Reconfigurable consumer direct logistics systems

Sébastien Truchat[1], Alexander Pflaum[2]

[1]Department of Computer Science "Computer Networks and Communication Systems"
University of Erlangen-Nuremberg
Martensstr. 3
D-91058 Erlangen, Germany
Sebastien.Truchat@informatik.uni-erlangen.de

[2]Fraunhofer Applications Centre for Transport Logistics
and Communication Technologies
Nordostpark 93
D-90411 Nürnberg, Germany
pflaum@atl.fhg.de

Abstract: The goals of the interdisciplinary project HORN were as well to improve competitiveness of service providers that deliver grocery items as to develop new hardware and software needed for its realisation. In the first part, we introduce this home replenishment project, and analyse the needs and challenges of mobile services with similar constraints. In the second part, we introduce our modular software engineering approach and especially present the adopted reconfiguration mechanism for autonomous mobile applications, and its profiling concept.

1 Introduction

Today's world of mobile communication and pervasive computing is still characterised by a bright variety of technical IT devices [Bu01]. Getting an overview of available software engineering possibilities and hardware platforms is hard. Many problems would be technically solvable for particular systems, but when it comes to make these different solutions work together, i.e. interoperate, the lack of clearly defined and widespread standards makes it impossible to implement durable real platform-independent solutions.

A strategy has to be developed, that can be used continuously from the design phase to implementation. This kind of system would permit rapid software prototyping to obtain an open interfaced reconfigurable mobile device. In that same way, the infrastructure installed for mobile services would be dynamically reusable for new services. The idea is to take a home and office replenishment project, and to use it as basic system to analyse the needs for interoperability.

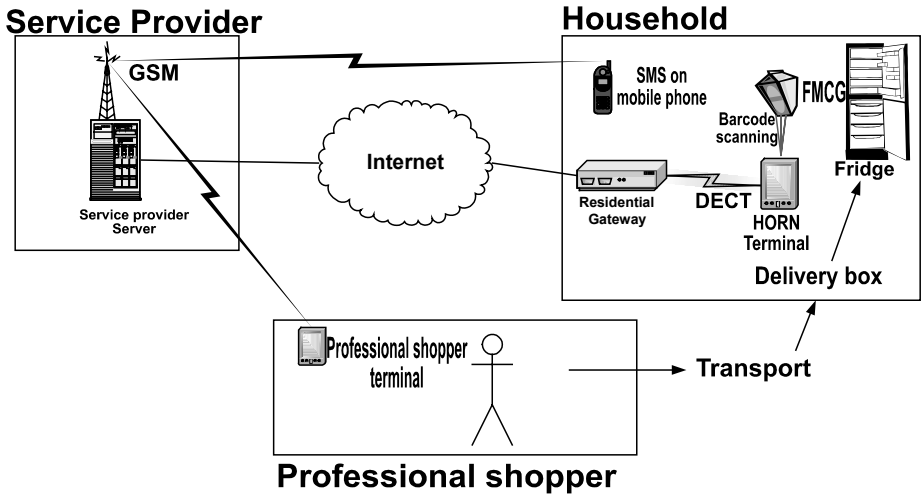

Fig.1: HORN infrastructure

2 HORN: Home and Office Replenishment Nuremberg

The Home and Office Replenishment service in Nuremberg (Figure 1) supplies consumers with dry packaged and fast moving consumer goods (FMCG). The consumer buys his stock of products to be replenished once at the beginning of the service process and defines maximum and minimum stock as well as the medium range for each product using a Pocket PC that is equipped with an identification module (barcode scanner) and a module for wireless communication (DECT) that provides the connection to the service provider's data base via a residential gateway that is connected to the internet through the standard public telephone network. After consumption of a product, the EAN article number that is printed on the product package is scanned and a message is generated automatically to be sent to the service provider using absolutely secure communication lines. Here the consumption messages are gathered and stored in a data base. Once each day, a special decision support software algorithm that takes various system and customer specific restrictions into account, checks the data base, and generates picking & packing orders and shopping lists that are sent to a professional shopper who is also equipped with a wireless pocket PC with integrated GSM module and barcode scanner. The shopping is done, products are packed into transport boxes, carried to the consumer and put into delivery boxes for unattended delivery. The consumer is supported with an SMS on his mobile phone that contains dispatch information. An electronic bill of delivery is sent to his wireless Pocket PC. After taking out the items, the delivery is accepted or not, an acknowledgement of receipt is sent back to the service provider who then initiates the invoice. The conceptual idea from an economical and logistical point of view can be found under [Pf03].

2.1 An infrastructure for mobile autonomous services

Why do we need this kind of infrastructure? Commonly, mobile services use the internet model [Ha01]: the application runs on an internet server, while the mobile client has an access to this service through a standard browser. This client browser is used as a visual human machine interface, while data are actually processed on the server side. One obvious advantage of this architecture, from the point of view of software engineering and software maintenance, is that the software only needs to be implemented for the server platform. Processing power restrictions are to be considered on the server side. Mobile clients only need a browser to use the service. One drawback of this strategy is that there must be a browser implementation for every new mobile platform, and the system must agree with the restrictions of this browser, but the major problem is that the service accessibility depends on the communication to the server: it must be fast enough, reliable, and always available. As the consumer wants to be mobile everywhere in his house, even e.g. in the cellar where there is no reception of any kind, to manage his beverage stock for example, the mobile service has to be an autonomous application on the mobile terminal. This is one essential characteristic why HORN should be considered as an m-commerce initiative rather than e-commerce: most of the transactions are made by the customer in an offline situation [Zo01]. Furthermore, the price of the communication to the internet can be restrictive, and battery life is reduced by a higher power consumption of the communication interface that has to be powered during the time of connection to the server. So the idea is to use local applications and short range wireless networks.

Some approaches, not constrained with local applications, combine the advantages of a server based solution with short range charge-free wireless communication such as the Flower framework [HLS02]. Some others combine server based services with some client specific software pieces using existing WSDL standards [HMM04].

The "HORN architecture" would reveal expensive for the operators of the residential gateways and servers, and as a consequence for the consumer, if it could only be used for this one single service, so the goal of our research is to enhance the reusability and interoperability of existing infrastructures of that kind. In our research infrastructure, the mobile device has no direct contact to the service provider but can start a communication to a residential gateway. Our "residential gateway" can rather be considered as an association of a local server that stores data for the mobile terminal and a residential gateway to access the internet on demand (no permanent connection to the internet). Since the mobile device has only sporadically a connection to a (local) server, the applications must run directly on it. So the "service provider", who offers applications for the mobile clients, must implement software modules for every spectrum of devices on which he wants his application to run. Moreover, he may want to update some of these modules sometimes. Once the infrastructure is standing, and when there is a possibility to update or reconfigure the mobile devices, there might be other service providers who could want to use this infrastructure to offer their services. Thus our goal is to find a way to reconfigure most mobile devices in that infrastructure, independently of their operating system, and to improve the software development process for such mobile applications.

The introduction of new mobile services could be made more efficient by dedicated business models, as shown by the example of "B4U" [HR03]. Our approach would like to design a model for interoperability in order to allow the operators of existing infrastructures to offer the use of these to new service providers.

3 Modular software-engineering for interoperative systems

The goal of the Mo.S.I.S. (Modular Software-engineering for Interoperative Systems) project is to make the development of mobile services in general, and especially for HORN-like architectures and systems with few resources, more cost efficient [Tr04]. First of all, there is a need for a generic reconfiguration framework for the mobile devices and residential gateways (i.e. local servers), since there does not exist a standard for software deployment that works for every combination of hardware and operating system platform, especially very lightweight devices. To achieve this, an ontology to relate the profiles from devices to software modules with regard to the description of application profiles has been developed. These reconfiguration rules can be considered as the necessary clear invariants that govern the entire system (known as volatility principle [KF02]). A middleware solution may be the best approach to achieve interoperability between software components [GB03], and for exchanging components or downloading new ones. In the case of Mo.S.I.S., the presence of such a middleware on every mobile device is not assumed: the fast implementation of services through design patterns is being considered less time consuming than the implementation of a complete middleware for every new device, since devices are considered to have a "short" life. Besides, not every mobile device may have enough resources to host a mighty middleware.

3.1 Reconfiguration and communication

For the first tests, the local server part is played by a standard PC connected to a WLAN access point. In the future, this part might be embedded in the residential gateway platform, which assumes of course much less memory resources and computing power. In the following section we use indifferently "local server" or "residential gateway". The mobile terminal can try to connect to any accessible local server over WLAN on request of the user. This approach has been preferred to a periodical scanning in order to save battery life, though it assumes the user must be aware of the presence of a potential local server. The reconfiguration process can be described as follows (figure 2): 1) the user asks the mobile terminal to try to connect to some local server. 2) if successfully connected, the mobile terminal sends its profiles to the local server. 3) the local server matches these profiles with those of available software modules in its repository. 4) as soon as there is a module that matches, it is offered for download. 5) if the user acknowledges, the module can be downloaded. 6) the module is being downloaded. 7) the connection is being terminated. 8) after successful download, the software profiles of the mobile terminal can be updated according to the new modules.

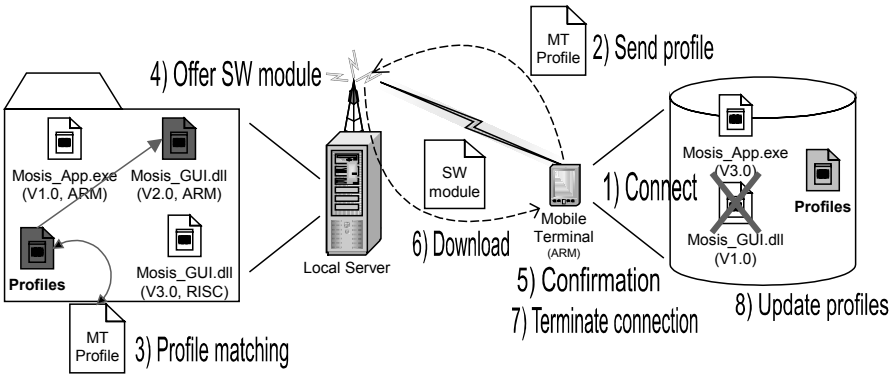

Fig.2: Reconfiguration process of the first prototype

The core of this reconfiguration concept is implemented in two "reconfiguration manager" classes (Figure 3): CRecoMaLS (stands for reconfiguration manager on the local server side) and CRecoMaMT (reconfiguration manager on the mobile terminal side). Basically, the reconfiguration mechanism and also diverse data exchange mechanisms rely on RPC (remote procedure call).

To make sure that this mechanism can be used even on lightweight mobile platforms, a special simple and extensible XML based RPC protocol has been designed:

```
<MOSIS_V0.1>
   <Befehl> order </Befehl>
   <Empfaenger> addressee </Empfaenger>
   <Laenge> length in byte </Laenge>
   <Daten> data </Daten>
</MOSIS_V0.1>
```

In the basic version, *order* can be "Daten_senden" in the case of a simple data exchange between two applications, or "File_senden" in the case of a file (e.g. software module) transfer. When *order* is "Daten_senden", the *addressee* is the function that has to process the sent data. When *order* is "File_senden", the *addressee* is the memory place where the file has to be stored (e.g. path and filename when the OS allows it). The *length in byte* of the transmitted *data* is an important information, when the communication is not interrupt driven, so the raw data has to be parsed and filtered.

The essential idea in XML based RPC frameworks is to use XML to define a type system that can be used to communicate data between clients and servers [MS03]. In this case, it is useful to transmit information such as the length of transmitted data, or a checksum (to verify that the software module has been transmitted uncorrupted) independent of the receiver platform, as long as it possesses an elementary XML parser.

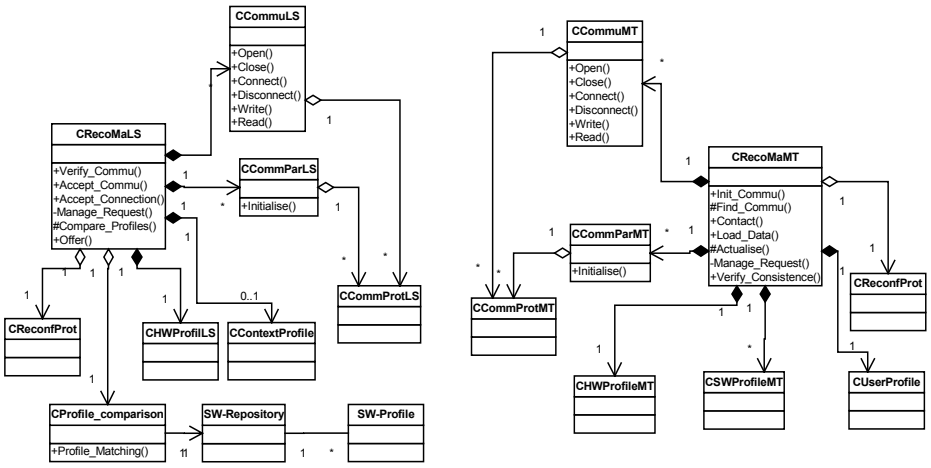

Fig.3: Class diagram of the reconfiguration part on the local server and mobile terminal side

As communication mediums can be very diverse from one device to another, a special attention has been spent to the development of generic (or reusable) communication classes to shorten development time even on this side. The interface of these classes keeps the same, but there are mainly two kinds of implementations to cover most of the current communication devices: a socket based communication (if the appropriate API exists in an appropriate programming language for the communication device and the OS used on the mobile terminal), and a modem based communication. Concerning this latter, lots of devices can be controlled through a (virtual or not) serial port interface, that allows to communicate directly with the device through its AT protocol. This technique becomes more interesting, when the communication device itself is still a prototype (as it was the case during the HORN development phase) the firmware of which may change, so that existing modem based APIs do not always work. In our model, parameters like the AT protocol of the device can be encapsulated in a dedicated communication-parameter-class (CCommProtMT and CCommProtLS) so that the developer does not need to change the code of the communication class when something changes in the AT protocol (e.g. the device identification answer changes, which is needed to detect the presence of a specific communication device).

3.2 Profiling

Our reconfiguration environment has the following features: there are diverse mobile terminal platforms with potentially several users (technically skilled or not), there are several local server platforms, and it should work with any combination of user, mobile terminal, and local server, without assuming deep technical knowledge and complicated operations by the user. Besides, it must work and be quickly implementable for (almost) any new platform. To make this "any-to-any reconfiguration" possible, we need some rules to classify hardware platforms, to structure the software modules of applications, and finally to set them all in relation. This "reconfiguration ontology" is a kind of profiling, which is most often used for content adaptation problems.

In a nutshell, the mobile terminal has a hardware profile, a software-library profile, and a user profile. The hardware profile contains information like processor type, operating system, memory, display type and communication means. The software-library profile consists of the profiles of the applications, and the profiles of the software modules (that are already on the mobile terminal). The application profile contains the name of the application, its version, and the names of all the software modules that compose this application. A module profile contains information like name, version, processor type for which it has been compiled, memory needed and so on. The user profile contains preferences such as the language (e.g. German or English).

Context awareness can be achieved when the local server first transmits its context profile to the mobile terminal, so that this last one can make context dependant decisions such as if it wants to transmit its profiles to the local server or not, or if it wants to activate some tasks in this context (not implemented yet).

The rules are as follows. Is there any application profile on the local server that is not present on the mobile terminal? If yes, the system checks if all the needed software modules composing this application are present on the local server, and if these modules fit to the mobile terminal. If yes, these modules including their profiles and application profile are offered to be downloaded for the mobile terminal. If a newer version of an existing application profile is available, the procedure is the same but only the necessary modules are proposed to be downloaded. For every module of the mobile terminal, the system checks if there is a newer version of this module available on the local server. If yes, this newer version is offered for download. A software module fits to a mobile terminal when it has been compiled for the right type of processor, for the right operating system, if memory resources are sufficient, the display matches to the GUI and so on. To keep the system flexible and extensible, the rules and profiles must be capable of being modified with little inconvenience for the developer (better software maintenance).

The first profile matching prototype has been implemented in the frame of a Diplomarbeit [Fu04] in Erlangen. The selected solution was a symmetric attribute based matching with external rules. That means the vocabulary used is the same for all the profiles, and the rules that explain how properties have to meet requests are edited separately. The language chosen for the implementation of the profiles was CC/PP (composite capability / preference profiles) because it seemed to be the most used standard.

4 Conclusions and further work

The result of our work should be a kind of design process handbook to make the development of new mobile services more cost efficient by helping to choose the right infrastructure, by reusing existing infrastructures with new services and devices through interoperability and reconfiguration, and by shortening development time thanks to design patterns.

As for the evolution of home replenishment, it seems an evidence that scanning manually the consumer goods is a fastidious task, but it was the best way to realize the HORN pilot project yet. The future in this domain belongs obviously to "smart labels", so that a sensor in the fridge or pantry could notice the input and output of goods in order to make the process really pervasive. The mobile device would only serve to manage and supervise the desired stock of goods, and to use context specific services and information.

Literaturverzeichnis

[Bu01] Jochen Burkhardt, Horst Henn, Stefan Hepper, Klaus Rindtorff, Thomas Schäck , "Pervasive Computing, Technologie und Architektur mobiler Internetanwendungen", Addison-Wesley 2001, 241-264.

[Fu04] Gerhard Fuchs, "Mobile autonome Dienste und ihr Profiling", Diplomarbeit am Lehrstuhl Informatik 7, Friedrich-Alexander Universität Erlangen-Nürnberg, July 2004.

[GB03] Paul Grace, Gordon S. Blair, "Interoperating with Heterogeneous Mobile Services", ERCIM News Number 54, July 2003, 24-25.

[Ha01] Uwe Hansmann, Lothar Merk, Martin S.Nicklous, Thomas Stober, "Pervasive Computing Handbook", Springer 2001, 327-340.

[HLS02] Tero Hakkarainen, Ali Lattunen, Vespe Savikko, "Flower - Framework for Local Wireless Services", ERCIM News Number 50, July 2002, 51-52.

[HMM04] Markus Hillenbrand, Paul Müller und Kristian Mihajloski, "A Software Deployment Service for Autonomous Computing Environments", International Conference on Intelligent Agents, Web Technology and Internet Commerce, July 2004.

[HR03] Timber Haaker, Oscar Rietkerk, "Introducing New Mobile Services Faster", ERCIM News Number 54, July 2003, 44-45.

[KF02] Tim Kindberg, Armando Fox , "System software for ubiquitous computing", IEEE Pervasive Computing, January-March 2002, 70-81.

[MS03] Friedemann Mattern, Peter Sturm, "From Distributed Systems to Ubiquitous Computing - The State of the Art, Trends, and Prospects of Future Networked Systems", in: Klaus Irmscher, Klaus-Peter Fähnrich (Ed.): Proc. KIVS 2003, pp. 3-25, Springer-Verlag, February 2003.

[Pf03] Alexander Pflaum, "Die Zukunft des "E-Fullfillment" für Lebensmittel: Versuch einer Prognose", Logistik Management, 5.Jahrgang 2003, Ausgabe 1, 25-39.

[Tr04] Sébastien Truchat, "Interoperative Systems for Replenishment" (doctoral colloquium of the Pervasive 2004 conference, April 2004) In: Alois Ferscha, Horst Hörtner, Gabriele Kotsis (eds.) : Advances in Pervasive Computing. Österreichische Computer Gesellschaft. 161-166.

[Zo01] Jörg Zobel, "Mobile Business und M-Commerce", Hanser, 2001, 3-4.