RFID: From Concepts to Concrete Implementation

Patrik Fuhrer
University of Fribourg
Department of Informatics
Bd de Pérolles 90
CH-1700 Fribourg
patrik.fuhrer@unifr.ch

Dominique Guinard
University of Fribourg
Department of Informatics
Bd de Pérolles 90
CH-1700 Fribourg
dominique.guinard@unifr.ch

Olivier Liechti
Sun Microsystems
Switzerland
Route des Avouillons 12
CH-1196 Gland
olivier.liechti@sun.com

Abstract. Technologies of identification by radio frequencies (RFID) and their standardization with the Electronic Product Code (EPC) architecture experience a fast development. After briefly introducing the common terminology of the RFID field and a short presentation of its current standards, this paper presents one possible application domain: the tracking of objects.

In this context, the distributed software architecture of a J2EE based assets tracking application, called RFIDLocator, is further described. RFIDLocator allows to trace electronically labeled objects within a predefined area (e.g. a building, a campus, a site, ...). Indeed, when tagged objects are moved, the monitoring information system is automatically informed thanks to readers deployed in their environment.

As the number of assets can increase dramatically, this kind of application has high requirements for scalability and reliability which are supported through the use of solid object oriented software systems (Enterprise Java Beans technology along with an implementation of the Event Manager standard). Some critical remarks about this emerging technology, the important questions it raises and the barriers it has to overcome to be fully accepted conclude this paper.

Keywords: RFID, Enterprise Computing, EPC, Savant, Java, RFIDLocator

A shorter version of this paper has been accepted for presentation at
IPSI-2006 MARBELLA*, February 10-13, 2006, Marbella, Spain.*

Table of Contents

1 Introduction

More than any other discipline, computer science jumps from one technology to another. Some will die early, some will remain longer. During the last years, the increasing use of Radio Frequency as a means of identifying objects, placed this technology close to the latter category. According to many experts [18], RFID (or Radio Frequency IDentification) matured and is in the process of becoming one of our everyday-life partners.
In this fast evolving context, the first goal of this paper is to provide a brief overview of the field's "state of the art". Thus, the second section begins with a short history of Auto-ID technologies and an introduction to the common terminology of this domain. Then it offers an overview of the standars forming the "Internet of Things", a vision shared by many computer scientists where the objects surrounding us would all be part of a global infrastructure: the EPC (Electronic Product Code) Network. Finally, the second section is concluded with the presentation of some interesting applications.
The third section describes a concrete application where the use of RFID and EPC presents a great benefit: the tracking of assets within a predefined area (e.g. a building, a campus, a site, ...).
The implementation of the described application presents many challenging issues requiring the knowledge of advanced object-oriented technologies and best practices. Indeed, different hardware devices (e.g. RFID readers, RFID transponders, servers) have to be interconnected, specific middleware has to be used (e.g. Sun Java System RFID Software) and enterprise software products have to be deployed (e.g. J2EE application servers).
Section 4 describes the technological choices and the adopted software architecture for the RFIDLocator, a fully functional prototype respecting all the related current standards and satisfying high requirements for scalability, robustness and reliability.
Finally, the conclusion summarizes the main achievements of this project and provides insight into the barriers RFID still has to overcome as well as its possible future.

2 RFID Overview

2.1 Short history

Radio Frequency IDentification (RFID) is part of the more general Automatic Identification systems (also known as Auto-ID systems). Auto-ID systems are not new. However, these systems are becoming more and more popular in many business areas. In short, the term Auto-ID groups the technologies helping computer to identify objects, animals or people[1].
In use since the seventies[2], Automatic Identification procedures were developed in order to create means of providing information about objects in transit. Figure 1[3] offers an overview of existing Auto-ID technologies.
Barcodes. Technically speaking, the barcode comprises a field of bars and empty spaces, vertically printed on a sticker or a product label. The sequence (bars, gaps) as well as the width of the sticks are converted into an ASCII[4] sequence using optical lasers and a complex set of mirrors. Barcodes are a worldwide available and popular mean of identifying objects since the late seventies. There is a simple reason for this long-lasting success: barcodes have been made of paper and ink which makes these tags cheaper that any other Auto-ID system. Many barcodes standards were published, with the most popular among them certainly being the EAN (European Article Number) code which is nothing but an extension

[1] In the rest of the paper, the word object is used and is to be taken in its wide meaning, ranging from non-living objects to animals or human beings.
[2] On June 26th 1974 in Ohio, USA the first product using barcodes, a 10-pack of Juicy Fruit chewing gum, is scanned at the check-out counter [3].
[3] On this figure, OCR stands for Optical Character Recognition.
[4] ASCII is the acronym of American Standard Code for Information Interchange.

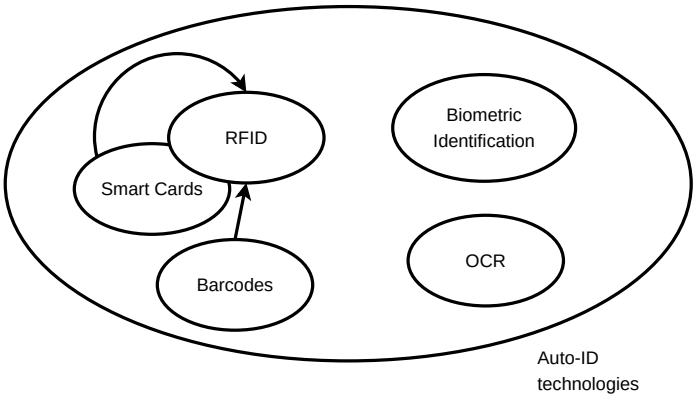

Figure 1: The Auto-ID technologies.

of the widely used UPC (Universal Product Code) introduced in the USA in 1974. Further information on the way barcodes evolved can be found in [50].

Smart cards. Invented in 1974 by the french inventor, Roland Moreno and first used in 1984, smart cards are credit-card sized pieces of plastic containing a data storage system and a microprocessor[5]. To extract or write data into such a card, one needs a special reader. For so called "contact" smartcards, the reader initiates a galvanic connection between the reader's metallic contact pins and the card contact surface (usually a gold-plated area of about 1 cm^2). After supplying energy and clock pulse, the reader is all set to start extracting (or writing) data out of the card. On the other hand "contact-less" smartcards work without galvanic contact but need to be really close to the reader.
Smart cards are widely used in the identification business: for instance, most companies use it to restrict access to the buildings by automatically identifying their employees.

RFID. Radio Frequency IDentification (RFID) is a method of remotely storing and retrieving data using devices called RFID *tags* or transponders. An RFID tag is a small object, such as an adhesive sticker, that can be attached to or incorporated into a product. RFID tags are composed of an antenna connected to an electronic chip and are thus basically contact-less smartcards. These chips transform the energy of radio-frequency queries from an RFID *reader* or transceiver to respond by sending back information they enclose[6]. Finally, a computer hosting a specific RFID application pilots the reader and processes the data it sends. This whole process is depicted in Figure 2.

The fast, automatic, pervasive and ubiquitous identification of living and non-living objects is one of the challenges of today's computer science (see [24]).
Thus Auto-ID technologies have to be highly flexible to be used in modern assets tracking applications with chaotic or unstructured business processes. RFID's ability to read objects *in motion* and *without a direct line of sight* give it the edge over traditional bar-coding methods relying on complicated readers.

Even if this aspect is the main difference between these two technologies, other minor differences are also opening the path for new Auto-ID applications. The increased (and increasing...) storage capacity of the transponders (up to 64 KBytes) and their readability under hard conditions (extreme temperatures, chemicals, high pressure,...) are just two other advantages of the RFID systems.

[5]The Central Processing Unit) is an 8 to 32 bits microprocessor sometimes coupled with a cryptographic coprocessor [24].

[6]In fact this is only true for passive tags. There are also active tags, which have batteries and initiate the communication to actively send information to readers.

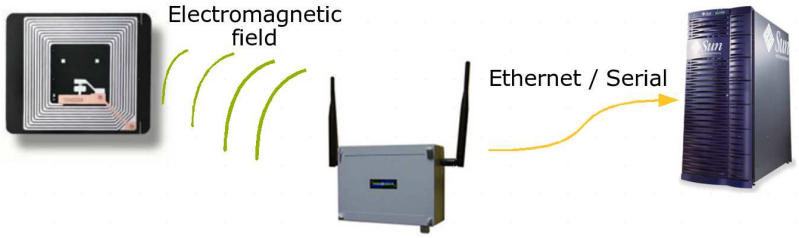

Figure 2: RFID transponders and readers interaction.

2.2 EPC Network Standards

Having labeled or tagged objects being identifiable in an ubiquitous, seamless and flexible manner is already a good start. Building a network out of these objects, so that with a unique number we can easily retrieve information about the detected object, would enable much more interesting use cases. In order to make the dream of a *seamless global network of physical objects* come true, an open standard architecture is required. The Auto-ID Center has been created in 1999 to foster this challenging task. After several years of research the *EPC Network* concept was born. In October 2003, the Auto-ID Center was split into two entities: the Auto-ID Labs [5] and the EPCglobal [14]. The former is the academic[7] part of the field and is chartered to research and develop the EPCglobal Network and applications. The latter is a neutral, consensus-based, non-profit organization[8] in charge of the administration of the EPC standards and of bringing then to the market. Although quite young, the EPC Network related researches already lead to several independent standards[9].

Electronic Product Code (EPC). The EPC Numbering System uniquely identifies objects and facilitates tracking throughout the product's life cycle. This makes it similar to the Universal Product Code or an EAN Code, with the main exception that EPC was primarily designed to be efficiently referenced on networks.

It is worth noting that as long as we stay in closed circuits, standards are not really needed. But if the goal is to have a global (i.e. between several companies and information systems) tracking system, standards such as EPC are mandatory.

The EPC is the fundamental identifier of assets in the so called EPC Network. It basically contains information about:

- The manufacturer of the tagged object.
- The product class or the nature of the tagged object.
- The actual (unique) item. This serial number is indeed the main benefit over "classical" barcodes where two bottles of orange juice, from the same brand, of the exact same kind, have the same code.

Each of the above information is encoded in a separate field which makes it quite easy to extract only part of the data. For instance, the sorting of items by manufacturers is made really straightforward by

[7]Auto-ID Labs is currently headquartered at the Massachusetts Institute of Technology (MIT) in Boston, USA and further based at six other leading universities worldwide: the University of Cambridge in the United Kingdom; the University of Adelaide in Australia; Keio University in Tokyo, Japan; Fudan University in Shanghai, China; the University of St. Gallen in Switzerland and the Information and Communications University (ICU) in Daejeon, Republic of Korea.

[8]EPCglobal Inc is a joint-venture between GS1 (formerly know as EAN International) and GS1 US (formerly the Uniform Code Council, Inc.).

[9]Some of which still need to be finalized in order to be fully adopted by industry partners.

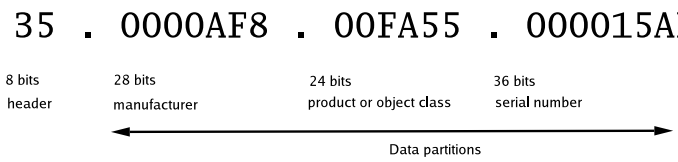

Figure 3: The hexadecimal representation of a GID-96 EPC.

Message 1:
Tag 8 (GID encoded) of our company was detected in Bern, Switzerland: today at 13h15 by the reader 3.
The temperature there is about 25 degree celsius.

(a) A simple, non-formalized, goods tracking message.

```
<Sensor>
    <ID>urn:epc:id:gid:1.1.3</ID>
    <Observation>
        <ID>1<ID>
        <Data>
            <XML>
                <TemperatureReading xmlns="http://sensor.example.org/">
                  <Unit>Celsius</Unit>
                  <Value>25</Value>
                </Temperature>
            </XML>
        </Data>
        <DateTime>2006-04-14T13:15:00.812+02:00</DateTime>
        <Tag>
            <ID>urn:epc:id:gid:1.1.8</ID>
        </Tag>
    </Observation>
</Sensor>
```

(b) The same message in standard PML format.

Figure 4: PML represents data captured by the RFID readers in a standard way.

this latter fact. There is an additional, rather technical, field whose purpose is to define which of the six standard encoding scheme [12] was chosen.

EPCs are often represented as Uniform Resource Identifiers (URI[10]) in order to be used on large networks and to be easily manipulated and exchanged by software applications.

Indeed, doing so enables the handling of EPCs in a tag-level independent manner and decouples the application logic from the way of obtaining the EPC. The URI contains the EPC fields required to distinguish an object from another. A pure entity representation of the EPC on Figure 3 is:

urn:epc:id:gid:2808.64085.88828

where the first field corresponds to the GID encoding schema, the second to the company prefix, the third to the asset type, and the last to the unique serial number.

To conclude, let us remark that a chip can be of type RFID while not EPC standard compliant; thus RFID transponders or "tags" storing in its chip a unique EPC identifier, are commonly called "EPC tags".

Physical Markup Language (PML Core). As defined in [16] the "Physical Markup Language (PML) is a collection of common, standardized XML vocabularies to represent and distribute information related to EPC Network enabled objects." It defines a generic markup language for information interchange modelling and encapsulating the data captured by the RFID readers. An example PML message is shown on Figure 4.

Object Name Service (ONS). According to [16], "the Object Name Service provides a global lookup service to translate an EPC into one or more Internet URLs where further information on the object may be found". In other words, the ONS provides URLs to authoritative information relevant to an object, as for instance the web site of the object's manufacturer (see Figure 5). ONS is based on the well known

[10]URIs are often defined as the super-set of the well-known URLs. More information on this topic can be found at [57].

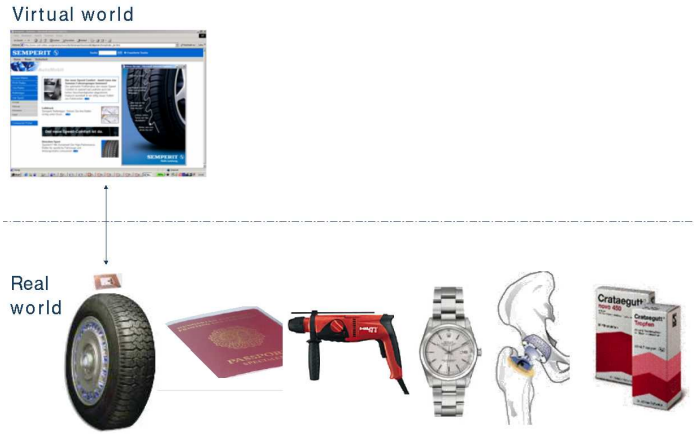

Figure 5: Linking real world things to homepages [33].

Domain Name Service (DNS) of the Internet, which is intended to map host names to their IP address[11]. More information about the ONS facilities can be found in [13] and [23].

Savant. The Savant[12] is the specification for standard *RFID middleware*, i.e. software that bridges RFID hardware and enterprise applications [32]. It defines an EPC events handling framework and is thus the primary means of data gathering for any RFID deployment. It acts as the central nervous system of the EPCglobal Network [15]. The following sentence found in [20] illusrates well the issue the Savant addresses: "If each object currently using a barcode were to make the use of [...] EPC, then the managing infrastructure would have to handle millions of events every second [...]".

The Savant is composed of three main components (see Figure 6):

- The Real-time In-memory Event Database (RIED), an optimized database. This component offers much better performances than a standard database when more than a few hundred transactions per second occur.

- The Task Management System (TMS), a generalized scheduling manager. It can be used to command the execution of various tasks in the Savant. It also contains modules for restarting tasks after a failure.

- The EMS (or Event Management System aka Event Manager or EM), in charge of managing the readers' output flows and preparing it for higher-level applications. The EM is a framework providing methods and tools to manage and capture EPC events by combining following three types of modules (see Figure 7): *(i)* the *adapters*, which are device drivers; *(ii)* the *filters*, which are responsible for the processing or the skipping of the received events; *(iii)* the *loggers*, which are used to monitor, forward or store the events. This flexible and modular architecture is analogous to the one described by the Intercepting Filter pattern of [4].

[11] In January 2004 EPCglobal awarded VeriSign a contract to manage the directory for looking up EPC numbers on the Internet [56]. This announcement was commented in many articles like [44], [25] or [7].

[12] The Savant is a registered trademark. The term Savant is currently being deprecated and is being referred to as Filtering and Collection Specification.

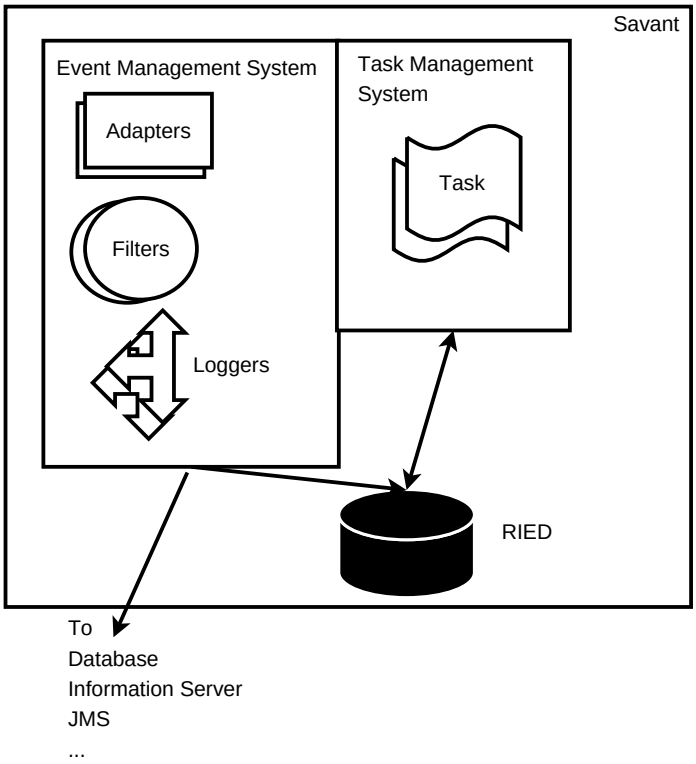

Figure 6: Components of the Savant.

2.3 Application domains

Using standardized identifiers (EPC), exchange formats (PML), query systems (ONS) and events processors (Savant), we are virtually able to connect every single object of this world to a global network. Indeed, if you paste an RFID label onto the table of your living-room and create a mapping between its description and the EPC stored on the attached tag, your table becomes part of a global network. Starting from this point, it can be identified and traced by computer systems coupled with RFID readers. This simple idea of matching a computer-readable standardized number to any object surrounding us has an incredible number of applications in various domains, such as access control (e.g. domestic door locks, ski lifts, public transportation...), logistics (e.g. laundry automation, smart shelves, bike rental businesses,...) or animal tracking (e.g. millions of sheep, cows, pigs, dogs, cats are identified and tracked).
Let us say some words about a few interesting or even "killer" applications.
Anti-Counterfeiting. Counterfeiting is a huge threat to global businesses[13] and concerns all kind of products and companies (e.g. pharmaceutical industry, automotive industry and their suppliers, luxury goods, media, food and beverage, banknotes, passports,...). The EPC Network allows the tracking and tracing of products and companies can thus assemble a product's history. Furthermore, by providing the EPC Network with an authentication server, the RFID technology allows the products to authenticate themselves to the user. Sun Microsystems recently released an RFID package focused on helping pharmaceutical companies track and authenticate drugs [31]. Thus RFID technology really helps to combat

[13]Counterfeiting drugs and knowingly offering them for sale has now evolved into a serious problem for world health and can have dramatic consequence on final users (see [30] and [11]).

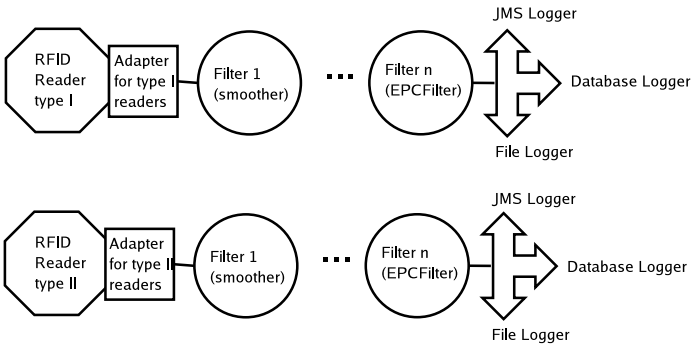

Figure 7: An example of modules combination in the EM.

counterfeiting, product piracy and smuggle [53].

e-Government. Since the tragic events of 2001, there is a great demand for higher security standards and governments require secure identification of individuals. Thus, electronic documents is one of the hottest topics in today's industry in the area of RFID technology. Sokymat [52], one of the leading manufacturer of RFID transponders, is only one example of a company which has decided to focus on RFID components for electronic documents such as e-passports or MRTD[14], as well as driving licenses, national ID cards.

e-Health. The healthcare sector is adopting RFID with enthusiasm. Among many possible use cases [43], RFID can be used to keep track of the entire medical information of a patient undergoing treatment, thus improving their safety. For example, a bluetooth equipped mobile phone can receive pill removal events from a smart blister pack and send them to the doctor's server via a GSM network. Questions such as "Did the patient get the medicine on time? Is it the right medicine?" can thus be easily answered [51].

Waste Management (see [45]). RFID is now being used by municipalities (for example in Victoria, Canada) to monitor waste pickup in order to promote recycling and reduce the amount of waste sent to landfills. RFID tags are installed on household waste bins and fully-automatic trucks lift each high-tech plastic cart, weigh and empty the garbage, and later download the information in order to accurately bill its owner according to the amount of trash he generates.

Documents Identification and Tracking. According to [36], RFID technology enables a breakthrough revolution in tracking documents. It is especially beneficial in those environments where the documents are of high value to the organization, and the temporary or permanent loss of a document would have significant negative impact. Examples are: government offices, hospitals and other medical offices, lawyers offices. Using RFID-based software system to locate files and track file history increases the overall efficiency and productivity[15]. Obviously the same principle holds for any kind of assets [42] or people. As an example [8], the Beth Israel Hospital in New York is using RFID to monitor tagged medical devices for location and staff and patients can be located easily due to the tagged bracelets they wear.

[14] MRTD stands for Machine Readable Traveler Documents.
[15] 3M's RFID Tracking System [2] is a concrete commercial example of such an application.

3 An assets tracking application

3.1 Description

The RFIDLocator application described in the rest of this paper addresses the assets tracking issue. The basis of the idea behind the RFIDLocator is quite simple but, as seen above, still satisfies a concrete need for many companies: "Given a set of objects moving (intensively) in a predefined area, where can we find them at a given time t".
The reader should understand at this point that RFIDLocator is not intended for assets tracking within the supply chain (i.e. RFID's favorite field) but for locating objects within a given area.
To make this important difference more obvious let us provide a use case example of this software in a hypothetical law firm:
Law and Co. employs more than 100 layers to respond to the demands of 20'000 clients. As in the law business paper is still quite relevant, the company stores two physical files (filled with contracts, important documents, etc.) for each of its clients' cases. Many lawyers are working on one case. As a consequence the files travel from one office to another several times a week. Thus, the lawyers spend quite a part of their days looking for the documents within the 20 floors the company occupies.
A straightforward solution to Law and Co's problem would be to hire less qualified (and thus, less paid...) employees to do the seeking job for the lawyers. Another solution could be to buy a dozen of RFID Readers, paste RFID labels onto each physical file and setup a software like RFIDLocator.
The lawyers would then only need to query the system in order to get the location within the building of Mister Doe's case.
This use case reflects a real need for an automatized way of tracking documents within the buildings of a company. The dubious reader is invited to look at a concrete example summarized in [1].

3.2 Object Model

To introduce the terminology used within the RFIDLocator application, its object model is described here in an implementation independent manner.
Location. The Location object models a *place* (e.g. a building, a room, a shelf, a desk,...) within the area controlled by the RFIDLocator. It is identified by a unique Business Location Number. The Location is the central object of RFIDLocator as the latter aims to *locate* physical objects. That is, to answer the question: given a physical object to look for, in which registered Location did it go through. As a consequence the LocatorObservations (i.e. observations that are valid in the context of RFIDLocator) are always linked to a Location.
TraceableObject. Such an object is a physical element that can be tracked by RFIDLocator. It could be about anything that is physically big enough to hold an RFID tag. The TraceableObject is the integration point between the legacy business system and the RFIDLocator application. As a consequence, it contains two fields that uniquely identify the object: *(i)* one on the business system side: the businessNumber (e.g. Case-Gu-2411); *(ii)* the other on the RFIDLocator side: the epcString (e.g. urn:epc:id:gid-96:1.1.150). The use of both these unique identifiers enables an object to be searched by businessNumber even if the latter is actually identified using its epcString for the RFIDLocator application and its attached readers. Furthermore, a TraceableObject is always attached to the User who registered it. Depending on the policies of the organization running the RFIDLocator software, displaying it when querying for the location of an object could be of great help. Indeed, in some cases, this User is likely to have a better knowledge of where the TraceableObject might be located.
User. A User is basically someone allowed to access and query RFIDLocator. This version of the application does not distinguish the users accordingly to their respective rights (administration, querying, etc.). Yet the reader should note that this fact could be of great relevance in a commercial application.

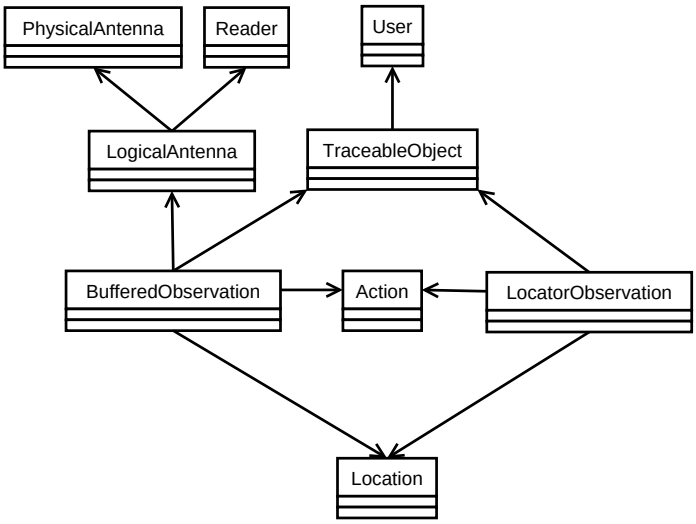

Figure 8: The object model of the RFIDLocator.

LocatorObservation. It is the persistent result of an RFID event. Its semantics is basically that a TraceableObject was *observed* at a given time going IN or OUT a particular Location. When tracing TraceableObjects, the queries will be made on these objects.
BufferedObservation. Such an Observation is basically a *potential* LocatorObservation. It is used by the solving algorithms when elements are missing to actually persist the Observation. Unlike a LocatorObservation, each of these objects is connected to a LogicalAntenna. This can be explained by the fact that the algorithms solving the Observations are commanded by the LogicalAntennae.
Action. An Action is the fundamental element for the algorithm that traces physical objects. This version of the RFIDLocator supports two actions: IN and OUT. An IN on a Location means that the object entered the Location, whereas an OUT means that the object exited the Location. The INs and OUTs are distinguished at the PhysicalAntenna level. That is: a PhysicalAntenna is always attached to either an IN or an OUT Action. The way the algorithms use these elements to find the actual Action that happened with an Observation is described in Subsection 4.3.
LogicalAntenna. A LogicalAntenna is an aggregate of $1..n$ PhysicalAntennae. It is used to determine what Action (i.e. IN/OUT) was effectively recorded by the n PhysicalAntennae. The LogicalAntenna[16] is thus the central component in the solving of an Observation. Furthermore, it is always associated with a Location.
PhysicalAntenna. A PhysicalAntenna represents the hardware able to capture RFID events by producing an electro magnetic field. As such an antenna is not standalone, it is always connected to a Reader. A PhysicalAntenna must be identified by and EPC (or another type of unique identifier). Finally, a PhysicalAntenna is always connected to a LogicalAntenna.
Reader. An RFID Reader (aka Sensor) is a physical hardware device controlling a set of $1..n$ PhysicalAntennae which detect tagged objects within their fields.

3.3 Working with the Application

The requirements of the RFIDLocator application are best addressed by a distributed software architecture. Indeed, the need to be able to use it from anywhere and not just on the computer which processes

[16]Basically, the concept of LogicalAntenna was created in order to be able to capture the direction of the motion using $2..n$ PhysicalAntennae. Indeed, most RFID readers do not have the native ability to capture the tag's direction.

Figure 9: The thin-client GUI of the RFIDLocator.

the observations, is a reality.
Its GUI (Graphical User Interface) is a web based thin-client. Thus the application is accessible by typing the application's URL [22] in the address bar of your web browser. The welcome page of the application is shown on Figure 9. All the pages of the user interface are based on the same well structured template: *(i)* the main menu is placed on the left: *(ii)* a quick access to the same functionalities is provided in the bottom zone; *(iii)* the content of the page is displayed in the "body" of the page; *(iv)* the upper right corner contains a link to logout (i.e. end the session of the current user) and another one to the API and documentation of the application. The main functionalities accessible using the menu are briefly described below, while a more detailed description, a guide for the administrator of the application as well as some hints for the programmer interested in extending RFIDLocator are available in [23].
Create a new user. This link permits to create a new user. All the fields are required. Once the form is filled and submitted the new user can login via the home page[17].
Configure the Environment. This page is required to set the initial environment of RFIDLocator. An XML description of which reader is located where has to be provided in the text field. The syntax of the XML that must be used is described in Figure 14. Figure 10 provides a screenshot of the readers' configuration page.
Attach / Detach a Tag. To inform the RFIDLocator that it should trace an object, one needs to attach an RFID Tag to the object both physically and virtually. This page offers to do the latter.
First the user enters the BusinessNumber. As described in Subsection 3.2 it is a unique string that the company uses to identify the object (e.g. case_JP_guinard_05 or portable_computer_0205).
Then, the user enters the unique number of the RFID tag. It corresponds to the unique number recorded

[17]The interested reader can test the application on-line [22] with following test user who has administrator rights: enter test as username and test as password.

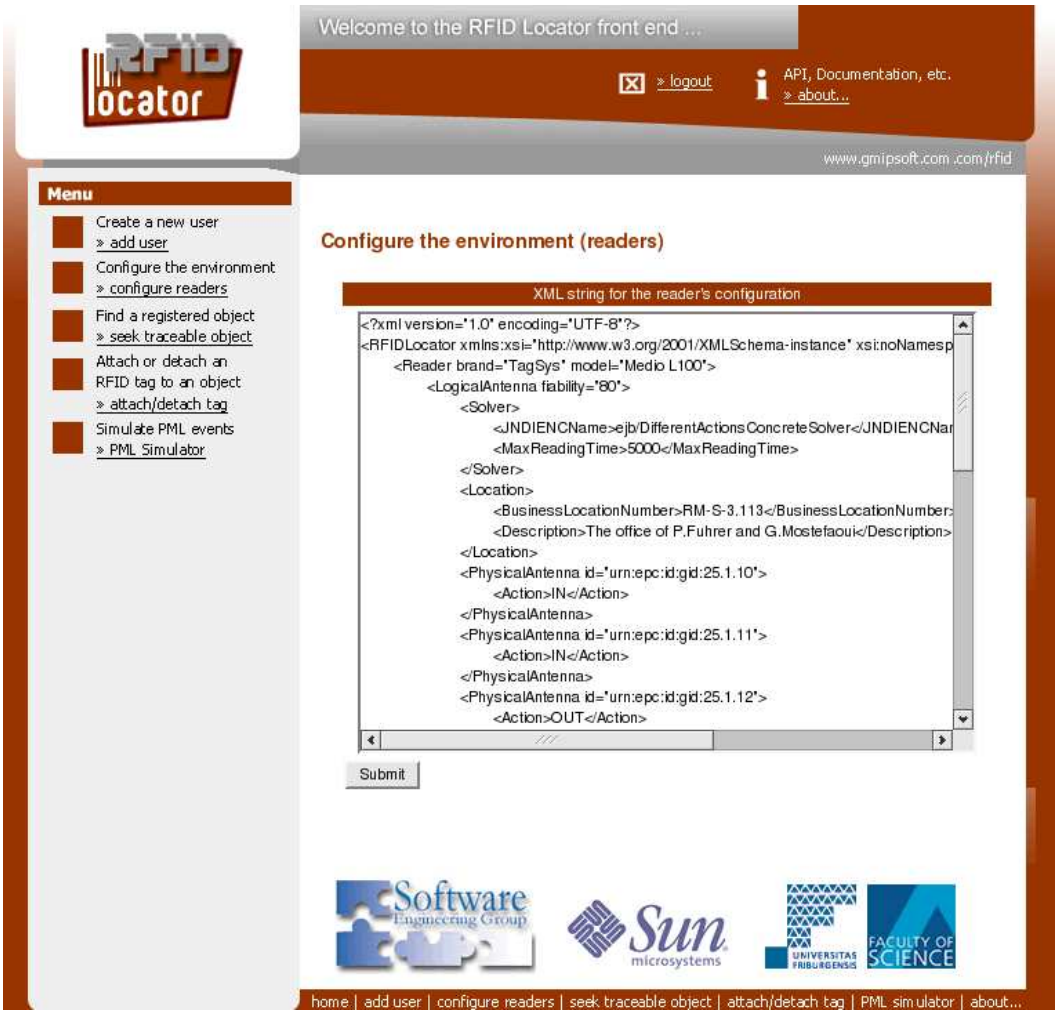

Figure 10: Readers' configuration page.

on the tag that is physically pasted onto the object.

Finally the user can choose between Attach and Detach. The former binds the tag id to the BusinessNumber, creating a TraceableObject. The latter deletes a TraceableObject from the system. It is worth noting that only the BusinessNumber is mandatory when Detach is selected.

Once a tag is attached to an object, it becomes a Traceable Object and thus, can be traced by RFIDLocator.

Find a Registered Object. This permits the user to seek a Traceable Object by providing its BusinessNumber. The system will return all the observations the RFID readers made from this particular object. Figure 11 presents the results of a typical seeking query.

Simulate PML Events. This page offers to simulate the events reported by an RFID reader. It permits to test the application without having an actual physical reader. The user is prompted for a PML string that describes an observation[18].

[18]Section 3.3 of [23] provides detailed information on the syntax of the PML.

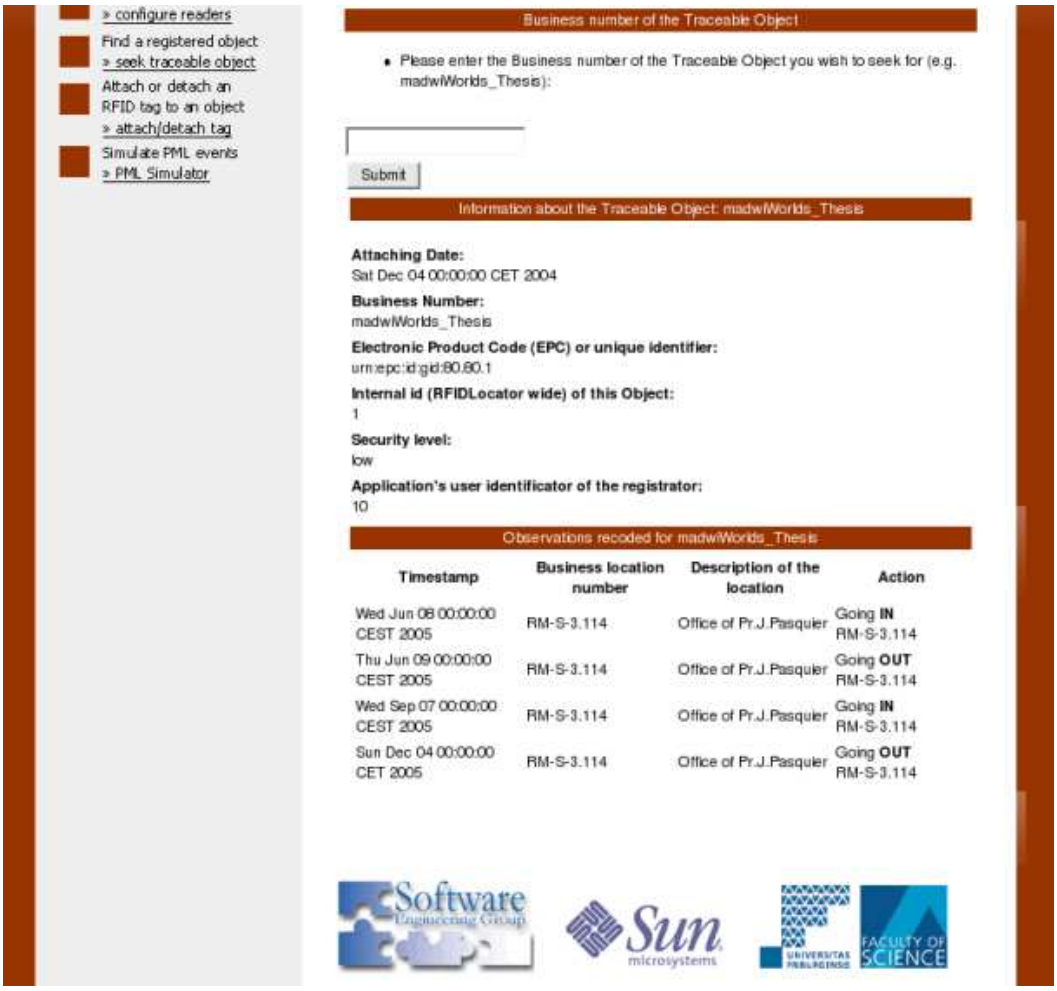

Figure 11: Seeking a Traceable Object.

4 The RFIDLocator Software Architecture

4.1 Technological choices

Before going into further details of RFIDLocator's distributed software architecture, it is worth explaining the technological choices that had to me made prior to its implementation.

Event Manager. The first software component required for an enterprise application working with RFID and EPCs is an implementation of the Event Manager. Such software are available on the market but not all of them meet the Savants' standards elaborated by the EPCglobal. Still, several software vendors provide flexible and standard compliant RFID middleware: *(i)* IBM and its WebSphere RFID Premises Server (see [9] and [26]); *(ii)* Oracle and its RFID and Sensor-Based Services [39]; *(iii)* the RFID application proposed by SAP [46] as an extension of its well known ERP[19]; or *(iv)* the Sun Java System RFID Software (see [55] and [54]). We opted for Sun's implementation of the Event Manager, one of

[19]ERP is the acronym of Enterprise Resource Planning.

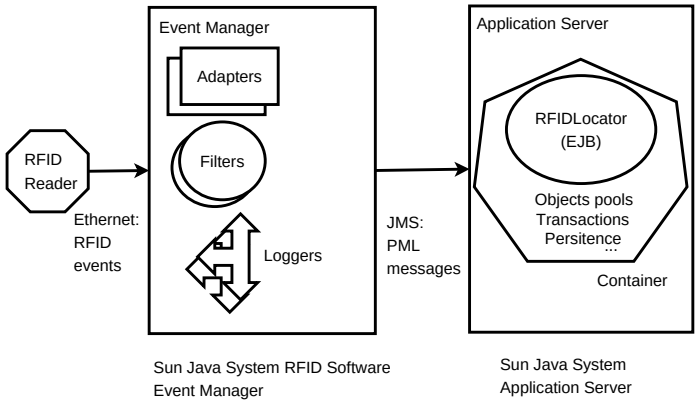

Figure 12: The Event Manager and the Application Server.

the first Savant's implementation on the market. The fact that Sun Microsystems is among the leaders and early movers of the RFID software field is just one reason that influenced our choice.

Enterprise Java Beans. Nowadays many technologies propose to solve the problem of distributed computing. However, for enterprise applications, where robust and reliable software is a need, two technologies are leading the field. The first is the .NET Framework [10] of Microsoft Corp. and the second is known as J2EE [27]. The war against the detractors and admirers of both technologies is certainly not over. Nevertheless in a concern of choosing an *open* standard, the J2EE and its "EJBs" (or Enterprise JavaBeans[20]) were finally chosen. The Enterprise Java Beans technology was adopted because they are managed by an application server, thus *(i)* offering a complete and integrated solution for the persistence layer; *(ii)* making the integration of JMS messages easier; and *(iii)* reducing development effort by providing services for dealing with critical issues such as scalability, concurrency and security.

Application Server and Database. The Application Server is a software on which a J2EE application can be deployed. Because of the EJB specification, the choice of an Application Server should not be an irreversible decision. Indeed, any Application Server that implements the EJB specification would theoretically be able to run the RFIDLocator without any changes (or after some minor changes in reality). However, the RFIDLocator is best designed for the Sun Java System Application Server coupled with the bundled PointBase RDBMS[21]. We choose them for convenience reasons: both are freely available for download from [47] and avoid compatibility issues, as they lead to a completely Sun based configuration. Among the other free and open-source application servers that should be able to run RFIDLocator let us cite: *(i)* Jonas [29]; *(ii)* JBoss Application Server [28]; *(iii)* Apache Geronimo Application Server [19]. Similarly, about every RDBMS that can be accessed from a J2EE architecture is potentially able to host the RFIDLocator database. Among the well-known databases that are easily coupled with J2EE let us cite: *(i)* MySQL [35]; *(ii)* Oracle Database [38]; *(iii)* PostGreSQL [41].

Finally Figure 12 shows the respective role of the Event Manager and the Application Server in our RFID application. Besides these software choices, adapted RFID hardware (readers, connectors, servers, RFID tags) must also be selected. The hardware settings for the RFIDLocator project are explained in [23], and Figure 21 depicts the physical deployment of the different parts of this distributed application.

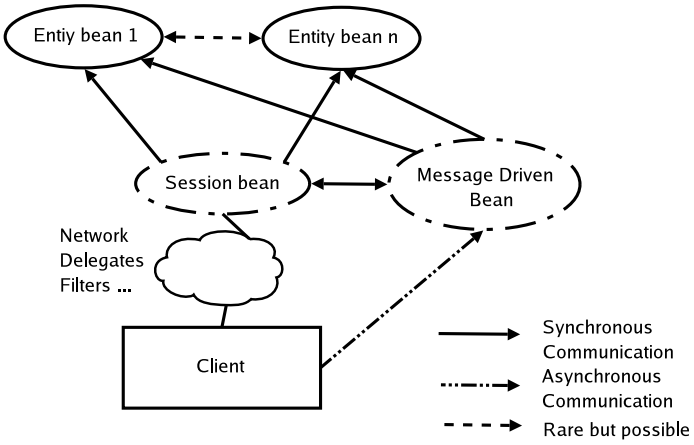

Figure 13: Elements of the EJB framework

4.2 Software Architecture

RFIDLocator contains more than a hundred classes. Thus the description of its architecture will only summarize the most important components[22].
The core of the application is built around the relatively small set of objects described in Subsection 3.2. These objects were all implemented as Entity Beans (aka Entities) and represent actual data in a storage medium (in the case of our application, the Entities represent tuples in a relational database). On the other hand, Session Beans (aka Sessions) contain the specific business logic of the application. RFIDLocator proposes several Session Beans offering business services. The place of the Session Beans in the overall EJB framework is shown on Figure 13. Basically, Session Beans are accessed by the client (often through a number of interfaces and proxies) to *solve a business task*. In turn, the Sessions are interacting with 1..n Entities to achieve their goal.
It is worth noting that most of the Session Beans composing the RFIDLocator implement the Session Façade Pattern [4]. That is: where the client could access the Entities directly, the Façades provide him with a unique entry point to gain access to many Entities at the same time.
The following paragraphs provide an overview of the most important Session and Message-Driven Beans as well as a summary of the services they expose.
User Manager. The UserManagerSessionBean offers methods related to the management of RFIDLocator's Users. As an example, the method registerUser() can be used to add a new User to the system.
Location Manager. This Session Bean is intended to offer methods regarding the places within the predefined area covered by the application. For instance, the method addLocation() provides a way of creating a new Location. The newly created Location is going to be part of the places RFIDLocator can monitor (provided a PhysicalAntenna is placed in this Location). It is primarily intended to be used by other Session Beans (such as the SensorManager SessionBean).
Traceable Object Manager. The TraceableObjectManager offers methods for managing the objects that can be traced by RFIDLocator, i.e. the TraceableObjects. It also provides a central method called locationHistory() which is in charge of returning the Locations a TraceableObject went through. This latter service is the core business of the RFIDLocator application as it permits the approximation of the current

[20]The book [34] extensively explains all the concepts needed to develop Enterprise JavaBeans. Besides, the article [40] provides already a good summary of the subject.
[21]RDBMS stands for Relational Database Management System.
[22]A more detailed view of the classes is available using the javadoc of the project at [22] or in [23].

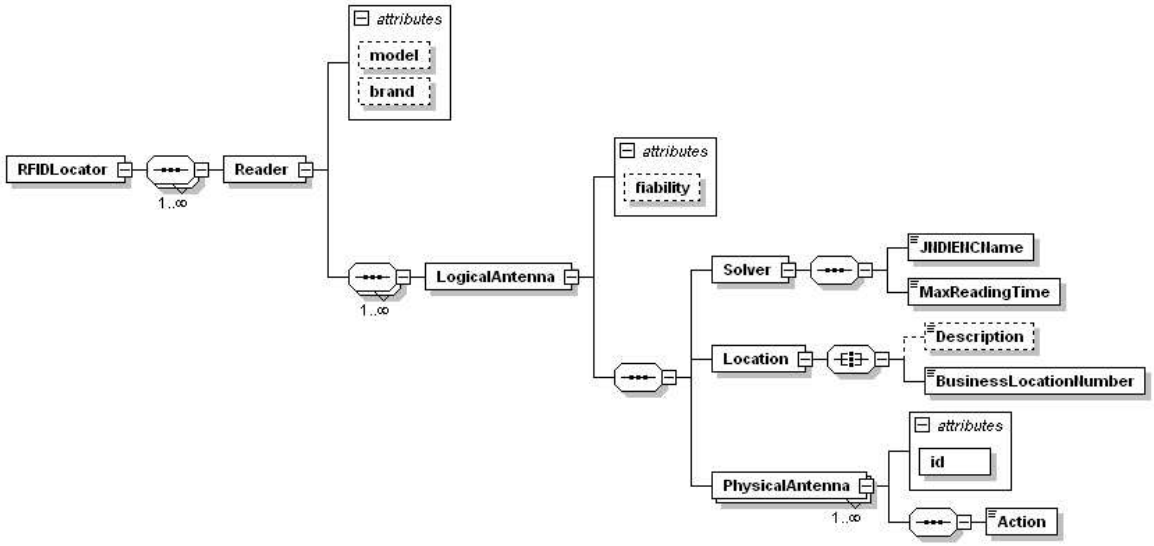

Figure 14: Diagram of the configuration formalism.

place an object is in.

PML Simulator Publisher. In order to test the system without the need of many (or even one) RFID readers, RFIDLocator is provided with a PMLSimulatorPublisher. This Session offers methods to simulate the sending of Java Message Service (JMS) PML events. To convey such messages it provides the publishPML(String PMLCoreString) method.

Observation Manager. The ObservationManager is in charge of persisting Observations using the method addLocatorObservation(). Note that an Observation has to go through various steps before being eventually persisted as a LocatorObservation. These steps are the matter of the Solvers described below.

Reader Manager. This Session Bean basically provides a method for parsing an XML file containing the settings of the environment in which RFIDLocator has to be deployed. Indeed, the method parseConfigString() takes an XML string as argument and builds an object graph containing the following elements: *(i)* Readers; *(ii)* LogicalAntennae; *(iii)* PhysicalAntennae; *(iv)* Solvers; *(v)* Locations. The syntax of the stream to be parsed is defined by an XML schema. Figure 14 exposes the chosen formalism which is further described in Section 5.7 of [23].

The transformation of the input string into a set of objects is achieved by using Sun's implementation of the Java Architecture for XML (Data) Binding (JAXB). This specification enables the conversion of an XML document into Java Objects in a very straightforward and elegant manner. In this particular context, the conversion is called *unmarshalling to Content Objects*. Figure 15 provides a schemed view of this process.

Eventually, after creating the Content Objects the ReaderManager persists them into the corresponding Entity Beans as described in the Object Model.

Sensor Listener Message Driven Bean. The SensorListener is the integration point between the Event Manager and the final application (see Figure 16). The events reported by the Event Manager go through several steps ending to a JMS Queue called the RFIDLocatorQueue. The SensorListener is a Message Driven Bean (MDB) listening to this latter Queue. That is, the SensorListener is a component being notified by the EJB container each time a message is posted on the RFIDLocatorQueue.

A message arrives at the MDB in the form of a PML string. The PML is then unmarshalled using the JAXB API. After this conversion, the Message Driven Bean does the first filtering by checking whether the

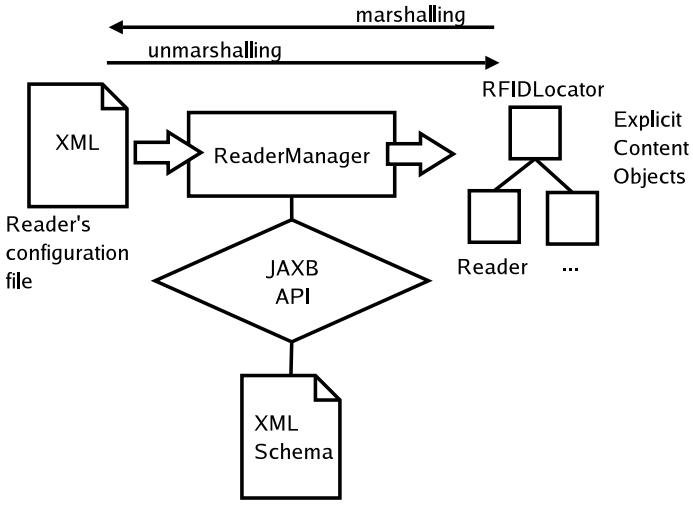

Figure 15: The marshalling/unmarshalling process of the Reader Manager.

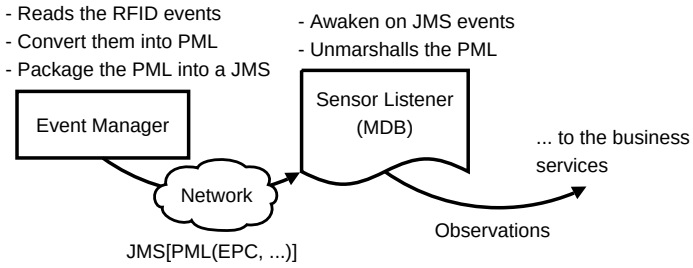

Figure 16: Asynchronous communication between the EM and the application.

PhysicalAntenna that made the Observation is registered within the RFIDLocator. If this turns out to be the case, the SensorListener contacts the corresponding LogicalAntenna and asks it to *solve the Observation*. That is: to decide what to do with the incoming Observation.
Such a Message Driven Bean is a very convenient way of listening to a JMS Queue. Indeed, the EJB container automates all the receiving/listening overhead which enables us to concentrate on the business logic of the asynchronous component.
The Solvers. The Solvers are Session Beans as well. In short, the Solvers implement the *algorithms* used by RFIDLocator to trace the position of the TraceableObjects. As said above when a message arrives at the JMS Queue of the application the Sensor Listener Message Driven Bean contacts the corresponding LogicalAntenna (see steps 1.2 and 1.3 of Figure 18) and asks it to solve the incoming Observations by invoking its solveObservation() method. To do so, the LogicalAntenna *passes the Observations to its Solver*. According to the algorithm it implements, a Solver has three possibilities when handling an Observation:

1. Persist the Observation as a LocatorObservation, which can be interpreted as a direct *validation* of the incoming Observation.

2. Buffer the Observation as a BufferedObservation in order to *wait* for some more information before actually taking a decision.

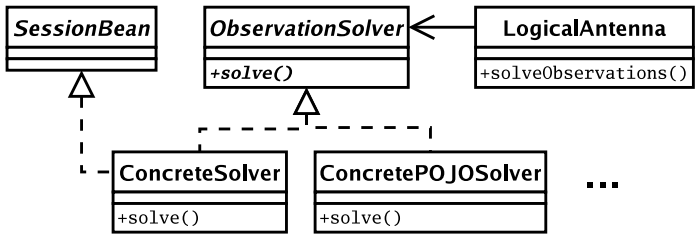

Figure 17: Logical antenna and observation solvers.

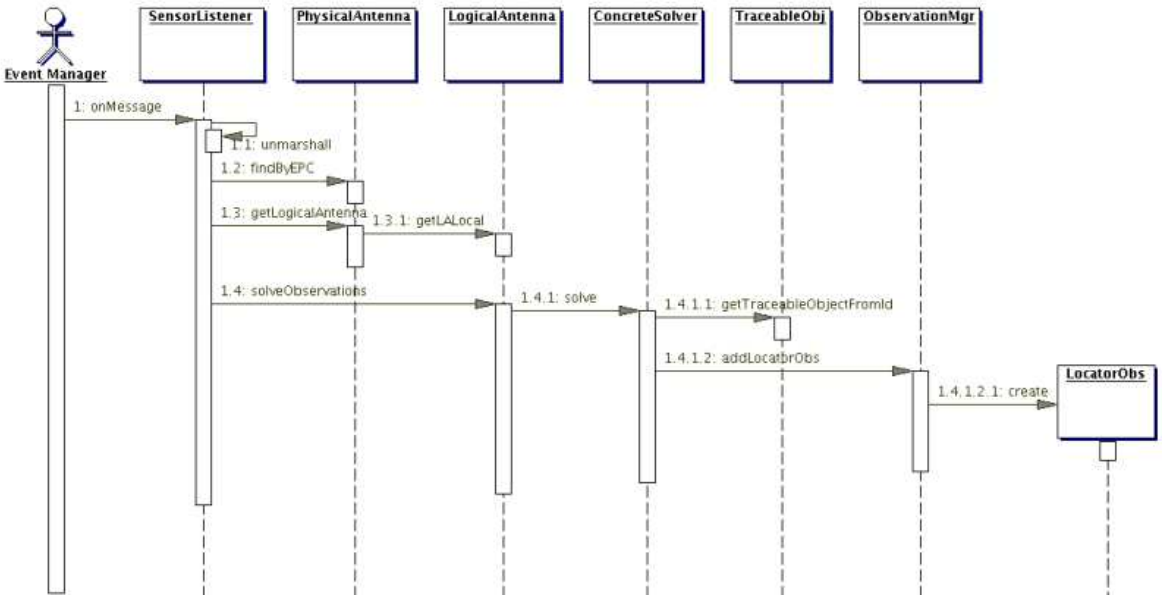

Figure 18: Sequence diagram of an Observation solved using a simple concrete solver.

3. *Discard* the Observation, which can be interpreted as declaring it to be invalid.

The number of Solvers one can imagine is not limited. To satisfy this criterion the Solvers are based on the modular architecture depicted on Figure 17.

To start with, they all implement the ObservationSolver interface. It defines a single method called solve(). This method is called by the LogicalAntenna using its concrete Solver when the validation of an Observation is required (see step 1.4.1 of Figure 18). Thus, to add a new Solver one just needs to implement it and attach it to the concerned LogicalAntennae.

Two Solvers were developed for this version of RFIDLocator. Both of them are implemented as Session Beans because of the high performances of these components as well as its convenient integration with the final application. However, developing them as Sessions is not mandatory and a Solver could as well be a POJO (Plain Old Java Object) implementing the ObservationSolver interface.

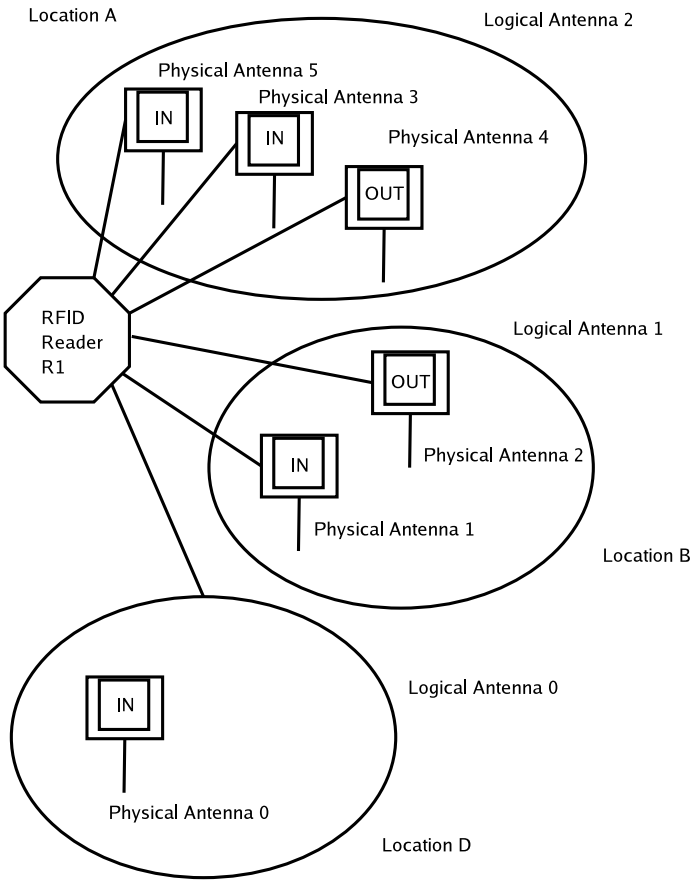

Figure 19: Schematic view of a reader's settings.

4.3 Solving Algorithms

As previously exposed, when an Observation arrives at the gate of the final application (that is, at the SensorListenerMessageDrivenBean) it has to be filtered according to a set of rules and eventually to be validated and persisted as a business observation (i.e. a LocatorObservation).
This is the role of the *Solver*. However, because both the use cases and sensors are heterogeneous, one need to be able to develop as many Solvers as required.
To emphasize the differences between the various Solvers let us provide some examples. Consider the reader R_1 of Figure 19. It shows a device to which three LogicalAntennae are attached. In turn, the LogicalAntennae contain up to three PhysicalAntennae with an Action (either IN or OUT) assigned to them. From the figure we can deduce that the Observation solving occurs at a LogicalAntenna level (e.g. LogicalAntenna 2 controls the Observations for Location A).
To begin, let us focus on the solving of an Observation that occurred at PhysicalAntenna 0. From the fact that the LogicalAntenna 0 contains only one PhysicalAntenna and thus can report only one type of Action it follows the simple validation rule:

- As long as the TraceableObject observed at PhysicalAntenna 0 is registered in the RFIDLocator, validate the Observation, i.e. persist it as a LocatorObservation (see Figure 18).

This rule is the core of the Solver for LogicalAntenna 0. It has been implemented in this version of the RFIDLocator and is called: SingleTypeConcreteSolver.
The case of the LogicalAntennae 1 and 2 needs more attention. Indeed, in both settings the LogicalAntennae can report IN *and* OUT Actions. The use of such antennae is not obvious at first. One could, for instance, think of separating the two PhysicalAntennae of the LogicalAntenna 1 into two distinct LogicalAntennae. As a result the SingleTypeConcreteSolver would do the solving job perfectly. However, coupling both PhysicalAntennae and creating a new Solver enables us to *capture the direction of the motion* without the need of a more sophisticated reader.
To achieve this goal, the algorithm of LogicalAntennae 1 and 2 must only follow this set of rules; for each incoming Observation:

1. Check if the TraceableObject is registered, if not: discard the Observation and quit the algorithm.
2. Scan the BufferedObservations to find at least one Observation:
   (a) With the Action *opposed* to the current one (i.e. $IN \rightarrow OUT$ or $OUT \rightarrow IN$).
   (b) Which has not expired in regards to the MaxReadingTime attribute of the LogicalAntenna.

   If none is found goto 3, else goto 4.
3. Buffer the Observation as a BufferedObservation and quit the algorithm.
4. Persist the Observation as a LocatorObservation. The Action attached to this latter is going to be the last observed Action.

Additionally, the buffer is cleaned before and after the solving, deleting BufferedObservations that are too old to be part of a valid motion. Hence, using such a Solver both LogicalAntennae 1 and 2 can *capture the direction of the motion*. This algorithm is implemented in the RFIDLocator by the DifferentActionsConcreteSolver.
To emphasize the idea of capturing motion consider the concrete example of Figure 20. On this picture two PhysicalAntennae can be seen. Using a configuration file, both are aggregated to form a single LogicalAntenna. Now, using the DifferentActionsConcreteSolver described above, these two PhysicalAntennae can report whether an object was going OUT or IN the office.

5 Conclusion

5.1 Achievements

RFIDLocator is a fully functional open-source[23] J2EE application allowing to locate and trace electronically tagged objects within a predefined area. RFID is the underlying technology used to achieve this goal. The development of such an application is not only of theoretical interest, as it reflects a need that many organizations or administrations formulated.
The adoption of the standards of the RFID field, as well as the use of established Java enterprise framework are prerequisites to build a performant, scalable, robust and reliable application. The clean, flexible and well-documented software architecture of the RFIDLocator allows interested people to extend it to fit their particular requirements.
Furthermore, this paper presents a complete example of a modern distributed application. Figure 21 summarizes the various hardware devices and communication technologies involved in the processing of the information.

[23]The source code is provided under the GNU GPL License [17] and can be freely downloaded from [21].

Figure 20: A concrete example of capturing the direction of the motion.

5.2 Future of RFID

Until now RFID's markets have remained niche ones, but the potential of the still young EPC Network might well boost the number of researchers and manufacturers around the technology. Moreover recent developments in digital technologies and networks have brought RFID tags to a price/performance ratio allowing cross organizations deployments. The price of tags and readers may be the key factor of large adoption and deployment of RFID for companies expecting a positive return on investment (ROI).

But one should not underestimate the fact that the customers also have to embrace this new technology. As revealed in recent surveys made in Europe and in the USA, only about half of the common people thinks the use of RFID is a good thing [37]. In order to overcome this barrier, privacy and security concerns have to be solved at both technological and legal levels.

If privacy concerns are not addressed, RFID technology may be misused. For instance, RFID passports require the data on the chip to be encrypted in order to prevent eavesdropping[24]. Such applications show that challenging cryptographic issues are raised in relation with wireless transmission [6].

Besides, there is a need for clear laws and recommendations about the tracking of goods, animals and even people [37]. This would restore confidence among the folk and permit governments to crack down when needed.

[24]The internationally renowned security technologist Bruce Schneier (see [48] and [49]) proposes an interesting discussion of the U.S. State Department proposal [58] for the new electronic passports.

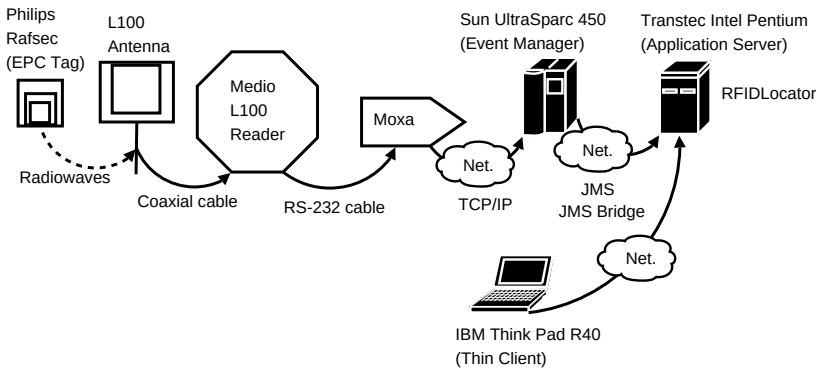

Figure 21: The hardware devices of our configuration.

Acknowledgments

The project was realized in the department of Informatics of the University of Fribourg in Switzerland and was possible thanks to its head, Prof. Jacques Pasquier-Rocha.
We are also thankful to Sun Microsystems for letting us freely test and use their Sun Java System RFID Software in our purely academic and non-commercial environment and specially to Marc Vidal-Alaiz for his support concerning special issues of the system's configuration.

References

[1] 3M. 3M RFID Tracking Solutions - Case Studies: Law Firm Finds Relief in Automated File Tracking. [online]. http://cms.3m.com/cms/US/en/2-115/kurRIFV/view.jhtml (accessed December 20, 2005).

[2] 3M. The 3M RFID Tracking System. [online]. http://cms.3m.com/cms/US/en/2-115/illzEY/view.jhtml (accessed December 20, 2005).

[3] R. Adams. BarCode 1: a quick tour FAQ. [online]. http://www.adams1.com/pub/russadam/barcode1.html (accessed December 15, 2005).

[4] D. Alur, J. Crupi, and D. Malks. *Core J2EE Patterns*. Sun Microsystems Press, second edition, 2003.

[5] Auto-ID Labs. [online]. http://www.autoidlabs.org/aboutthelabs.html (accessed December 15, 2005).

[6] G. Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, December 2005. [Retrieved December 23, 2005, from http://lasecwww.epfl.ch/~gavoine/download/papers/Avoine-2005-thesis.pdf].

[7] B. Brewin. EPCglobal taps VeriSign to operate RFID product code directory. [Retrieved December 23, 2005, from http://www.computerworld.com/mobiletopics/mobile/technology/story/0,10801,89036,00.html].

[8] A. Caffrey. Location tracking – for people, products, places – is fast coming into its own. *The Boston Globe*, October 2005. [Retrieved December 21, 2005, from http://www.boston.com/business/globe/articles/2005/10/10/location_tracking____for_people_products_places____is_fast_coming_into_its_own?mode=PF].

[9] J. Collins. IBM launches RFID middleware. *RFID Journal*, 2004. [Retrieved December 21, 2005, from http://www.rfidjournal.com/article/articleview/1291/1/1/].

[10] Microsoft .NET. [online]. http://www.microsoft.com/net (accessed December 21, 2005).

[11] M. Downs. Counterfeit Drugs: A Rising Public Health Problem. [Retrieved December 20, 2005, from http://www.webmd.com/content/article/95/103346.htm].

[12] EPCglobal. EPC Tag Data Standards Version 1.1 Rev.1.24. Technical report, EPCglobal, 2004.

[13] EPCglobal. EPCglobal Object Name Service (ONS) 1.0. Technical report, EPCglobal, 2005. EPCglobal Ratified Specification, Version of October 4, 2005.

[14] EPCglobal Inc. [online]. http://www.epcglobalinc.org/ (accessed December 15, 2005).

[15] Frequently Asked Questions about EPCglobal. [online]. http://www.epcglobalinc.org/about/faqs.html (accessed December 19, 2005).

[16] C. Floerkemeier, D. Anarkat, T. Osinski, and M. Harrison. PML Core Specification 1.0. Technical report, Auto-ID Center, 2003.

[17] Free Software Foundation Inc. GNU General Public License Version 2. [online], June 1991. http://www.gnu.org/copyleft/gpl.html (accessed December 23, 2005).

[18] Gartner. Hype cycle for emerging technologies 2005, August 2005. http://www.gartner.com/DisplayDocument?ref=g_search&id=484310.

[19] Apache Geronimo. [online]. http://geronimo.apache.org (accessed December 21, 2005).

[20] A. Goyal. Technical Report: Savant Guide. Technical report, Auto-ID Center, 2003. [Retrieved December 19, 2005, from http://www.autoidlabs.org/whitepapers/mit-autoid-tr015.pdf].

[21] D. Guinard. Radio Frequency IDentification: Evaluation of the Technology Supporting the Development of an Assets Tracking Application. [online]. http://diuf.unifr.ch/softeng/student-projects/completed/guinard/ (accessed December 23, 2005).

[22] D. Guinard. The RFIDLocator Web Application. [online]. http://diuf.unifr.ch/softeng/rfidlocator (accessed December 23, 2005).

[23] D. Guinard. Radio frequency identification: Evaluation of the technology supporting the development fo an assets tracking application. Bachelor thesis, Department of Informatics of the University of Fribourg, Switzerland, Sept. 2005. [Retrieved December 22, 2005, from http://diuf.unifr.ch/softeng/student-projects/completed/guinard/download/report_rfid_guinard_unifr.pdf].

[24] U. Hansmann, L. Merk, M. S., and N. T. Stober. *Pervasive Computing*. Springer, second edition, 2003.

[25] A. Hesseldahl. Master of the rfid universe. *Forbes.com*, 2004. [Retrieved December 23, 2005, from http://www.forbes.com/2004/06/29/cx_ah_0629rfid.html].

[26] IBM Software. WebSphere RFID Premises Server. [online]. http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/ (accessed December 21, 2005).

[27] Java 2 Platform, Enterprise Edition (J2EE). [online]. http://java.sun.com/j2ee (accessed December 21, 2005).

[28] JBoss Application Server. [online]. http://www.jboss.com/products/jbossas (accessed December 21, 2005).

[29] JOnAS: Java Open Application Server. [online]. http://jonas.objectweb.org (accessed December 21, 2005).

[30] L. Kontnik. Counterfeits: The Cost of Combat - Fake prescription drugs are a growing problem. What can pharma do? [Retrieved December 20, 2005, from http://www.pharmexec.com/pharmexec/article/articleDetail.jsp?id=75674].

[31] S. Kuchinskas. Keeping Drugs Legit With RFID. [Retrieved December 20, 2005, from http://www.ccmsectorinvest.com/detailednews.asp?intReleaseID=8458&d=11∗2005&n=36].

[32] O. Liechti. RFID: Middleware et intégration avec le système d'information. In *Radio Frequency IDentification: Applications, Software Tools and Future Vision*. Department of Informatics, University of Fribourg, Switzerland, December 2005. [Retrieved December 23, 2005, from http://diuf.unifr.ch/softeng/rfid/download/rfid_fribourg_liechti.pdf].

[33] F. Michahelles. RFID: Bridging the gap between the virtual and the real world. In *Radio Frequency IDentification: Applications, Software Tools and Future Vision*. Department of Informatics, University of Fribourg, Switzerland, December 2005. [Retrieved December 20, 2005, from http://diuf.unifr.ch/softeng/rfid/download/rfid_fribourg_michahelles.pdf].

[34] R. Monson-Haefel. *Enterprise JavaBeans*. O'Reilly, fourth edition, 2004.

[35] MySQL: The world's most popular Open Source Database. [online]. http://dev.mysql.com/ (accessed December 21, 2005).

[36] NJE Consulting. NJE Consulting: Canada RFID, Data-Collection System Integrator. [online]. http://www.nje.ca/ (accessed December 23, 2005).

[37] M. C. O'Connor. Survey Reveal Dubious Consumers. *RFID Journal*, 2005. [Retrieved July 26, 2005, from http://www.rfidjournal.com/article/articleview/1409/-1/1/].

[38] Oracle Database. [online]. http://www.oracle.com/database/index.html (accessed December 21, 2005).

[39] Oracle. Oracle RFID and Sensor-Based Services. [online]. http://www.oracle.com/technologies/rfid/ (accessed December 21, 2005).

[40] E. Ort. Ease of Development in Enterprise JavaBeans Technology. *Sun Developer Network (SDN)*, 2004. [Retrieved December 21, 2005, from http://java.sun.com/developer/technicalArticles/ebeans/ejbease/].

[41] PostGreSQL: The world's most advanced open source database. [online]. http://www.postgresql.org/ (accessed December 21, 2005).

[42] RFID Gazette. RFid Gazette (Radio Frequency Identification news and commentary): Asset Tracking. [online]. http://www.rfidgazette.org/asset_tracking/ (accessed December 20, 2005).

[43] RFID Gazette. RFid Gazette (Radio Frequency Identification news and commentary): Healthcare. [online]. http://www.rfidgazette.org/healthcare/ (accessed December 20, 2005).

[44] RFID Journal. VeriSign to Run EPC Directory. [Retrieved December 23, 2005, from http://www.rfidjournal.com/article/articleview/735/1/1/].

[45] A. Sabetti. Applications of Radio Frequency Identification (RFID). [Retrieved December 20, 2005, from http://www.aimglobal.org/technologies/rfid/resources/papers/applicationsofrfid.htm].

[46] SAP. SAP RFID: Making Adaptive Enterprise Vision a Reality. [online]. http://www.sap.com/solutions/business-suite/scm/rfid/index.epx (accessed December 21, 2005).

[47] Sun Java System Application Server. [online]. http://www.sun.com/software/products/appsrvr/index.xml (accessed December 21, 2005).

[48] B. Schneier. Schneier on Security: RFID Passport Security. [Retrieved December 22, 2005, from http://www.schneier.com/blog/archives/2005/04/rfid_passport_s.html].

[49] B. Schneier. Schneier on Security: RFID Passport Security Revisited. [Retrieved December 22, 2005, from http://www.schneier.com/blog/archives/2005/08/rfid_passport_s_1.html].

[50] S. Shepard. *RFID Radio Frequency Identification*. McGraw-Hill Professional, 2004.

[51] F. Siegemund, C. Floerkemeier, and H. Vogt. The Value of Handhelds in Smart Environments. In *Organic and Pervasive Comuputing - ARCS 2004*, pages 291–308. Springer, 2004.

[52] Sokymat. [online]. http://www.sokymat.com/ (accessed December 21, 2005).

[53] T. Staake, F. Thiesse, and E. Fleisch. Extending the epc network – the potential of RFID in anti-counterfeiting. In H. Haddad, L. Liebrock, A. Omicini, and R. Wainwright, editors, *Symposium on Applied Computing – SAC*, pages 1607–1612, Santa Fe, New Mexico, USA, March 2005. ACM, ACM Press.

[54] Sun Microsystem. Sun Java System RFID Software 2.0 Installation Guide. [Retrieved December 23, 2005, from http://docs.sun.com/app/docs/doc/819-1696?q=rfid].

[55] Sun Microsystems. Software Solutions - EPC and RFID. [Retrieved December 21, 2005, from http://www.sun.com/software/solutions/rfid/].

[56] VeriSign. VeriSign Selected to Operate Root Directory for EPCglobal Network. [Retrieved December 23, 2005, from http://www.verisign.com/printablePages/page_000846.html].

[57] World Wide Web Consortium: Naming and Addressing. [online]. http://www.w3.org/Addressing/ (accessed December 15, 2005).

[58] R. Yu. Electronic passports set to thwart forgers. *USA Today*, September 2005. [Retrieved December 22, 2005, from http://www.usatoday.com/travel/news/2005-08-08-electronic-passports_x.htm].