tutorial; http://www.db.cs.ucdavis.edu/teaching/sqltutorial.

Hemphill, D. and White, J., 2002, *Java 2 Micro Edition*, Manning Publications, Greenwich.

Hori, M. et al., 2000, Annotation-Based Web Content Transcoding, in: Proceedings of the 9th International World Wide Web Conference on Computer Networks, Amsterdam, pp. 197-211.

Hwang, Y., Seo, E., Kim, J., 2003, Structure-Aware Web Transcoding for Mobile Devices, in: *IEEE Internet Computing* 7 (5): 14-21.

IBM, 2002, IBM: Voice Server SDK; http://www-3.ibm.com/software/voice/.

IBM, 2003, VoiceXML Programmer's Guide; http://www.elink.ibmlink.ibm.com/public/applications/publications/cgi bin/pbi.cgi.

Infor Global Solutions, 2005, infor:COM; http://www.infor.de.

Kurbel, K., Dabkowski, A. and Zajac, P., 2002, 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, pp. 673-676.

Kurbel, K., Teuteberg, F. and Hilker, J., 2003, Mobile Business-Anwendungen im Enterprise Resource Planning: Mobilitätspotentiale entlang der ERP-Funktionskreise, *Industrie Management* 19 (1): 72-75.

Myerson, J. M., 2002, *The Complete Book of Middleware*, Auerbach Publishers, Philadelphia.

Narayanan, V., 2002, Interfacing with SAP R/3; http://www.info-sun.com/docs/wp_sapinter.pdf.

Opera, 2005, Opera Products for Mobile; http://www.opera.com/products/mobile/.

Paragon Corporation, 2003, Separation of Business Logic from Presentation Logic in Web Applications; http://www.paragoncorporation.com/ArticleDetail.aspx?ArticleID=21.

Pfeifer, C., 2001, XML Processing with TraX; http://www.onjava.com/pub/a/onjava/2001/07/02/trax.html.

Riggs, R., Taivalsaari, A. and VandenBrink, M., 2001, Programming Wireless Devices with the Java 2 Platform Micro Edition, Addison-Wesley, Boston.

SALT Forum, 2002, Speech Application Language Tags 1.0 Specification; http://www.saltforum.org/saltforum/ downloads/SALT1.0.pdf.

Sun Microsystems, 1997, JavaBeans; http://java.sun.com/products/javabeans/docs/spec.html.

Sun Microsystems, 2002, JavaServer Pages Specification, Version 2.0; http://jcp.org/aboutJava/communityprocess/first/ jsr152/.

Vaidyanathan, R., 2001, Wireless Application Integration. EAI Journal; http://www.eaijournal.com/Article.asp? ArticleID=450&DepartmentId=3.

W3C, 1999, HTML 4.01 Specification. 1999; http://www.w3.org/TR/html4/.

W3C, 2000, Composite Capabilities/Preference Profiles: Terminology and Abbreviations, Working Draft; http://www.w3.org/TR/2000/WD-CCPP-ta-20000721/.

W3C, 2002, Delivery Context Overview for Device Independence; http://www.w3c.org/2001/di/public/dco.

W3C, 2003, Voice Extensible Markup Language (VoiceXML) Version 2.0; http://www.w3.org/TR/voicexml20/.

W3C, 2003a, XHTML 2.0 The Extensible HyperText Markup Language Specification; http://www.w3.org/TR/xhtml2/.

W3C, 2003b, XSL Transformations (XSLT) Version 2.0; http://www.w3.org/TR/xslt20.

W3C, 2004, Extensible Markup Language (XML) 1.0 (Third Edition); http://www.w3.org/TR/REC-xml.

WAP Forum, 2001, UAProf; http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20010530-p.pdf.

WAP Forum, 2002, Wireless Application Protocol WAP 2.0, Technical White Paper; http://www.wapforum.org/what/ WAPWhite_Paper1.pdf.

Weiser, M., 1991, The Computer for the 21st Century, *Scientific American* 265 (3): 94-104.

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