ID-Services: An RFID Middleware Architecture for Mobile Applications

Joachim Schwieren and Gottfried Vossen

European Research Center for Information Systems (ERCIS)
University of Münster, Leonardo-Campus 3, 48149 Münster, Germany

Abstract. The use of RFID middleware to support application development for and integration of RFID hardware into information systems has become quite common in RFID applications where reader devices remain stationary, which currently represents the largest part of all RFID applications in use. Another field for applying RFID technology offering a huge set of novel possibilities and applications are mobile applications, where readers are no longer fixed. In order to address the specific issues of mobile RFID-enabled applications and to support developers in rapid application development, we present the architecture of an RFID middleware for mobile applications. The approach has been used to implement *MoVIS* (Mobile Visitor Information System), a mobile application which allows museum visitors to request individually adapted multimedia information about exhibits in an intuitive way.

1 Introduction

Many information systems are being used to manage objects or processes from the "real" physical world (e.g., products, shipping goods or other physical resources). The gap that exists between the physical world and its domain-specific model in the information system which is also referred to as "physical digital divide" [13], needs to be bridged by manual user interaction. Apart from that, auto-ID technologies like barcodes, smart-cards and RFID (Radio Frequency Identification) can help to integrate information systems very closely with the physical world and to align them with physical processes. This reduces manual interaction and helps to increase reliability, to speed up processes and to reduce costs. Especially RFID has emerged tremendously in the last years, mainly due to lower prices and a higher level of maturity of the available technology [8]. However, the actual success of RFID is based on its superior characteristics: it allows a contact-less and robust identification without requiring a line of sight. This works from a few centimeters up to several meters, depending on the specific RFID technology used. These characteristics are offered by no other auto-ID technology. Apart from very specialized, highly integrated applications (e.g., car immobilizers, contactless micro-payment systems, access control systems) where RFID is already widely established, most effort is currently put on implementing RFID in business contexts (e.g., supply chain automation, tracking and tracing of shipping goods, etc.) [2]. But also industry can benefit from RFID technology in

Schwieren J. and Vossen G. (2008).
ID-Services: An RFID Middleware Architecture for Mobile Applications.
In *Proceedings of the 2nd International Workshop on RFID Technology - Concepts, Applications, Challenges*, pages 19-31
DOI: 10.5220/0001728500190031

order to extend the level of automation in production and manufacturing processes [10]. Even though these RFID applications originate from completely different fields, most applications have in common that the reader device is stationary whereas the transponders are mobile. In this paper we present a framework for RFID application development for the opposite case of mobile readers that opens a new branch of RFID uses which we refer to as "mobile RFID".

Indeed, a rather novel field for applying RFID technology is mobile applications where the RFID reader device itself is a mobile unit (usually combined with a mobile computing device) [19, 20]. Since today's mobile computing devices such as PDAs (Personal Digital Assistant), UMPCs (Ultra Mobile PC) or mobile phones have a relatively strong computing power and offer many functionalities, the combination with RFID offers a huge set of new possibilities and applications. Because mobile applications usually have a higher affinity to the physical world in which they are used, the need for a close integration into the physical world is very high. RFID can be used to directly interact with physical objects which are equipped with transponders. This can be used to reduce the complexity of user interfaces and can help to speed up and align processes. Examples for mobile RFID applications are for example maintenance applications [18] where objects like e.g. fire extinguishers need to be maintained from time to time. However, also field force automation applications that help field staff to quickly access information about product samples they carry with them or keeping track of field service activities are examples of mobile RFID applications. Also in the healthcare and medical context, mobile RFID applications can be used to identify patients or medicine to prevent errors in treatment [4] and to speed up documentation processes [27]. Another application is tracking and documenting sentry patrols at fixed physical locations. Finally, a more consumer-oriented use case is mobile guide systems like *MoVIS* [24], which allows museum visitors to request information about exhibits in a very intuitive way by simply touching RFID transponders that are attached to exhibits. And yet another possible application of mobile RFID is using PDAs for in-field e-learning applications that for example teach engineers about the different components of a complex machine while standing in front of it.

Though there are many possible use cases for mobile RFID applications, these applications are still not very widely spread. A major reason for this is the fact that developing such applications is not a trivial task: The available hardware is very heterogeneous and many different standards and technologies exist. Since the integration of the specific RFID hardware is in the main focus, the development is often very hardware-oriented. This means that developers have to take care of both, integrating the hardware, modeling the physical context and implementing the application's business logic. While developing *MoVIS*, we have encountered several challenges and general requirements that can also be applied to other mobile RFID applications like identifier resolution, information retrieval on physical objects or tracking physical interactions. Therefore we have developed an architecture of an RFID middleware for mobile applications called *ID-Services*. *ID-Services* has been used to implement a production version of *MoVIS* after a first prototype version has been developed to evaluate the general concept of this application.

The remainder of this paper is organized as follows: In Section 2 we cover related work and derive the motivation for our approach from that. We then present the *ID-*

Services architecture in Section 3. Section 4 discusses a particular use case for *ID-Services*, and Section 5 concludes the paper.

2 Related Work and Motivation

As stated before, there are many possible use cases for mobile RFID applications. However, it is difficult to develop such applications since the integration of the RFID hardware often requires a very low-level interaction. Also, the developer has to model the physical context that bridges the physical world with an information system. In the context of stationary RFID applications a special component in the system's architecture is used which is often referred to as *RFID middleware* [11]. In contrast to traditional middleware (like for example CORBA[1]), this *RFID middleware* is seen as a mediator between the RFID reader hardware and the underlying information systems. There is no exact definition of what an *RFID middleware* is and where it is located in an application's architecture [5]; some RFID reader devices even include a middleware in the reader's embedded controller unit.

The idea of using a special component within a system's architecture to mediate between RFID hardware and other components is very common. The term *RFID middleware* has been used to describe such a component. All major DBMS and ERP vendors provide products that they refer to as *RFID Middleware* [17]. Even though this term is frequently used in the literature and also in commercial products, there is no exact definition of what an RFID middleware is and what exact functionality it is expected to have [5]. Furthermore the term *RFID middleware* is almost exclusively used in the context of RFID applications where the RFID reader units are stationary. Most attention is put on typical problems of these applications such as handling large amounts of data or detecting and handling read errors. This especially applies to applications in the context of the *EPCglobal network*[2]. An RFID middleware component named "Savant" has also been part of the initial *EPCglobal* architecture specification [9]. Finally *EPCglobal* has decided to leave the implementation of the middleware up to software vendors and to only specify the interface (ALE – Application Level Events) between the RFID middleware and business application. Several implementations of this specification already exist [6]. While the attention on RFID middleware is almost completely focused on stationary applications, it has only been regarded very little in the mobile context [28]. However, especially for mobile applications a mobile RFID middleware could help to make the development process of mobile RFID-enabled application easier and more efficient. When dealing with middleware approaches in the mobile context, the focus is mostly put on different aspects of ubiquitous computing applications such as context modeling, context detection, or data and network connectivity management but not on integrating RFID in order to create a direct bridge to the physical world [7, 25, 23].

In the field of mobile RFID applications, *NFC (Near Field Communication)* has gained some attention over the last years [1]. NFC is a standard released by the *NFC*

[1] CORBA (Common Object Request Broker Architecture)
[2] EPCglobal Inc.: http://www.epcglobalinc.org

Forum[3] for hardware, protocols, and partly also software components to primary enable mobile phones with RFID capabilities. Most NFC standards have either been ratified as ISO standards or cover existing ISO standards[4]. Since NFC offers different sub features like peer-to-peer communication, smart-card emulation and basic RFID reader capabilities for accessing certain 13.5 MHz high-frequency transponders, there are many possible use cases for NFC. Most use cases focus on applications that support transactions like micro-payment or electronic ticketing. But also applications like "smart posters" where the URL to a web page is encoded into a passive RFID transponder which is embedded into public posters [21]. NFC seems to be a promising solution for mobile RFID applications but currently lacks support in both services provided and hardware. Also only very few mobile phone handsets are available so far that support NFC. For application development only a basic SDK from *Nokia* is available [22]. The NFC SDK is mainly focused on low-level support of the developer. High-level concepts are only supported partly and need to be explicitly addressed. NFC is a standard that focuses on mobile RFID applications but cannot be regarded as middleware that supports a developer with ready-to-use high-level functionality.

The idea of providing a framework that allows a rapid application development process (RAD) for RFID applications has already been proposed so far [14]. Since mobile applications are getting more important, a RAD approach is also reasonable here. A middleware that addresses both data management and services and offers out-of-the-box functionality can help to facilitate this approach. A similar idea is also presented by Wu et al. [28]. They propose a service-oriented architecture that gives developers the possibility of constructing a customized mobile OSGi[5]-compliant RFID middleware to support different combinations of requirements. Apart from that a fully fledged RFID middleware for mobile applications is not yet available.

In order to offer a better support for developing mobile RFID applications, we propose an architecture for a mobile RFID middleware in this paper. While RFID middleware in the stationary context often has to handle large volumes of data and is mainly used for supporting data collection processes, in a mobile scenario the requirements are different: In most cases information about physical objects that is stored in a database either within the mobile device itself or on a remote database server needs to be retrieved upon selecting a physical object by its RFID transponder. Also in some cases the transponder has the ability to store data on the chip. Apart from that not only data aspects need to be addressed; in many cases it is desirable to also map functionality which physical objects advertise. These can be referred to as "physical services". Finally, certain events like reading a transponder need to be captured and processed further in order to facilitate process integration. Other general aspects of mobile information systems are connectivity and data management. A mobile RFID middleware will also have to address these. Some mobile applications serve as thin clients and are connected to a remote server via networks like WLAN, GSM, etc. Other applications work completely offline while there are also many hybrid variants that cache or replicate data in order to work also in temporary offline situations.

---

[3] The NFC Forum: http://www.nfc-forum.org

[4] The NFC Forum Specifications: http://www.nfc-forum.org/specs/

[5] OSGi (Open Services Gateway initiative)

These requirements have been taken into account when designing our *ID-Services* middleware. Apart from hardware abstraction a main concept of *ID-Services* are *proxy objects* that represent a placeholder for a physical object in the virtual world of the mobile information system that can be accessed in the same way by the developer as objects known from OOP (object oriented programming). The developer does not (necessarily) have to care about RFID specific characteristics like transponder IDs or how to access the reader hardware – he can fully focus on the development of the actual business logic of the application. The idea is to provide a flexible RFID middleware for mobile applications that allows a rapid application development (RAD) by mapping data, services and process related events to proxy objects that can be used by developers in a convenient and easy way.

3 ID-Services Architecture

ID-Services is an architecture of an RFID middleware for mobile applications. Basically it is not bound to a specific hardware or an operating system platform. We have chosen the *Microsoft .NET-Compact-Framework*[6] for a reference implementation *(ID-Services.NET)* because applications written for this platform can be executed on multiple devices that use the *Windows Mobile* or *Windows CE*[7] operating system, no matter what processor is actually used [29]. Furthermore developing applications for this platform is well supported and a lot of components and functionality like database systems, replication, access to web-services, etc. are provided by the *.NET-CF*. Furthermore many mobile applications in the industry and business context today are *Windows CE*-based.

The main idea of the *ID-Services* middleware is to support the developer as much as possible when implementing an RFID-enabled application but also to still offer a maximum of flexibility. Therefore we have chosen a three-layer-approach which offers at each layer access to RFID-specific functionality based on different levels of abstraction (see Figure 1). Each layer has a core concept on which the main focus is put on. While the first layer is dedicated to hardware abstraction, the second layer offers a seamless integration of data functions and processes by providing "proxy objects". Finally the third layer provides access to high level concepts such as "physical hyperlinks" [24], "physical services" and process integration. Developers can choose with which layer they want to interact and to what extend, depending on the type of application being developed.

The (lowest) first layer *(ID Device Layer)* represents a hardware abstraction layer that allows the developer to access the RFID reader device over a standardized interface. This layer offers all features and functions that can be expected from a "conceptional" mobile RFID device in a standardized form. These include basic functions like reading or writing transponder IDs or accessing the transponder's build-in memory on a raw level (if available). Because not all reader devices offer the same functio-

[6] Microsoft .NET Compact Framework:
http://msdn2.microsoft.com/en-us/netframework/aa497273.aspx

[7] Microsoft Windows CE: http://www.microsoft.com/windows/embedded/default.mspx

nality and not all transponders have features such a writeable memory, the *ID Device Layer* also offers an interface to query the reader's and the transponder's capabilities. Usually the device capabilities are known in advance to the developer but this approach allows writing a single piece of software that supports multiple hardware devices with different capabilities and features. Also operating the reader device is done by the *ID Device Layer*: A reader can either be used in a user-triggered mode which means that the user of an application activates the reader to select a transponder for example by pressing button or it can be used in a "polling mode" where the reader continuously scans for transponders within range and reacts as soon a transponder has been detected. This type of operation mode is especially useful when the intention of the application is to provide an intuitive physical user-interface that is touching a transponder with the mobile device [26]. In order to support a specific RFID device, the driver or API that is supplied by the device manufacturer, is accessed by a hardware-specific *ID-Services-Connector* components which maps the functionality of the specific hardware to the *ID Device Layer*. This abstraction allows using different RFID hardware devices without any changes to the application or other parts of the middleware. Only the *ID-Services-Connector* needs to be modified or re-implemented. Another advantage of this approach is that an RFID device does not necessarily have to be a "real" hardware device. A reader device can also be completely implemented in software (for example the "DummyReader" that is part of the *ID-Services.NET* reference implementation). Such virtual RFID devices can support (automated) testing scenarios or may be helpful during the development phase when the actual hardware is not available to all developers, because mostly mobile applications are being developed using an emulator instead of the actual hardware.

If developers want to implement the required, RFID-specific functionality themselves, this can be done by accessing the various features of the *ID Device Layer*. So far *ID-Services* offers a hardware abstraction that allows access to RFID hardware and to perform rather low-level RFID specific tasks. If the developer needs more support when interacting with physical objects, the second layer which is called *Physical Object Layer* offers further-reaching functionality.

The core concept of the *Physical Object Layer* is "proxy objects". Physical objects from the real world have characteristics that need to be mapped with the data model and functions of the information system. The idea of proxy objects is to keep all RFID-specific aspects such as transponder IDs away from the developer and to provide a placeholder for a physical object in the information system that can be directly accessed and modified in the same way as "ordinary" objects known from object oriented programming. The idea of mapping database entities into objects is not new and is a well known and proved concept used in many applications. In most cases an OR-Mapper (object-relational mapper) is used to fulfill this task [3]. But the idea of proxy objects goes beyond this approach. While typical meta-data like the name of the physical object is stored in the proxy objects properties, it also offers support for methods and events. Methods are used to advertise services that are provided by physical objects (referred to as "physical services") and events are used to allow the application developer to perform certain tasks during the interaction with the physical objects. While the properties and the methods of a proxy object depend on the physical object which the proxy object represents, the events that can be fired by a proxy object are always the same: `OnCreated()`, `OnModified()` and `OnCommit()`

The `OnCreated()` event is fired when a proxy object is created. The only way to "create" a proxy object at runtime is to scan the RFID transponder which is associated with the physical object. As a result this event can be used for logging or tracking purposes. If a property of the proxy object is modified the `OnModified()` event is fired. Finally the `OnCommit()` event gets fired when a proxy object has been modified and its properties are now being made persistent.

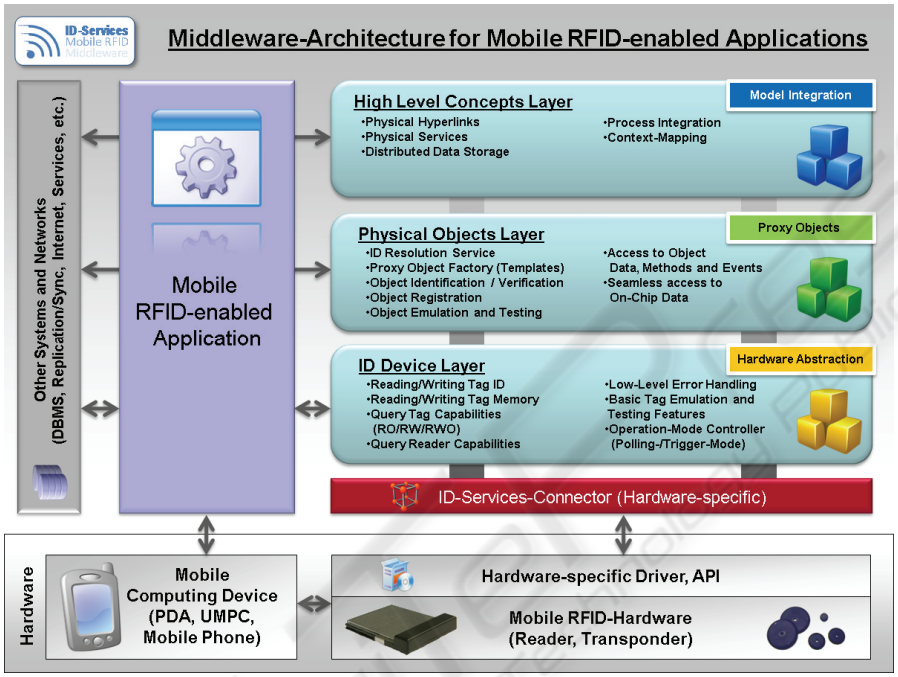

Fig. 1. Three-level-architecture of the *ID-Services* RFID middleware.

Proxy objects can either represent real, physical objects or "virtual objects" that can be used to describe fixed locations or transponders without any object affiliation. The basic process of interacting with proxy objects is described in the following: Before a physical object can be used in the application, an RFID transponder needs to be attached to it. This transponder needs to have a unique identifier which is referred to as *Tag-ID*. The level of uniqueness of this identifier depends on the specific application. Applications like those in the *EPCglobal network* need a world-wide unique identifier[8] that also fulfills certain criterions like a scalable resolution process [15] while in a closed-loop scenario, an application-wide uniqueness of the identifier is sufficient. Since the idea of the *Physical Object Layer* is to hide all low-level RFID-specific aspects from the developer, there is no need to enter the Tag-ID into the system. *ID-Services* also provides a functionality to register new transponders into the system. The process of registration takes the Tag-ID and maps it to a proxy object. A proxy object is created by the *Proxy Object Factory* which is closely integrated with

[8] EPC (Electronic Product Code)

the OR- and service-mapper. The source of the data that is mapped with the proxy object can either be a local database or any other kind of (remote) data source (for example a data-centered web service). It is also possible that data which is mapped to a property is originated from the transponder's internal memory which makes addressing the transponders memory very easy. To be able to fulfill the mapping task, the *Proxy Object Factory* uses meta-data that has been provided by the developer in advance. This meta-data contains the Tag-ID, the data source that is used for the mapping and the value of a primary key filed that refers the object's data. Also further mapping information like the names and data types of the mapped properties that are exposed by the proxy objects are provided. Additionally to the mapping of the properties, also methods can be mapped to the proxy object. Therefore either locally implemented method calls or remote web services are referenced in the proxy object meta-data. The parameters of these methods represent the context of a method invocation and are filled with values upon invocation. Typical context elements are the user (-name) that uses the application and the point of time when the service is invoked. Parameter values that have been queried from the user over the applications GUI can also be used.

In order to speed up the process of modeling proxy objects a concept similar to the "prototypes" from the experimental object oriented programming language *SELF* [12] is used. Prototypes can be regarded as class definitions. But in contrast to traditional OOP programming models, no inheritance is available. Instead a copy of an existing prototype is created and modified according to the developers needs (for example by adding or removing properties). We use the term "template" instead of "prototype" in this context. Templates can also be merged to quickly add new properties or methods to a new object definition. This very flexible concept allows a quick creation of proxy objects and helps to speed up the OR- and service mapping process. In order to analyze the structure of a proxy object at runtime, a reflection mechanism can be used, if required.

Before the *Proxy Object Factory* can create a proxy object out of a template and the object's data from the database, a resolution service maps the Tag-ID with the appropriate proxy object meta-data. This resolution service can either be a simple table look-up on a locally available database (which is usually replicated in order to propagate changes back to the backend-system) or a remote resolution service (for example a service similar to an *ONS (Object Naming Service)* like proposed in to the *EPCglobal* architecture). The resolution service is a pluggable component in the architecture that can also be implemented by the developer to support different kinds of resolution types.

After the proxy object has been created, it can be used by the application in the same way as "traditional" objects know from OOP. All data management and the invocation of methods are controlled by the OR- and service-mapper. The whole process which consists of the Tag-ID resolution, the proxy object creation and finally the use of the proxy object is shown in Figure 2.

Finally the idea of the third level (*High Level Concepts Layer*) which has not yet been implemented so far in the reference implementation is to provide access to typical high-level concepts of mobile, RFID-enabled applications such as "physical hyperlinks", "physical services" and a seamless integration of RFID-functionality into mobile (business) processes. While interaction with proxy objects still requires the

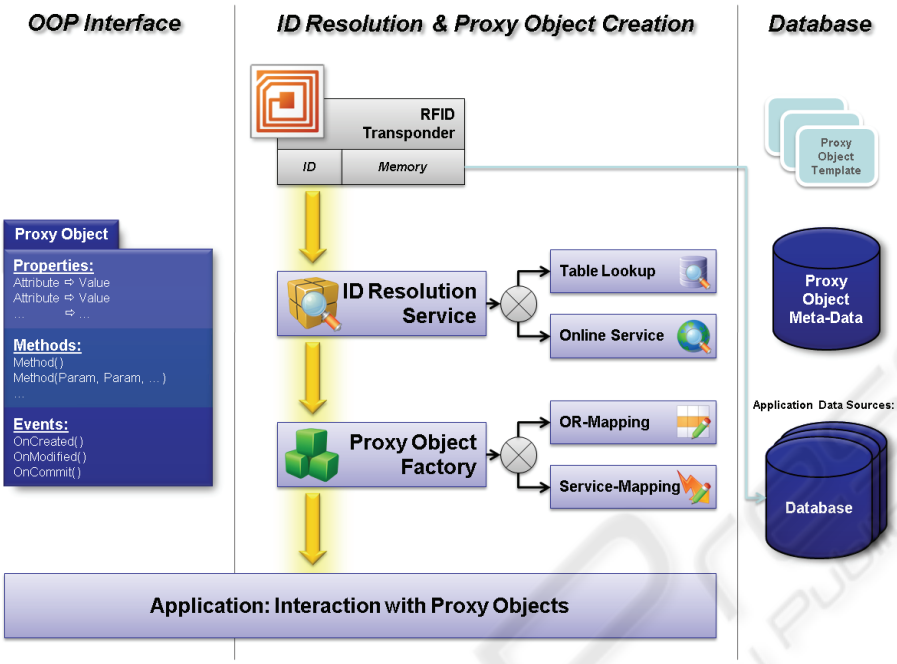

Fig. 2. Resolution of Tag IDs into proxy objects that represent physical objects.

developer to implement the desired features himself, the *High Level Concepts Layer* is supposed to provide ready-to-use functionality for certain applications. In case of physical hyperlinks a local or remote resource (for example a web page) is referenced, which needs to be retrieved in order to be presented to the user. Physical services might need additional parameters from the user in order to be invoked. Another aspect regards the fact that an RFID transponder might advertise more than just one physical hyperlink or service. This leads to the conclusion that the user will have to use some kind of selection mechanism, for example a menu on the mobile computing device. Currently we investigate what characteristic functionality of mobile RFID applications can be generalized in an appropriate way to be used for other application types. A major requirement is that this functionality has to be "compact" enough to be provided in a ready-to-use way by the *ID-Servcies Middleware*.

4 Use Case and Implementation

ID-Services can be used for many types of mobile, RFID-enabled applications. Those applications that make use of high-level concepts such as *physical hyperlinks, physical services* or that require a close integration of physical objects into business processes derive most value from the use of this middleware architecture. An application that we have developed in cooperation with *Elatec GmbH*, a German RFID- and electronics company, is *MoVIS (Mobile Visitor Information System)* [24], a PDA-based multimedia guide system for museum visitors. By touching RFID tags that are

attached to or near the exhibits, a visitor can access individually adapted multimedia information (like text, speech, images, video and even interactive content such as quizzes) with a PDA that has a 125KHz LF RFID reader integrated. The basic idea of *MoVIS* is to use RFID transponders as an intuitive extension of the user interface in order to allow an easy access to information about museum exhibits. With *MoVIS* a visitor can explore the museum on his own initiative or do a guided tour. The transponders are either used to identify exhibits in the explorative mode or to verify exhibits in the tour mode.

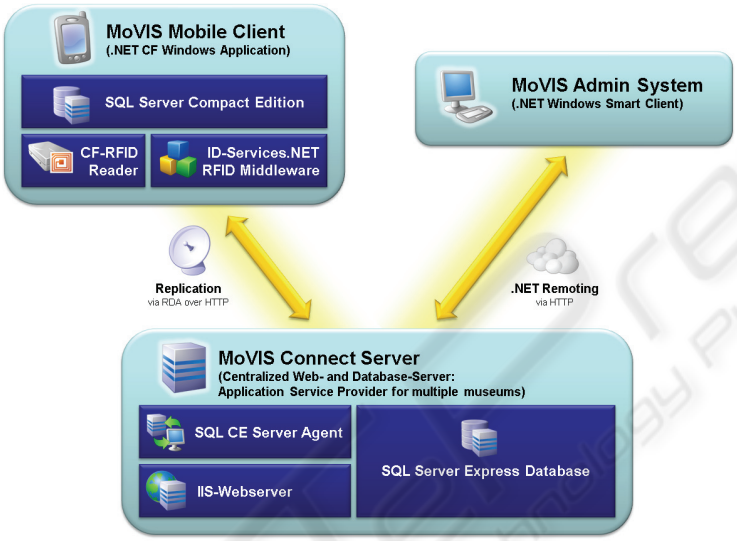

Fig. 3. Architecture of the Mobile Visitor Information System.

The application and the content are stored locally on the mobile device. This "offline approach" has been chosen to make the system more stable and to lower the systems requirements (for example a WLAN network is not necessary). The content is replicated on the device from a central database server using the *RDA (Remote Data Access)* replication mechanism of *SQL Server CE* (see Figure 3). A standalone smart-client application that connects to the server via *.NET-Remoting* is used by the museum staff to add and maintain content and to analyze the reports that are being generated by the system from the anonymously collected tracking data of the visitors. The tracking data is generated by the code following the `OnCreated()` events of the exhibit proxy objects.

The use of *ID-Services* highly simplifies the use of RFID in this case because the application developer does not have to deal with low-level concepts such as Tag-IDs etc. – he can fully concentrate on working with objects such as "exhibits", "rooms" and the actual content. But also the museum staff can profit from this, because common tasks such as resetting a device after it has been returned by a visitor or synchronizing data with the backend server have been made available through *physical services*. By touching special transponders, which are not available to visitors, the desired maintenance service is invoked which makes them very easy to execute without

any further interaction on the PDA's screen. These "service tags" are implemented as virtual proxy objects. But also adding new exhibits (transponders) to the system is simple because it is supported by *ID-Services*. By using another "service tag", the device can be put into a mode that allows registering new transponders. The new transponder is simply attached to the exhibit and then scanned with the PDA's reader device. The museum operator can give the transponder a speaking name and after synchronizing with the backend server, the newly registered transponder can be associated with an exhibit or content. Both the museum operator and the developer of the application will never see the ID which is stored on the ready-only transponders that are used for this application. The whole ID resolution is done internally by *ID-Services*. *MoVIS* is currently being used and evaluated at the "Mühlenhof" open-air museum in Münster, Germany to inform visitors about the exhibits and their historical background.

5 Summary and Conclusions

The use of *RFID middleware* as a central component in RFID-enabled applications has become common for most applications where stationary readers are involved. An RFID middleware is usually used to abstract from the actual hardware and to provide RFID-specific functionality such as cleaning or aggregating RFID data, as well as handling massive data-streams and to offer a close integration into information systems. In the case of mobile RFID applications, such a kind of middleware has not been used so far. Therefore developers have to directly access the hardware over the drivers and APIs that are supplied by the device manufacturers. This makes application development a very hardware-centered task. Even though mobile RFID applications do have different requirements in comparison to "stationary" RFID applications, the concept of a middleware component that mediates between the hardware and the actual application can also be applied here, as well. While a basic hardware abstraction fosters compatibility and portability and also allows an easier testing process, typical high-level concepts and functionality such as mapping physical objects with entities from databases or services can also be supported.

Our approach allows an easy, yet flexible access to RFID-specific functionality in the context of mobile applications, depending on the individual level of support that is needed by the developer for a specific application. A developer can choose to use very basic RFID functionality such as reading a transponder's ID or a transponder's memory storage with only a few lines of code over a standardized API or he can make use of ready-to-use high-level functionality. This includes concepts like *physical hyperlinks* or *physical services* as well as common ID management and resolution tasks. The flexibility is achieved by separating the functionality into three bottom-up-stacked layers that can be accessed according to the developers needs.

The conceptual architecture of *ID-Services* is basically platform-neutral and can be used for any kind of mobile computing device. The reference implementation *ID-Services.NET* which makes use of several built-in features of the *.NET Compact Framework* and also the *SQL Server CE* database engine has been used to implement *MoVIS*, an RFID-based, mobile visitor information system for museums. Many other

applications from completely different domains use similar high-level concepts which are natively supported by *ID-Services*. An issue that has not been addressed so far is security and privacy. Some applications will require ways to protect data that is stored on the transponders or prevent that data which is stored on the mobile device or the network is being accessed by non-authorized users [16]. The further development of *ID-Services* will also focus on this issue, as well as the ability to allow a seamless integration into existing mobile (business) processes which will be a part of the *High Level Concepts Layer*.

References

1. ABI Research: “Near Field Communication (NFC): Leveraging Contactless for Mobile Payments, Content and Access”, 2007
2. ABI research: RFID Annual Market Overview, 2007, http:// www.abiresearch.com/products/ market_research/RFID_Annual_Market_Overview
3. Ambler, W.S.; Mapping Objects to Relational Databases: O/R Mapping In Detail, 2006, http://www.agiledata.org/essays/mappingObjects.html
4. Anshel, M. & Levitan, S. Reducing medical errors using secure RFID technology SIGCSE Bull., ACM, 2007, 39, 157-159
5. Burnell, J. What Is RFID Middleware and Where Is It Needed? RFIDupdate.com, 2006
6. Chen, N.; Chang, T.; Chen, J.; Wu, C. & Tzeng, H. Reliable ALE Middleware for RFID Network Applications CSREA EEE, 2007, 183-189
7. Dabkowski, A. XML-Based Middleware for Mobile Systems Berliner XML Tage, 2003, 432-438
8. Dortch, M., Aberdeen Group, Inc.: "Winning RFID Strategies for 2008", 2008, http://www.aberdeen.com/summary/report/benchmark/4205-RA-winning-rfid-strats.asp
9. EPCglobal, "The Application Level Events (ALE) Specification, Version 1.0," Specification, February 8, 2005
10. Finkenzeller, K. RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification John Wiley & Sons, Inc., 2003
11. Floerkemeier, C. & Lampe, M. RFID middleware design - addressing both application needs and RFID constraints GI Jahrestagung (1), 2005, 277-281
12. Hölzle, U. & Ungar, D. Reconciling responsiveness with performance in pure object-oriented languages ACM Trans. Program. Lang. Syst., ACM, 1996, 18, 355-400
13. Jeffery, S.R.; Alonso, G.; Franklin, M.J.; Hong, W.; Widom, J. Virtual devices: an extensible architecture for bridging the physical-digital divide. Tech. Rep. UCB-CS-05-1375, UC Berkeley CS Division, 2005
14. Kim, Y.; Moon, M. & Yeom, K. A Framework for Rapid Development of RFID Applications Computational Science and Its Applications - ICCSA 2006, 2006, 226-235
15. Kindberg, T. Implementing physical hyperlinks using ubiquitous identifier resolution WWW '02: Proceedings of the 11th international conference on World Wide Web, ACM, 2002, 191-199
16. Konidala, D. & Kim, K. Mobile RFID Applications and Security Challenges Information Security and Cryptology ICISC 2006, 2006, 194-205
17. Leaver, S. C.; Mendelsohn, T.; Spivey Overby, C. & Yuen, E. H. Evaluating RFID Middleware Forrester Research, Inc., 2004
18. Legner, C. & Thiesse, F. RFID-Based Facility Maintenance at Frankfurt Airport IEEE Pervasive Computing, IEEE Computer Society, 2006, 5, 34-39

19. Liu, S.; Wang, F. & Liu, P. Integrated RFID data modeling: an approach for querying physical objects in pervasive computing CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, ACM, 2006, 822-823
20. Nath, B.; Reynolds, F. & Want, R. RFID Technology and Applications IEEE Pervasive Computing, IEEE Educational Activities Department, 2006, 5, 22
21. NFC Forum: NFC Smart Poster Record Type Definition, SPR 1.1, NFCForum-SmartPoster_RTD_1.0, 2006
22. Nokia 6131 NFC SDK 1.1, http://www.forum.nokia.com/info/sw.nokia.com/id/ef4e1bc9-d220-400c-a41d-b3d56349e984/Nokia_6131_NFC_SDK.html
23. Park, S.; Kim, D. & Kang, B. Context-aware Middleware Architecture for Intelligent Service in Mobile Environment CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06), IEEE Computer Society, 2006, 240
24. Schwieren, J. & Vossen, G. Implementing Physical Hyperlinks for Mobile Applications using RFID Tags. IDEAS 2007, 154-162
25. Son, M.; Kim, J.; Shin, D. & Shin, D. Research on Smart Multi-agent Middleware for RFID-Based Ubiquitous Computing Environment Agent Computing and Multi-Agent Systems, 2006, 787-792
26. Välkkynen, P.; Korhonen, I.; Plomp, J.; Tuomisto, T.; Cluitmans, L.; Ailisto, H. & Seppä, H. A user interaction paradigm for physical browsing and near-object control based on tags 2003
27. Wang, S.; Chen, W.; Ong, C.; Liu, L. & Chuang, Y. RFID Application in Hospitals: A Case Study on a Demonstration RFID Project in a Taiwan Hospital hicss, IEEE Computer Society, 2006, 8, 184a
28. Wu, J.; Wang, D. & Sheng, H. Design an OSGi Extension Service for Mobile RFID Applications icebe, IEEE Computer Society, 2007, 0, 323-326
29. Yao, P.: Microsoft .NET Compact Framework for Windows CE .NET, July 2002. http://msdn2.microsoft.com/en-us/library/ms836805.aspx