MOBILE DISTRIBUTED SOLUTION FOR DELIVERY AUTOMATION: DESIGN AND PARTICULARITIES

Anita Lungu

XTS Software
Horia Str, I13, Sc1, Ap1,Craiova, Romania
anita@xts.ro

Abstract: The system presented is a mobile route management solution built for AldiPress, one of the leading Dutch companies involved in the distribution of magazines to the retail business. EcoSys provides routing itineraries and delivery related information to approximately 150 drivers that carry out the actual distribution throughout the Netherlands. The existent solution was paper based, thus tedious, time consuming and prone to human error. The paper describes the specific issues related to designing and implementing a mobile application that solves the above mentioned problems. The challenges on building this application are related, mostly to the limited resources available on a Symbol SPT 1800 (PalmOS) terminal. Various optimizations are required to overcome those difficulties and maintain the speed compulsory for a mobile application.

Keywords: Mobile, PDA, PalmOS, Scanning devices

INTRODUCTION

A short overview of the implemented system is provided for a better understanding of the overflow. AldiPress' distribution centres are situated in Duiven and Amsterdam. The Duiven center is responsible with the distribution of the magazines to the various transport companies involved in the actual deliveries, while the center in Amsterdam is equipped to handle all the returns (magazines that are not sold).

AldiPress distributes the magazines in bulk transports to 15 transport centers. Each retrieves the routes assigned to it through the Internet. This information (routes, delivery addresses and specific barcodes) is made available from within the prior existent AldiPress ERP system (SAP). This ERP is the "brain" that organizes and establishes the activity for each day.

DESIGN CONSIDERATIONS

The application had to transmit the information back and forth between the ERP (SAP), distribution center and mobile terminals. Two different applications were built, one to reside on the desktop computers of the distribution centers (EcoLite for Windows 2000) and the other on the mobile terminals (EcoSys for PalmOS). EcoLite was installed on approximately 10 desktop computers belonging to the distribution centers while EcoSys was installed on approximately 150 mobile PalmOS terminals.

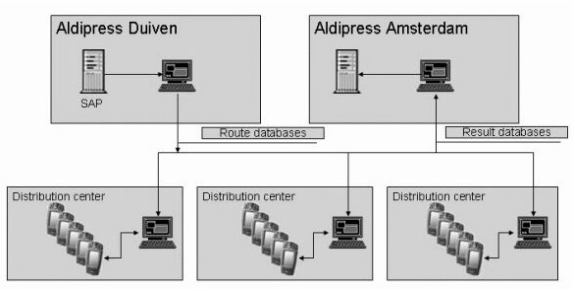

Fig.1. Main System Components

The distribution centers thus, run EcoLite which allows the center operators to view and modify (to

some extent) the received routes, as well as to upload routes to mobile terminals.

Display: High contrast, anti-reflective 160 x 160 monochrome LCD display
CPU: DragonBall VZ – 33 MHz
Operating System: Palm OS® 4.X
Memory:16 MB RAM / 4MB ROM
Application Development: FalchNet

Fig. 2. Symbol SPT 1800 Mobile Terminal

Each driver is equipped with a Symbol SPT 1800 mobile terminal. This barcode scanning capable PDA runs the EcoSys PalmOS mobile application throughout the route. At the beginning of the day, each driver inserts his terminal in a cradle, and the EcoLite operators in that center load a specific route onto it. At the end of the day, after finishing the route, the terminal is again put into cradle for synchronization purposes. The route processing information is retrieved by EcoLite, and later on sent to the ERP in Amsterdam.

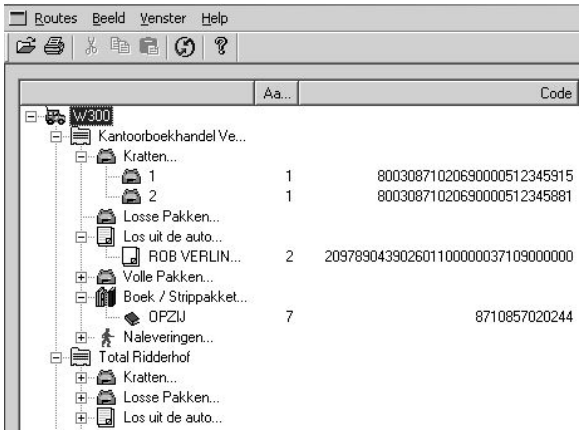

Fig.3. Desktop EcoLite Application - PrintScreen

TECHNICAL CONSIDERATIONS

The PDA will receive the data from EcoLite as a file in pdb (Palm Data Base) format then will launch the EcoSys application. Through it, all the information needed (such as the clients addresses, the deliveries for each customer) will be made available to the user. Identification of the products within this system is accomplished through a barcode label scanned by the Symbol terminals.

Operating System and Programming Language

The applications running on a Palm PDA generally have one thread of execution and are event driven. They run in turn, a user does not terminate (or exit) an application he only selects a different one to run. As a response to this action, the OS terminates the currently running application and launches the new selected one. Each application (.exe, .dll) is identified by the Operating System through a Creator ID that represents a signature for an application. An Official, unique list of this Creator IDs is maintained for reference.

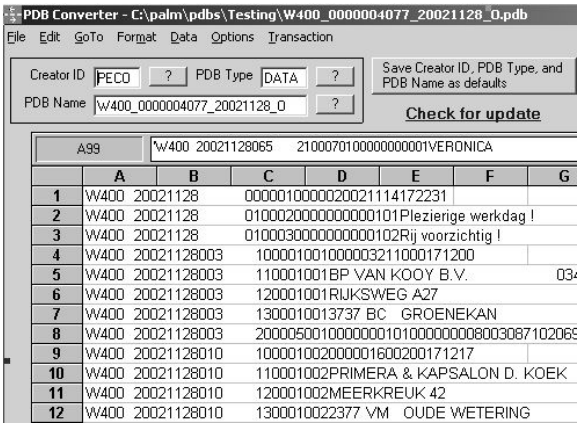

Fig.4. Information maintained inside a Palm database

The environments available for Palm development include C++, VB, Java as programming languages. The EcoSys was written using FalchNet Developer Studio for Palm OS. The programming language was C++ using a proprietary C API functions.

Database Particularities

Unlike the majority operating systems, the PalmOS has no file system. All data and programs are stored on PDA in generic "databases" (the term having no connection to a relational database model). Such a Palm database contains a variety of information in records but this information is useless unless manipulated by an application. Each database on PDA has a unique name. For example, the opening of such a database is done in the following way:

```
Boolean CRouteData::OpenDB(Char* szDBName){
        LocalID locID =
DmFindDatabase(CARDNR,szDBName);
        m_dbRef =
DmOpenDatabase(CARDNR,locID,dmModeReadWr
ite);
}
```

Working with a Palm database involves quite a low level programming as can be seen from the following example. A locking is executed on the record level in order to grant the unique access to a record. The next step obtains a pointer to the memory where the record is stored and the length in bytes. The information is copied in a buffer as an array of bytes from which the required data is extracted according to the predefined format of each record. In order to retrieve a record of information from a database, the following code should be executed:

```
Char* CRouteData::GetRecordByID(UInt32
id,DmOpenRef ref){
    Char* ret;
    UInt16 index;

    DmFindRecordByID(ref,id,&index);

    MemHandle hRec = DmGetRecord(ref,index);
//returns a  handle to a record

    Char* pRec = (Char*)MemHandleLock(hRec);
//lock handle, returns a  pointer to a  record

    UInt32 size = StrLen(pRec);
        ret = (Char*)MemPtrNew(size+1);
        StrCopy(ret,pRec);
        MemHandleUnlock(hRec);//release  handle
        DmReleaseRecord(ref,index,false);
        return ret;
}
```

Specific Optimizations

One of the specific requirements from the application regards data corruption. The information should not be corrupted in case a battery failure or a hard reset appears during normal operation. Moreover, should such events occur, on restart the user has to be repositioned in the same place of the route processing (the same customer and product delivery). To adjust these requirements, a continuous concord had to me maintained between the user's operations and the Palm database.

Considering the non-relational model of the database and the keeping of records as arrays of bytes, EcoSys has to implement functionalities specific to a database management system. The complexity of the application is thus increased while the requirements, in terms of response time to the user, have to be also accommodated.

In order to accomplish this, a preprocessing of the pdb takes place upon starting the application. The entire database is parsed byte by byte and an indexing structure (global vectors) is stored in RAM. These indexes offer information on the starting position of every customer related data as well as starting positions in memory for the deliveries of each client. By keeping all this information in RAM, it is significantly easier to access the needed information. The memory is not parsed every time in order to find a specific record, instead the exact location is addressed.

By choosing this option, part of the computational effort is done at the beginning of the application making a lot faster accessing the data during the current workflow. The time response of a user for accessing needed data or going through different screens is always under a second, which was considered acceptable.

To prevent the lack of synchronization between the user's actions and the database which could potentially lead to database corruption, the status of the route processing is continuously maintained in the database. The route can have the status as unprocessed, currently being processed or completed. Upon restart, the database is parsed and the index vectors rebuilt. Afterwards the user is repositioned on the exact position in completing the route as before the failure.

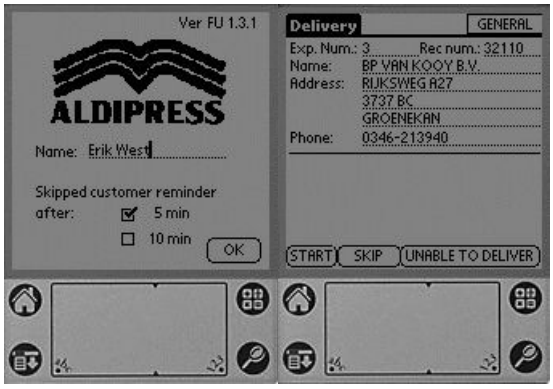

Fig.5. Applications Screens

As workflow, the application has 22 screens. The passing between them having as top difficulty optimizing the database accesses considering on one hand the continuous synchronization between the users actions and the database content and on the other hand how time consuming database accesses are.

Another specific requirement was for the application to be able to work in different ways while maintaining mostly the same workflow. One version of the application was to be used only for training purposes another for a simplified workflow. In order to achieve that without keeping different versions of the software, part of the code was written using precompiler directives (#ifdef) and each version is obtained from the same code by setting the corresponding directives and then recompiling.

The normal mode functioning of the application identifies a package solely by scanning. This was chosen so as to minimize the risk of human error: in case the scanned code does not belong to the current order or client, the user is warned and the operation can be restarted.

Still, in case the scanner malfunctions, a "manual" mode of working is provided for the application. The barcode data is manually imputed by the user. This specific working mode is protected by a password.

The application was developed in the Windows Emulator and after reaching a stable version it was tested on the real PDA device. In order to avoid making some copies of the data structures all the database interface functions worked through using pointers to structures. Considerring the limited memory available (RAM), special effort went into making sure there are no memory leaks. In order to ensure this, the Windows emulator testing mode was used. This specific mode randomly generates button click events (Gremlins). Using this technique, the application is put in an infinite loop of testing and the memory allocation and deallocation is continuously checked.

Because of the small screen (160X160 pixels), the interface design had to be done carefully. An application was built that that correlates the relatively complex interaction with the user (a lot of information has to be displayed, input taken from the user) to the small display surface.

A scanning library was developed for the scanning functionality. It was based on API level barcode functions from Symbol technologies.

Mobile Application Features Overview

- ✓ *Instant overview.*

  At first glance the driver is provided with an inventory of the customers to visit and the items to be delivered on that route.

- ✓ *Ease of use.*

  During the delivery all information and control resides within the mobile terminal. The driver has only to follow the workflow indicated by the terminal. To help in accommodating more unusual customer requests, special instructions can be transmitted to the driver by means of customer originated notes that are displayed on the PDA prior to that particular delivery. In case a delivery can not be completed the driver can select the reason with a simple screen tap. If needed, it can also add a text note describing the problem.

- ✓ *Flexible and accurate workflow*.

  EcoSys has support for on-the-spot delivery address reordering, providing flexibility in processing the route. In case the Symbol scanner is damaged during a delivery, the application can still function in a password protected "manual mode". Delivery of each item is confirmed through scanning its barcode and matching it against the local database of barcodes. In case of mismatch (eg. the scanned barcode pertains to another delivery) the driver is warned and the scanning can be repeated. Each transaction is time stamped for future reference.

Mobile Application Benefits Overview

- ✓ *Reducing Errors.*

  EcoSys' workflow provides fast and errorless data collection using barcode scanning. It replaces the paperwork with an exchange of digital information thus eliminating end-of-day manual data entry. This reduces human errors that most often translate into dissatisfied customers, low productivity and diminished revenue.

- ✓ *Increasing system efficiency*.

  EcoSys addresses some of the problems and slow downs existent prior to the introduction of the digital mobile system. The delivery process was costly in terms of time and prone to errors. It generated large amounts of paperwork, that had to be processed later on by company staff as deliveries and returns were verified. There were inevitable delays in using order and inventory information, affecting business forecasting, inventory control and company cash flow. EcoSys provides a reliable and cost-effective solution that keeps mobile workers well informed while in the field. It permits them to efficiently access and store information about each customer and their respective deliveries. The information resulted from a route processing is already in digital format, hence easily uploaded to the main SAP system for further analysis.

- ✓ *Accurate circulation numbers.*

  AldiPress delivery workers visit roughly the same outlets at regular intervals. EcoSys efficiently and effortlessly registers the circulation numbers in terms of copies distributed, empty crates intake, location of unsold magazines. By replacing manual paperwork with simple scanning and screen tapping, at the end of a route a complete and accurate report is already available on the terminal. These reports are sent back to the SAP system of AldiPress in Amsterdam. This data, when further analyzed, has a great business value, allowing AldiPress to reduce costs by properly allocating copies amongst locations, and allowing for targeted advertising actions.

REFERENCES

Neil Rhodes, Julie McKeehan (1998) *Palm Programming: The Developer's Guide*