ID-Services: an RFID middleware architecture for mobile applications

Joachim Schwieren · Gottfried Vossen

Published online: 31 July 2009

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 which is 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 *ID-Services*, an architecture for an RFID middleware that is designed to support mobile applications. The *ID-Services* 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.

Keywords RFID · Middleware · Mobile computing · Mobile RFID · Architecture

A preliminary version of this paper appeared in the Proc. 2nd Int. Workshop on RFID Technology—Concepts, Applications, Challenges (IWRT), Barcelona, Spain, 2008, pp. 19–31.

J. Schwieren · G. Vossen (✉)
European Research Center for Information Systems (ERCIS),
University of Münster,
Leonardo-Campus 3,
48149 Münster, Germany
e-mail: vossen@ercis.de
URL: http://www.ercis.de

J. Schwieren
e-mail: schwieren@ercis.de

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 an information system, which is also referred to as “physical digital divide” (Jeffery et al. 2005), 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 (Dortch et al. 2008). 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. 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 and industry contexts (e.g., supply chain automation, tracking and tracing of shipping goods, extending the level of automation in production and manufacturing processes, etc.) (ABI Research 2007b; Finkenzeller 2003). Even though these RFID applications originate from completely different fields, most applications have in common that reader devices are stationary, whereas transponders are mobile. In this paper we address the opposite case of *mobile readers* that opens a new and quite different branch of RFID applications which we refer to as “Mobile RFID”.

Applying RFID technology in mobile applications implies that the RFID reader device itself is a mobile unit (which is usually integrated with a mobile computing device) (Liu et al. 2006; Nath et al. 2006). 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, a combination with RFID opens the door to a wide range of new possibilities and applications. Since mobile applications usually have a higher affinity to the physical world in which they are used (“object-centered”), the need for a close integration into the physical world is 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.

Especially in mobile applications, the user interface is often very limited due to the small form factor. An additional tangible user interface, implemented with mobile RFID technology, can help to simplify the interaction with those applications. Examples of mobile RFID applications are maintenance applications (Legner and Thiesse 2006), where buildings, objects, or machines need to be maintained from time to time. Moreover, field force automation applications that help field staff to quickly access information about product samples they carry around or keeping track of field service activities are examples of mobile RFID applications. In the healthcare and medical context, mobile RFID applications can be used to identify patients or medicine to prevent errors in treatment (Anshel and Levitan 2007) and to speed up documentation processes (Wang et al. 2006). Another application, which also follows the design and development methodology we have described in (Schwieren and Vossen 2009), is tracking and documenting sentry patrols by using transponders that are applied to fixed physical locations (“checkpoints”) (Hunstig 2008). Finally, a more consumer-oriented use case is mobile guide systems like *MoVIS* (Schwieren and Vossen 2007), 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. In this scenario RFID can also be used to support augmented reality applications (Behzadan et al. 2006).

Though there are many possible use cases for mobile RFID applications, these applications are still not widely spread. A major reason for this, apart from the limited availability of suitable hardware, is the fact that developing such applications is not a trivial task: RFID hardware is very heterogeneous and many different standards and technologies exist. Since the integration of specific RFID hardware is typically the primary 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 the first prototype of *MoVIS*, we have encountered several challenges, but also general requirements that can be applied to other mobile RFID applications as well; these issues include identifier resolution, information retrieval on physical objects, or tracking physical interactions. To cope with these in a general fashion, we have developed an RFID middleware architecture for mobile applications, *ID-Services*. Since *ID-Services* is a platform-neutral architecture design and specification, we have moreover built a reference implementation called *ID-Services.NET* which is based on the .NET Compact Framework. This reference implementation has been used to implement the production version of *MoVIS,* after a first prototype version had 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 from that derive the motivation for our approach. Before we present the *ID-Services* architecture in Section 4 in detail, we give an overview of the characteristics of mobile RFID applications in Section 3. Section 5 discusses a particular use case for *ID-Services*, and Section 6 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 low-level interaction. In addition, 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* (Floerkemeier and Lampe 2005). Recently, all major database system and ERP vendors have started to provide products that they refer to as *RFID Middleware* (Leaver et al. 2004). In contrast to traditional middleware (like, for example, CORBA[1]), RFID middleware can be seen as a mediator between the RFID reader hardware and an underlying information system. Even though this term is frequently used in the literature

[1] CORBA (Common Object Request Broker Architecture)

and also in commercial products, there is no exact definition of what RFID middleware is and where it is located in an application's architecture (Burnell 2006); some RFID reader devices even include middleware in the reader's embedded controller unit.

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 (EPCglobal 2005). 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 (Chen et al. 2007). While the attention on RFID middleware is almost completely focused on stationary applications, it has only been regarded very little in the mobile context (Wu et al. 2007). However, especially for mobile applications mobile RFID middleware could help to make the development process of mobile RFID-enabled application easier and more efficient, as we have described in (Schwieren and Vossen 2009). 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 (Dabkowski 2003; Son et al. 2006; Park et al. 2006).

In the field of mobile RFID applications, *NFC (Near Field Communication)* has gained some attention over the last years (ABI Research 2007a). NFC is a standard released by the *NFC 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 (Forum: NFC Smart Poster Record Type Definition & SPR 1.1, NFCForum-SmartPoster_RTD_1.0 2006). 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 which includes the *Java Contactless Communication API* (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).

The idea of providing a framework that allows a rapid application development process (RAD) for RFID applications (in general) has already been proposed so far (Kim et al. 2006). 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 presented by Wu et al. (Wu et al. 2007). 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. The approach by Wu et al. is basically the application of an ALE (Application Level Events Specification) compliant RFID middleware to mobile systems with a special focus on hardware integration. The specific use cases and appliances of mobile RFID applications in general are not addressed.

3 A general framework for mobile RFID applications

In order to offer better support for developing mobile RFID applications, we propose an architecture for 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 quite 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" which are made available by special "service tags". Finally, certain events like reading a transponder need to be captured and processed further in order to facilitate process integration.

[2] EPCglobal Inc.: http://www.epcglobalinc.org
[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)

Other general aspects of mobile information systems are connectivity and data management. 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.

In order to systemize and to structure the different aspects of mobile RFID systems regarding both conceptional functionality and the technical foundation, we have developed a general framework that takes all relevant aspects into account (see Fig. 1).

The aspects of the framework have also been taken into account as requirements when designing our *ID-Services* middleware architecture. Apart from hardware abstraction a central concept of *ID-Services* are *proxy objects* that represent a placeholder for a physical object in the virtual world of the mobile information system. Proxy objects can be accessed in the same way by the developer as objects known from OOP (object-oriented programming) (Hölzle and Ungar 1996). 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. This follows the concept of providing a *"metaphysical data independence"* as proposed by Jeffery et al. (Jeffery et al. 2008).

The idea of *ID-Services* is to provide a flexible RFID middleware architecture for mobile applications that allows a rapid application development (RAD) by integrating data, services and process related events to proxy objects so that developers are able to create mobile RFID applications in a convenient and easy way.

4 The 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 Microsoft's *.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 (Yao 2002). 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.

4.1 Overview

The main idea of the *ID-Services* middleware architecture 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 that follows the *relaxed layered system architecture pattern* (Buschmann et al. 1996) which offers at each layer access to RFID-specific functionality based on different levels of abstraction (see Fig. 2). 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 interaction with physical objects based on data functions and processes by supporting a special concept called "proxy objects". While first and the second layer provide the "infrastructure" of the middleware architecture, the third layer finally provides access to ready to use high level components and functionality that can be used in conjunction with the other layers. The functionally provided supports applications such as "physical hyperlinks" (Schwieren and Vossen 2007), "physical services" and basic process integration.

Developers can choose with which layer they want to interact and to what extent, depending on the type of application being developed. This also implies that it is not required to implement the architecture as a single monolithic component. It can also be implemented only partially (if certain components are not needed) or it can be implemented in a distributed fashion, where either the layers or certain components within these layers reside on other systems than the actual mobile device.

4.2 The ID Device Layer

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 "conceptual" 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 functionality and not all transponders have features such a writeable memory, the *ID Device Layer* supports both readers and transponders with capabilities. Since the representation of RFID hardware is based on an extensible interface, new features such as sensors on transponders or special features

[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

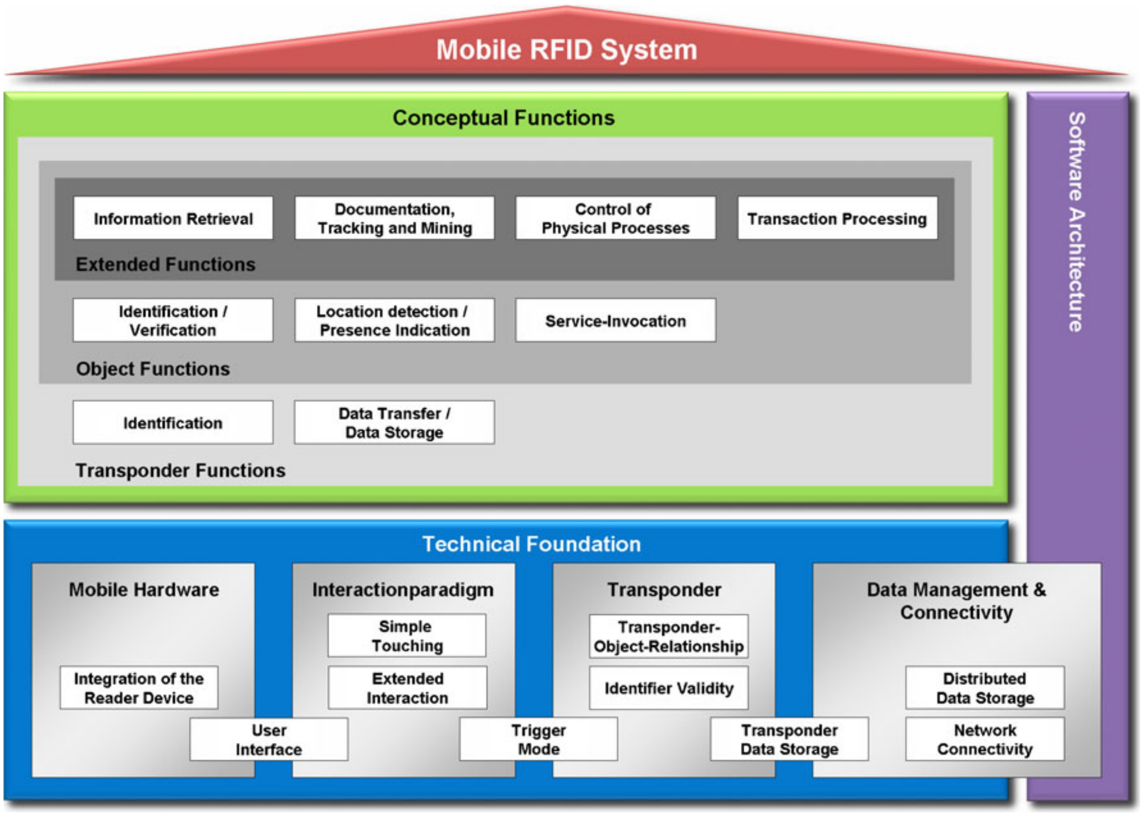
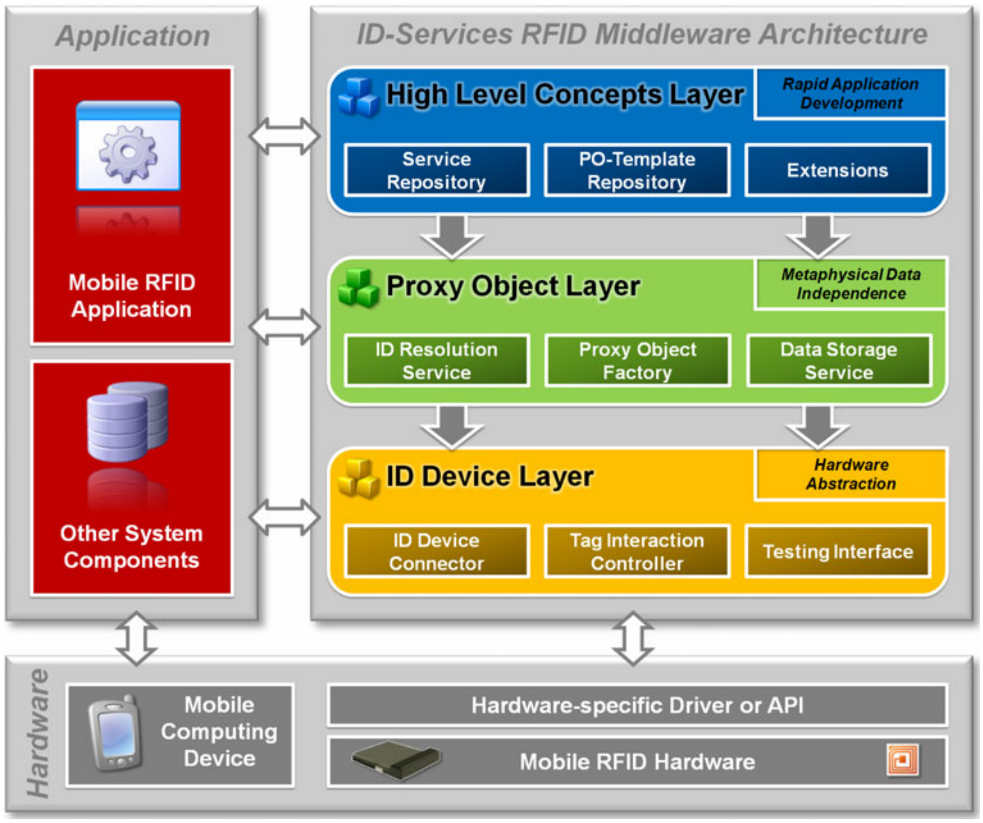


Fig. 1 General framework for mobile RFID systems

Fig. 2 Three-level-architecture of the *ID-Services* RFID middleware architecture



of certain readers such security-enhanced functions can easily be integrated without having to modify the entire middleware architecture. Usually the device capabilities are known and defined 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 completely done by the *ID Device Layer*. Therefore no "typical" functions such as initialization or de-initialization of the reader are provided by the *ID Device Layer* to the client. Instead the *ID Device Layer* itself cares about keeping the reader device ready to use by itself. This also includes an automatic device recovery feature that gets activated when the reader device needs to be reactivated, e.g. after the mobile device was turned off and on afterwards.

The *Tag Interaction Controller* which is a core component of the *ID Device Layer* allows to use the reader either 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 (Välkkynen et al. 2003). 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 Device Connector* component 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 Device 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 "Virtual Reader" that is specified in the *High Level Concepts Layer*). 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 directly use the required, RFID-specific functionality, 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 *Proxy Object Layer* offers further-reaching functionality.

4.3 The Proxy Object Layer

The core concept of the *Proxy Object Layer* is "proxy objects". Physical objects from the real world have characteristics that need to be mapped with the data and process model of the information system. This also includes the functionality of interacting with these objects. 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, modified and interacted with 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 (Ambler 2006). 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 event handlers. Methods can be used to advertise services that are provided by physical objects (referred to as "physical services") and event handlers are used to allow the application developer to perform certain tasks during the interaction with the physical objects.

Proxy objects can either represent real, i.e., physical objects and locations (fixed locations) or transponders representing "virtual objects" like service tags without any object affiliation. The basic process of interacting with proxy objects is described next: 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 criteria like a scalable resolution process (Kindberg 2002) while in a closed-loop scenario, an application-wide uniqueness of the identifier is sufficient. Since the idea of the *Proxy 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* offers functionality to register new transponders into the system. The registration process takes the tag id and maps it to a proxy object. In order to allow a flexible association between tag id's that reference physical objects and proxy objects that are referenced by their type (*proxy object template*) and their instance identifier (*contextual id*), an *ID Resolution Service* handles all aspects of this (registering, deregistering and resolution of tag id's). 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

[8] EPC (Electronic Product Code)

resolution service based on web services (for example a service similar to an *ONS (Object Naming Service)* like proposed in to the *EPCglobal* architecture). A resolution service is a pluggable component within the architecture that can be individually implemented by the developer to support different kinds of resolution types.

By supplying a tag id from the *ID Device Layer* during interaction with a transponder, the *ID Resolution Service* resolves the tag id into one or multiple pairs of proxy object templates and contextual id's (since a transponder can also be associated with multiple proxy objects). The structure of a proxy object is descried by the proxy object template which basically is a class definition. This class needs to implement a special interface (*IProxyObject*) in order to be handled as proxy object template. The contextual id is of a variant data type. It refers to the instance of a specific proxy object. In case of a database oriented application this can be a primary key value but also any kind of OID or even an URL is possible here. Only in combination with the proxy object template the contextual id has a meaning.

The *Proxy Object Factory*, which is part of the *Proxy Object Layer*, takes a pair of proxy object template and contextual id from the *ID Resolution Service* and creates the proxy object (by using a mechanism like reflection since the object is created dynamically). After the proxy object has been created, the OnCreated() event handler of this proxy object instance is called by the *Proxy Object Factory*. This phase is called *"impersonation"* according to the *ID-Services* terminology. Because the contextual id cannot be forwarded using the proxy objects constructor (an interface cannot declare the signature of a constructor), this is done using the OnCreated() method. Therefore this method is also referred to as *deferred constructor*. Inside the deferred constructor the proxy object initializes itself based on the contextual id which is provided. After that the proxy object receives the control flow of the application and can take over the further interaction. In contrast to a centralized event handler that gets called when a proxy object is created, this method allows to store the code that refers to a proxy object within the proxy objects class definition. This highly increases the clearness of the code and the applications architecture. It is even possible to extend an application with a new proxy object template without the necessity of having to modify the application itself.

Since a proxy object has no knowledge about the application by default, the *Proxy Object Layer* holds a reference to a special object named *"interaction context"*. This interaction context object bridges the application and the proxy objects and also holds contextual information like the current state of the application or any other relevant information that a proxy object will need in order to properly react upon creation. The complete process of the creation of a proxy object is visualized in Fig. 3.

Proxy objects simplify the interaction with physical objects that are identified by RFID transponders. Proxy objects do not only store data about physical objects within their properties (data perspective) but also control the interaction between with the application when being created (process perspective). Additionally, proxy objects offer methods to access the memory of the transponders to which they are associated to. The data access functionality is provided by a *Data Storage Service* which is a component of the *Proxy Object Layer*. In contrast to the (low-level) read and write methods provided by the *ID Device Layer*, a *Data Storage Service* allows proxy objects to access a transponders memory on a higher abstraction level. A *Data Storage Service* does not only handle the data access but also manages the way the data is stored on the transponder (e.g. by interpreting the raw bytes and adding semantics to them in the way of data types).

4.4 The High-Level Concepts Layer

Finally the idea of the third level (*High Level Concepts Layer*) is to support rapid application development for mobile RFID applications. Many components of the *ID Device Layer* and the *Proxy Object Layer* like, e.g. the *ID Device Connector*, the *ID Resolution Service* or the *Data Storage Service* are only specified as interfaces. Though their functionality is as well specified in a conceptual way, there are many possible ways of implementing them. For example an *ID Resolution Service* can either be a local resolution service where the resolution table is stored on the mobile device itself, or it can also be implemented in a way where the resolution table is stored on a remote system. While the use of such components still requires the developer to implement the desired features himself, the *High Level Concepts Layer* provides ready-to-use functionality for typical use cases and applications.

The *High Level Concepts Layer* consists of two repositories that contain these components: The *Service Repository* contains *ID Resolution Services*, *Data Storage Services* and a *"Virtual Reader" ID Device Connector* while the *Proxy Object Template Repository* contains proxy object templates that can be used to facilitate the implementation of service tags or to easily map URLs to physical objects. But not only "plug-in" components for the *ID Device Layer* and the *Proxy Object Layer* are provided. The *High Level Concepts Layer* also includes an *Extension* that holds an interaction context object that can instantly deal with the predefined proxy object templates from the *Proxy Object Template Repository*. Furthermore, this interaction context object which is referred to as *Generic Interaction Context Controller (GICC)* can be used as base class for implementing an individual interaction context object.

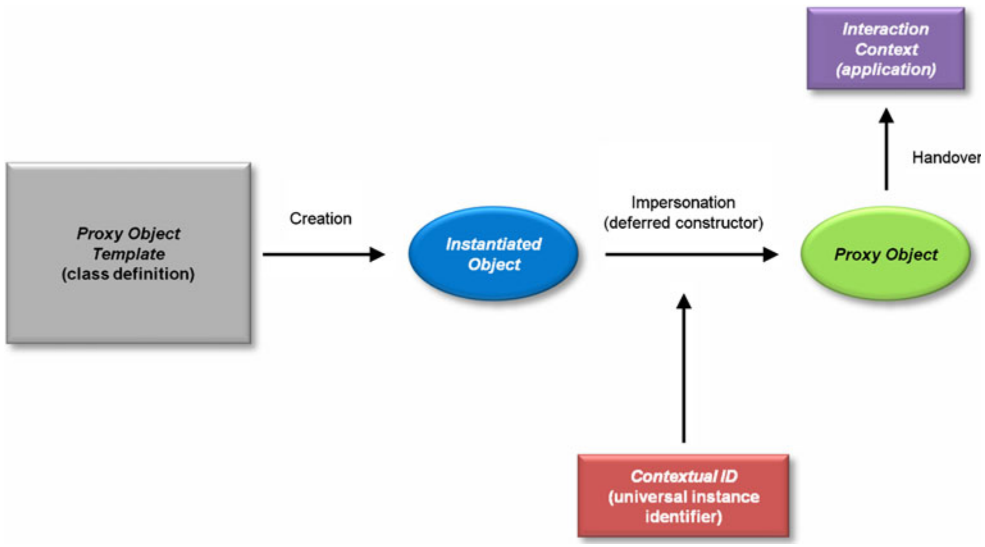

Interaction Context (application)
Handover
Proxy Object Template (class definition)
Creation
Instantiated Object
Impersonation (deferred constructor)
Proxy Object
Contextual ID (universal instance identifier)

Fig. 3 Process of creating of proxy objects

As shown, the ID-Services RFID middleware architecture for mobile applications copes with the heterogeneous use cases and manifold requirements of mobile RFID-enabled applications by providing a flexible architecture that can be customized to the actual needs by selecting the best-suited level of abstraction and by adding or reimplementing the necessary components that fulfill the specific requirements which are needed. Due to its flexibility this architecture can easily be used for the implementation of many types of mobile RFID-enabled applications, independently of the actual RFID hardware and system platform used, as explained in more detail in (Schwieren and Vossen 2009).

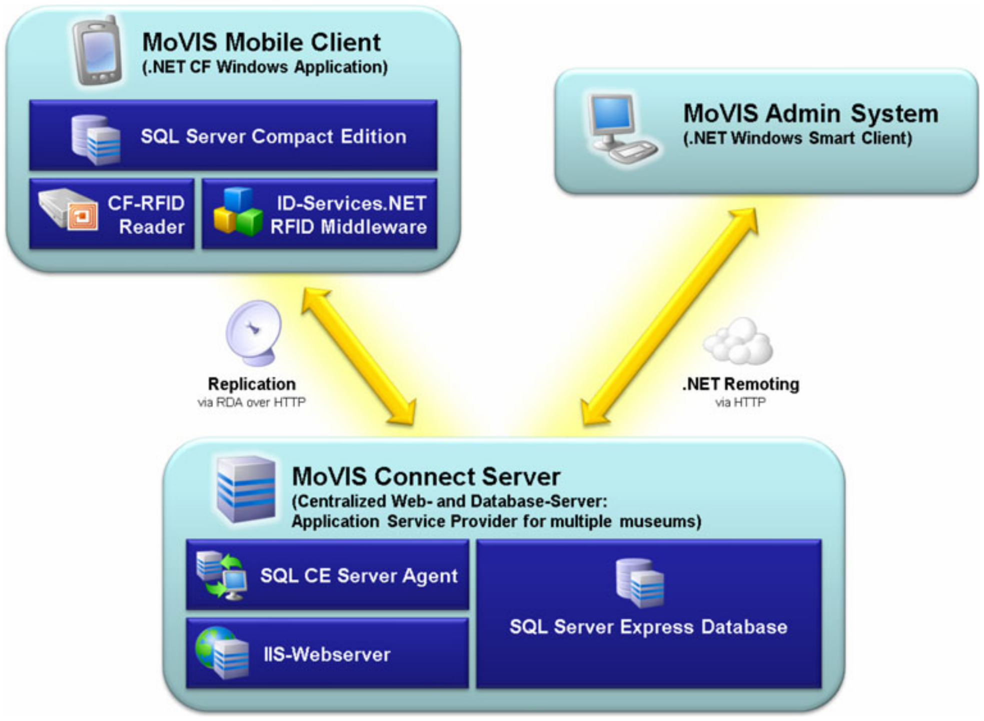
5 Use case and implementation

ID-Services can be used for a variety of mobile, RFID-enabled applications. 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)* (Schwieren and Vossen 2007), a PDA-based multimedia guide system for museum visitors. By touching RFID transponders 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 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.

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 Fig. 4). A stand-alone 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 transponder IDs etc.; indeed, he or she can fully concentrate on working with objects such as "exhibits", "rooms" and the actual content. But also the museum staff can benefit 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 proxy objects, too. 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

Fig. 4 Architecture of the mobile visitor information system


MoVIS Mobile Client
(.NET CF Windows Application)
SQL Server Compact Edition
CF-RFID Reader
ID-Services.NET RFID Middleware
MoVIS Admin System
(.NET Windows Smart Client)
Replication
via RDA over HTTP
.NET Remoting
via HTTP
MoVIS Connect Server
(Centralized Web- and Database-Server: Application Service Provider for multiple museums)
SQL CE Server Agent
IIS-Webserver
SQL Server Express Database

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 passive ready-only transponders that are used for this application. The entire ID resolution is done internally by *ID-Services*.

MoVIS has recently been used and evaluated during various events and at various locations, including the "Mühlenhof" open-air museum in Münster, Germany as well as the "Hightech Underground" technology exhibition of our university, where it was used to inform visitors about the respective exhibits and their historical or scientific background. Currently, MoVIS is used for presentations and public events all over the university (see Fig. 5) and will also be applied in some of the museums that are run by the University of Münster and beyond in the near future.

6 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. RFID middleware is usually used to abstract from the actual hardware and to provide RFID-specific functionality such as cleaning, aggregating or processing RFID data, as well as from 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 proposed so far. As a consequence, developers had to directly access the hardware through drivers and proprietary 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 dedicated middleware component that mediates between the hardware and the actual application can indeed be applied here as well. While a basic hardware abstraction fosters compatibility and

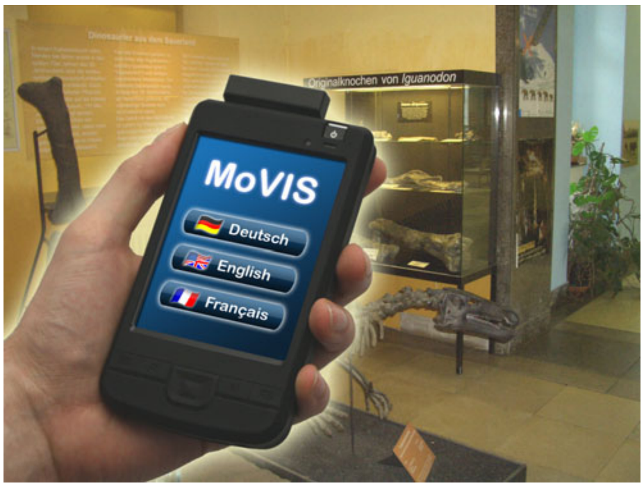


Fig. 5 MoVIS PDA with integrated RFID reader device in a museum

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 for 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 data storage (memory) 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. Our 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*. Apart from *Windows Mobile / CE*, the *Java ME* operation system platform is very common in mobile applications, especially in those that are executed on mobile phones. A further task will be the implementation of the *ID-Services* architecture on the *Java ME* platform.

Yet another 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 (Konidala and Kim 2006). Due to the modularized architecture, security-enhanced components such as special resolution services or data storage services can be easily integrated. These components will become part of the *High Level Concepts Layer* specification in order to be directly available for developers.

References

ABI Research (2007a): "Near field communication (NFC): leveraging contactless for mobile payments, content and access".

ABI Research (2007b): RFID Annual Market Overview. http://www.abiresearch.com/products/market_research/RFID_Annual_Market_Overview.

Ambler, W. S. (2006). Mapping Objects to Relational Databases: O/R Mapping In Detail, http://www.agiledata.org/essays/mappingObjects.html.

Anshel, M., & Levitan, S. (2007). Reducing medical errors using secure RFID technology SIGCSE Bull. *ACM, 39*, 157–159.

Behzadan, A. H.; Khoury, H. M.; Kamat, V. R. (2006). Structure of an extensible augmented reality framework for visualization of simulated construction processes, WSC '06: Proceedings of the 38th conference on Winter simulation, 2055–2062.

Burnell, J. (2006). What Is RFID middleware and where is it needed? RFIDupdate.com.

Buschmann, F., Meunier, R., Rohnert, H., & Sommerlad, P. (1996). Stal, M. Pattern-Oriented Software Architecture Volume 1: A System of Patterns, Wiley; 1st edition.

Chen, N., Chang, T., Chen, J., Wu, C. & Tzeng, H. (2007). Reliable ALE middleware for RFID network applications CSREA EEE, 183–189.

Dabkowski, A. (2003). XML-based middleware for mobile systems berliner XML tage, 432–438.

Dortch, M., & Aberdeen Group, Inc. (2008) Winning RFID Strategies for 2008. http://www.aberdeen.com/summary/report/benchmark/4205-RA-winning-rfid-strats.asp.

EPCglobal. (2005). The application level events (ALE) specification, version 1.0. Specification, February 8.

Finkenzeller, K. (2003). RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification John Wiley & Sons, Inc.

Floerkemeier, C. & Lampe, M. (2005). RFID middleware design—addressing both application needs and RFID constraints GI Jahrestagung (1), 277–281.

NFC Forum: NFC Smart Poster Record Type Definition, SPR 1.1, NFCForum-SmartPoster_RTD_1.0, 2006.

Hölzle, U., & Ungar, D. (1996). Reconciling responsiveness with performance in pure object-oriented languages. ACM Transactions on Programming Languages and Systems, *18*, 355–400.

Hunstig, A. (2008). Konzeptionierung und Entwicklung eines kontextsensitiven, RFID-basierten Wächterkontrollsystems. Master Thesis, University of Münster.

Jeffery, S. R., Alonso, G., Franklin, M. J., Hong, W., & Widom, J. (2005). Virtual devices: an extensible architecture for bridging the physical-digital divide. Tech. Rep. UCB-CS-05-1375, UC Berkeley CS Division.

Jeffery, S. R., Franklin, M. J., & Garofalakis, M. (2008). An adaptive RFID middleware for supporting metaphysical data independence. *The VLDB Journal, 17*(2), 265–289.

Kim, Y.; Moon, M. & Yeom, K. (2006). A framework for rapid development of RFID applications computational science and its applications—ICCSA 2006, 226–235.

Kindberg, T. (2002). Implementing physical hyperlinks using ubiquitous identifier resolution WWW '02: Proceedings of the 11th international conference on World Wide Web, ACM, 191–199.

Konidala, D. & Kim, K. (2006). Mobile RFID Applications and Security Challenges Information Security and Cryptology ICISC 2006, 194–205

Leaver, S. C., Mendelsohn, T., Spivey Overby, C. & Yuen, E. H. (2004). Evaluating RFID Middleware Forrester Research, Inc.

Legner, C., & Thiesse, F. (2006). RFID-based facility maintenance at frankfurt airport IEEE pervasive computing. *IEEE Computer Society, 5*, 34–39.

Liu, S., Wang, F. & Liu, P. (2006). 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, 822–823.

Nath, B., Reynolds, F., & Want, R. (2006). RFID technology and applications IEEE pervasive computing. *IEEE Educational Activities Department, 5*, 22.

Park, S., Kim, D. & Kang, B. (2006). 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, 240.

Schwieren, J. & Vossen, G. (2007). Implementing Physical Hyperlinks for Mobile Applications using RFID Tags. Proc. 11th Int. Conf. on Database Engineering and Applications (IDEAS), 154–162.

Schwieren, J. & Vossen, G. (2009). A Design and Development Methodology for Mobile RFID Applications based on the ID-Services Middleware Architecture; Proc. 10th Int. Conf. on Mobile Data Management (MDM), 260–266.

Son, M., Kim, J., Shin, D. & Shin, D. (2006). Research on smart multi-agent middleware for RFID-based ubiquitous computing environment agent computing and multi-agent systems, 787–792.

Välkkynen, P., Korhonen, I., Plomp, J., Tuomisto, T., Cluitmans, L., Ailisto, H. et al. (2003). A user interaction paradigm for physical browsing and near-object control based on tags.

Wang, S., Chen, W., Ong, C., Liu, L., & Chuang, Y. (2006). RFID application in hospitals: a case study on a demonstration RFID project in a Taiwan hospital hicss. *IEEE Computer Society, 8*, 184a.

Wu, J., Wang, D., & Sheng, H. (2007). Design an OSGi extension service for mobile RFID applications icebe. *IEEE Computer Society, 0*, 323–326.

Yao, P. (2002). Microsoft .NET Compact Framework for Windows CE . NET, July. http://msdn2.microsoft.com/en-us/library/ms836805.aspx.

Joachim Schwieren is CEO of mirabyte GmbH & Co. KG, a German-based ISV company that is focused on software development in the areas of Web development, enterprise communications and mobility - with a special focus on intuitive user interfaces and RFID integration. He received his master's and Ph.D. degrees in 2005 and 2009 from the University of Muenster in Germany. Between 2005 and 2009 he has also been working as research member at the database group of Prof. Vossen in the Department of Information Systems at the University of Muenster which is affiliated with the European Research Center for Information Systems (ERCIS). His research interests include mobile information systems, RFID integration, e-commerce and the Web. Prior to his academic activities he has been a start-up entrepreneur who founded his first business back in 1996.

Gottfried Vossen is a Professor of Computer Science in the Department of Information Systems at the University of Muenster in Germany. He received his master's and Ph.D. degrees as well as the German habilitation in 1981, 1986, and 1990, resp., all from the Technical University of Aachen in Germany. He has held visiting positions at the University of California in San Diego, at several German universities including the Hasso-Plattner-Institute for Software Systems Engineering in Potsdam near Berlin, at Karlstad University in Sweden and at The University of Waikato in Hamilton, New Zealand in 2003 and again in 2006. In 2004 he became the European Editor-in-Chief of Elsevier's Information Systems - An International Journal, and a Director of the European Research Center for Information Systems (ERCIS) in Muenster. In 2008 he additionally became the scientific director of Deutsche Informatik-Akademie, the oldest institution for executive IT and computer science education in Germany. His research interests include conceptual as well as application-oriented problems concerning databases, information systems, electronic learning, and the Web. Dr. Vossen has been member in numerous program committees of international conferences and workshops. He is an author or co-author of more than 200 publications, and an author, co-author, or co-editor of more than 20 books on databases, business process modeling, the Web, e-commerce, and computer architecture.