High-Level System Support for Automatic-Identification Applications

Matthias Lampe and Christian Floerkemeier

Institute for Pervasive Computing, ETH Zurich, CH-8092 Zurich, Switzerland
{lampe, floerkem}@inf.ethz.ch

Abstract. RFID systems have begun to find greater use in tagging consumer goods, in industrial automation, and in mobile asset and supply chain management as an enabling technology for smart objects. This introduces the need for software systems that manage RFID readers, filter and aggregate captured RFID data, combine and enrich the RFID data with application logic, and generate appropriate business events. We concentrate on the higher levels of a smart object system and address several challenges that application developers face when implementing Auto-ID applications. Derived from a requirements analysis, we propose an object model based on a symbolic location model. We also present a prototypical implementation of the model, the Object Monitoring System (OMS) that includes object persistence, query capabilities, and business event generation and present a case study as a proof-of-concept. We argue that our approach offers appropriate object and programming models that abstract from RFID specific details and provide the necessary services and the adequate level of reuse to facilitate the development of Auto-ID applications.

Keywords. Radio Frequency Identification (RFID), RFID middleware, Smart Objects

1 Introduction

Radio Frequency Identification (RFID) technology has recently seen growing interest not just from the research community, but also from a wide range of industries. RFID systems have begun to find greater use in the consumer object identification market, in industrial automation, and in mobile asset and supply chain management. Applications using RFID are often labeled automatic identification (Auto-ID) applications or smart object applications. The more general term Auto-ID also includes other identification technologies such as visual codes and other technologies which can be used to automatically identify objects (e.g., Bluetooth or Wireless LAN). In addition to Auto-ID technologies as the main enabling technologies for smart objects, many smart object applications also rely on sensors to track the object state in the physical world (e.g., temperature sensors to ensure quality of perishable goods).

In traditional RFID applications, such as access control, there was little need for an RFID middleware because the RFID readers were not networked and the RFID data were only consumed by a single application. In novel application domains (e.g., supply chain management), many readers capture RFID data that need to be disseminated to a variety of applications. This introduces the need for software systems that, on the lower levels towards the readers, manage readers, filter and aggregate captured RFID data, and on the higher levels towards the application, combine and enrich the RFID data with application logic, and generate appropriate business events, by providing a consistent object model to the application.

In this paper, we concentrate on the higher levels of a smart object system or RFID middleware and address the following challenges that application developer face when implementing Auto-ID applications: Instead of concentrating on the application or business logic they have to deal with RFID specific details and have to duplicate code when developing a new application for providing persistent storage and querying capabilities, and for generating business events such as the arriving of a shipment. We argue that our approach provides

Derived from a requirements analysis in section 2, we propose an Auto-ID object model that represents physical objects and is based on a symbolic location model (see section 3). Additionally, we briefly describe a prototypical implementation of the model, the Object Monitoring System (OMS) and present the case study of a retail store application as a proof-of-concept in section 4. After pointing out the differences of our approach to related work in section 5, we argue in the conclusion that our approach offers appropriate object and programming models that abstract from RFID specific details and provide the necessary services and the adequate level of reuse to facilitate the development of Auto-ID applications (see section 6).

2 Requirements

Based on an analysis of different smart object applications including the above and the study of other work on RFID middleware [1-5], we identified several application requirements an RFID middleware or smart object system should meet. For the scope of this paper we concentrate on the requirements for the higher levels of an RFID middleware, where RFID data is stored persistently, interpreted using context information and higher level application events are generated. Requirements dealing with RFID reader management, data filtering and aggregation and data dissemination can be found in [3]. Requirements concerning privacy are discussed in great detail in [6-8] and are not part of the scope of this paper. Other application requirements that relate to security, scalability and performance are not discussed here in detail, since they are not unique to RFID middleware design.

Persistency of object data. The data received by RFID readers and sensors has to be stored persistently to allow applications to query the data. For example, an application that wants to retrieve the track and trace information about a certain chemical from production to a warehouse needs to query all related RFID and sensor information.

RFID data interpretation. From an application perspective, it is also desirable to provide a mechanism that interprets the captured RFID data in a given, specific business context and that turns low-level RFID events into the corresponding business events. For example, the detection of a number of tags at a dock door over a certain time span should be automatically translated into a "shipment arrived" event.

Object data enrichment. In many cases, identified objects can be enriched with additional related data. This could, for example, be information about the color or size of a product, an expiry date for a food product or the number of usages of a tool. The additional object data can be retrieved from a database or even stored on the RFID tag attached to the object.

Location information. The system has to provide a mechanism to enrich RFID and sensor data with location information since applications are interested when a certain object was at a certain location. For many applications, symbolic location names are sufficient in the business context definitions. For example, the query of a track and trace application should return a list of times, locations and object states for a given object.

Object relationships. Many applications are interested in information related to several objects in a certain relationship. One important relationship is the neighboring objects relationship, i.e. objects are neighbors if they are together at the same location. A chemical monitoring application might, for example, be notified when containers with hazardous chemicals are stored together in a room.

Sensor support. In many applications it is not sufficient to only identify objects. The current state of the objects in the physical world has also to be detected. For a cool chain monitoring application it is crucial to also observe temperature data along the chain. The system has thus to provide the means to integrate sensors such as temperature, humidity or shock sensors and make their data accessible by the applications.

Actuator support. In addition to sensors, applications often have to quickly interact with the physical world using different kinds of actuators such as locks or even simple traffic lights to signal an application state to an operator.

3 Object Monitoring System

The Object Monitoring System (OMS) is the prototypical implementation of our proposed Auto-ID object model including a mechanism to define and generate business events. We argue that the object model, the business event generation and additional OMS services address the requirements presented in the previous section and that they provide the application developer with appropriate abstractions and a framework to automatically enrich the RFID information with business logic and facilitate the implementation of Auto-ID applications.

3.1 Object Model

The Auto-ID object model (see Fig. 1) is based on a symbolic location model, in which physical objects also define locations. The term object stands for the entity in

the model that represents physical objects in the real world (e.g., the object Fork-Lifter12 in an object model instance represents the fork lifter #12 in the physical world). *Objects* can have *properties* and *functions*. These three entities are the concepts with which the application and the business event generation are interfacing. *Auto-ID readers* that identify objects can also act as *property sources* that set the values of object properties. As *sensors* are the input channels from the physical world that also act as property sources, *actuators* are the output channels into the physical world that perform the functions.

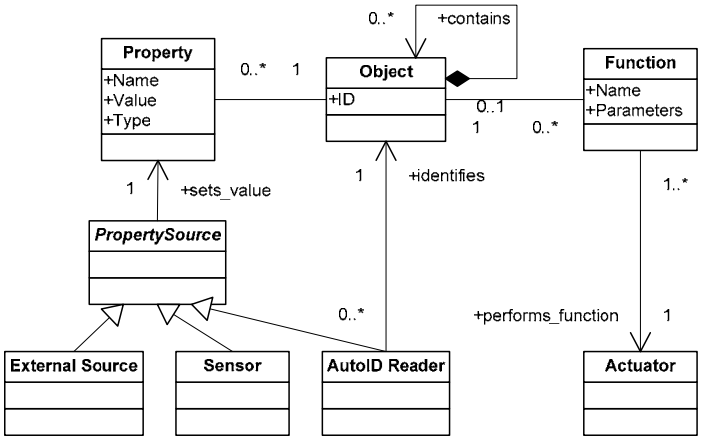

Fig. 1. Object Model of the OMS

Object. Objects play the central part in the model and are identified by a unique identification, for example, the electronic product code (EPC) as specified by EPCglobal. We do not differentiate between objects and locations. In our model, an object implicitly defines a location and can contain other objects. In an object model instance, the containment relationships between all the objects are represented as an object tree (see Fig. 4) in which an object can have zero to n child objects. Auto-ID sensors which have to be linked to corresponding objects automatically identify objects that are contained in the location defined by their corresponding object. For example, an RFID reader mounted on a fork lifter monitors the palette and its content the fork lifter is carrying. In the model instance, the detected objects will be contained in the fork lifter object. By itself the fork lifter might be detected by a dock door reader of the warehouse incoming goods placing the object fork lifter with its containing objects in the model instance as contained by the object warehouse.

Property. Objects may have static or dynamic properties that further describe an object and its state (e.g., temperature, expiry date or usage counts). Properties in our model simply consist of name, value and type. They are introduced to abstract from hardware dependencies to access sensor information, data stored on RFID tag memory or external databases, which all act as property sources that set the values of object properties. For example, the property temperature of the object cool box is set through a temperature sensor in the box, whereas the property usage counts of the

object hydraulic lift is stored on the RFID tag of the object and accessed through the Auto-ID reader.

Function. Functions of objects are defined by a name and a list of parameters. They act as the abstraction for actuator control (e.g., locks or light signals) and can be executed as actions if certain business events are generated. They thus offer fast interaction with the physical world. For example, in a smart medical shelf application, the OMS detects the badge of the person accessing it and only unlocks the doors via the lock actuator if the person is authorized.

3.2 Business Events

The dynamic changes in the object model correspond to the changes in the physical world. Many applications are only interested in exceptional states related to objects and their properties such as the completion of an incoming shipment, reached expiry dates of products in a shelf, or the temperature inside a cool box that exceeds a threshold. We present a state machine-based model implemented in the OMS to easily formulate subscriptions for business events that abstract from the object observations in order to limit notification of an application to the exceptional states.

The analysis of Auto-ID applications has shown that an event mechanism based simply on conditions of object and property configurations is not sufficient to describe many exceptional states. The decision if an exceptional state occurred often depends on a related state that has happened a certain time before the state in question. This dependency of states over time can very well be expressed and modeled using state machines. In addition, state machines express a kind of business process flow that is well known in industry and state machines can be modeled using available design and modeling tools.

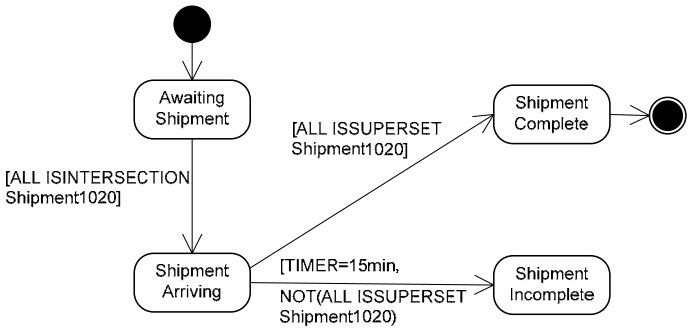

Fig. 2. State machine to monitor the incoming shipment #1020

In our state machine model, state transitions are coupled to certain configuration of objects and their properties. A transition happens if the condition describing the transition is true. Conditions can involve presence or absence of a set or number of objects or object properties and time constraints. It is also possible to logically combine different conditions to describe a transition. Business events to the application can be

sent when a state is entered or left. The conditions are defined using an OMS specific condition language that allows access to the object model and provides several operators to formulate the logical conditions.

For example, Fig. 2 illustrates the state machine to monitor the state of the incoming shipment #1020. The state machine is connected to the object Backroom in the object model instance illustrated in Fig. 4. All the products of the expected shipment are defined in the object set Shipment1020. The *Shipment Arriving* state is reached when at least one product of the expected shipment is detected at the incoming goods door entering the backroom (i.e., the set of ALL child objects of Backroom and the set Shipment1020 intersect). If all products of the shipment have been detected the *shipment complete* state is reached (i.e., the set of ALL child objects is a super set of Shipment1020 as other objects can reside in the backroom). If however after a given duration (in our case 15 minutes) some products are still missing, the *shipment incomplete* state is reached. All reached states generate corresponding business events.

3.3 OMS Implementation

We prototypically implemented the Auto-ID object model and the business event mechanism in the OMS which provides persistency of the object model (i.e. object history), query capabilities and services to access and modify the object model and the business event subscriptions.

The OMS is implemented using J2SE. Messaging with the application clients is handled using XML and TCP or HTTP message-transfer-bindings. The current object model instance is kept in an in-memory data structure to allow fast access and fast evaluation of state transition conditions. The object history containing all past object data including properties and object containment relationships is implemented using relational database tables similar to the approach of [9] which are stored in a MySQL database. The OMS supports two kinds of queries: First, get all locations of a given object (i.e. all the parent objects) over a certain time, and second, get all objects situated at a given location (i.e. the child objects) over a certain time.

In a common deployment the OMS is supported by low-level Auto-ID and sensor middleware that provides the preprocessed RFID and sensor data for the object model. Currently the OMS supports the Accada Reader and Accada Filtering and Collection components [10].

4 Case Study: Retail Store

The retail store demo application we implemented is part of a larger retail supply chain and demonstrates the capabilities of the OMS by monitoring the flow of goods and their states in the store. The store in our example consists of four different main locations that all contain further sub locations. As described in section 3, all locations are modeled as objects (see Fig. 4). Customers enter the store through the entrance area and take a shopping cart. In the sales area, they pickup goods from the shelves and they pay at the point-of-sales (POS). Hidden from the customer is the backroom where incoming goods are stored. We implemented the following locations using

RFID readers operating at 13.56 MHz from Feig in different set-ups and tagged sample products using ISO15693 RFID tags (see Fig. 3): (a) At the incoming goods department two readers monitor the door through which products arrive; (b) one RFID reader monitors the products stored in the backroom; (c) a shelf with 8 compartments is observed by a reader with a multiplexer and 8 antennas; and (d) a POS is monitored by one reader. The other locations in our application can be simulated using the visual or batch RFID reader simulator of the Accada middleware [10]. The data of the Feig readers is captured, filtered and aggregated using Accada Reader components that in the case of the incoming goods location feed their data to the Accada Filtering and Collection component, which aggregates the two readers to one logical reader for the OMS. The data from all locations is then fed to the OMS where it is processed according to the predefined object model instance (see Fig. 4).

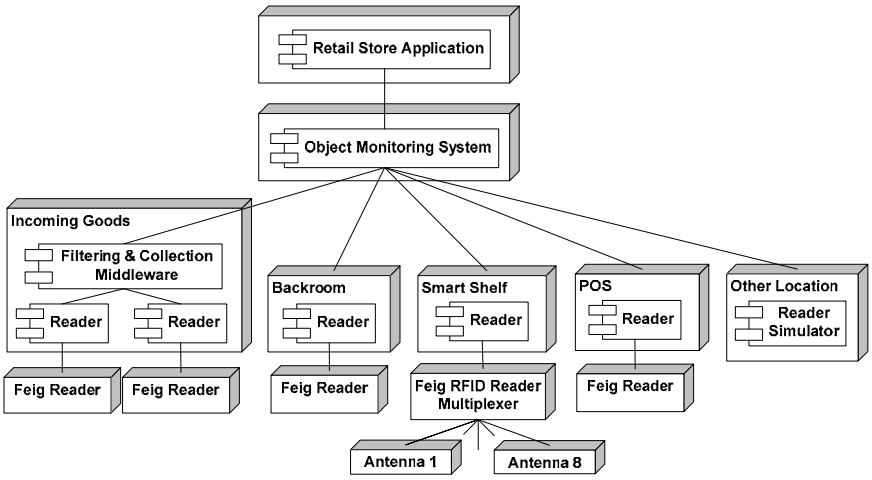

Fig. 3. Deployment of the demo retail store application

We defined several state machines which generate business events when important states are reached: A state machine for the smart shelf stays in the *shelf full* state as long as the number of each product class in the shelf stays above a certain threshold. If the number of a product class drops below the threshold and stays there for a certain duration the *replenishment needed* state is reached. In a similar way the expiry date of the products in the shelf are checked. In addition, state machines to monitor incoming shipments as described in the example in section 3.2 are included. Other state machines include temperature monitoring of the cooled backroom, and the allergen check for products in the shopping cart of a customer.

The retail store demo application itself consists of several components that react to the business event notification of the OMS and that provide user feedback and visualize the flow of goods in the store: (a) a window shows the flow of incoming goods the state of arriving shipments; (b) a backroom window lists the current inventory; (c) the smart shelf components sends messages to the PDA or cell-phone of a staff to notify about replenishments or nearly expired goods; and (d) a POS component simulates a

self-checkout. In addition, the whole object model can be dynamically visualized using an administration component of the OMS.

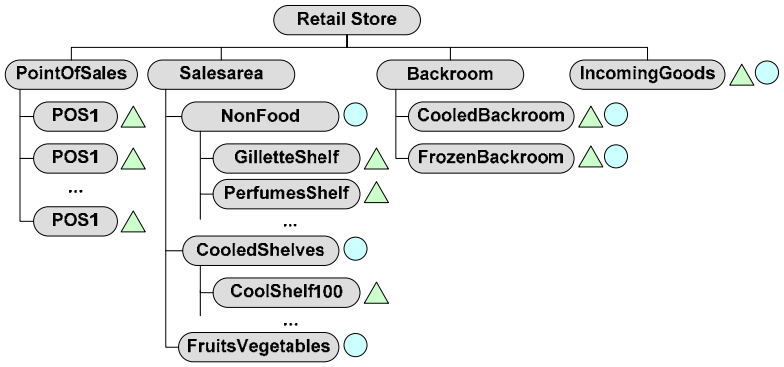

Fig. 4. The predefined instance of the Auto-ID object model of the retail store application (triangles denote connected RFID readers or sensors, circles denote related business event definitions)

In addition to the retail store application, we implemented a smart medicine shelf, a smart toolbox and automated tool inventory [11], and the Augmented Knight's Castle [12] based on the OMS and the Accada middleware.

5 Related Work

The need for an RFID infrastructure and the application requirements towards such an RFID infrastructure have been discussed in a number of publications [1, 3, 5]. However, they focus mainly on RFID data capture and filtering and aggregation of the data, and not so much on application level support.

Initially, the concept of a distributed networking infrastructure for RFID was proposed by the Auto-ID Center, an industry-sponsored research program to foster RFID adoption [13], which coined the term EPC Network [14]. In the EPC Network, the EPC Information Services (EPCIS) provides RFID data persistence and the possibility to enrich RFID data with application specific business information including location information. However, the EPCIS is merely a repository of business events which can not be generated automatically by predefined rules. Event generation has to be performed in the EPC Capturing Application which has to be implemented by an application developer. The EPC Network does not provide any generic EPC Capturing Application component. In addition, at the current state of specification, the EPC Network only focuses on RFID and does not support sensor integration.

The Auto-ID Node and the Auto-ID Repository components of the SAP Auto-ID Infrastructure (AII) [1] also provide object persistence. To certain extend, RFID data can be enriched with business context and structured using location information by predefined rules. However, these rules only define how the data is stored in the repository. The Auto-ID Node does not provide any event generation on the enriched

and structured data. It also does not provide a consistent object model to which all of the stored data has to comply.

There are a number of other commercial and non-commercial RFID middleware products available, among others [9, 15-19]. Most of them are database-centric approaches that extend existing application servers and mainly provide support for RFID data filtering, aggregation and simple data enrichment. Very little generic application-level support is offered.

6 Conclusion

We presented the Auto-ID object model as an appropriate abstraction of underlying Auto-ID and sensor data access and processing. Merging the two concepts of physical objects and location into one greatly simplifies an instance of the object model since the application only has to define and manage its objects. Many of these definitions happen automatically since objects carry their IDs which are obtained automatically by Auto-ID readers of the parent object. Only static objects and their relationships, for example, buildings or rooms have to be defined once for a model instance. In addition, we introduced a state machine-based model to easily formulate subscriptions for business events that abstract from the object observations in order to limit notification of an application to the exceptional cases. As a proof-of-concept of our approach, we prototypically implemented the OMS and deployed and instantiated it in a retail store scenario.

Many business processes can successfully be modeled with the state machine-based event subscription. However, the model has its limits if conditions and dependency get too complex. In this case, the state machine definitions would get rather complicated and its value for application developers would be diminished. In this case, only elementary parts of the process would be modeled in the OMS. The other part of the process is then handled in the application that reacts to the business events of the OMS. We currently implement another tool-based approach in which business processes and events are defined at a higher level of abstraction using a visual toolkit which then generates the OMS state machine definitions automatically.

In order to find the right balance between a simple but not very powerful and a powerful but too complex model we are in the process of implementing more smart object and Auto-ID applications using the presented concepts and the OMS.

References

[1] C. Bornhoevd, T. Lin, S. Haller, and J. Schaper, "Integrating Automatic Data Acquisition with Business Processes - Experiences with SAP's Auto-ID Infrastructure," presented at 30st international conference on very large data bases (VLDB), Toronto, Canada, 2004.

[2] S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma, "Managing RFID Data," presented at 30st international conference on very large data bases (VLDB), Toronto, Canada, 2004.

[3] C. Floerkemeier and M. Lampe, "RFID middleware design - addressing application requirements and RFID constraints," presented at Smart Objects Conference (SOC), Grenoble, France, 2005.
[4] K. Römer, T. Schoch, F. Mattern, and T. Dübendorfer, "Smart Identification Frameworks for Ubiquitous Computing Applications," *Wireless Networks*, vol. 10, pp. 689-700, 2004.
[5] S. Sarma, "Integrating RFID," *ACM Queue*, vol. 2, pp. 50-57, 2004.
[6] S. Garfinkel and B. Rosenberg, "RFID: Applications, Security, and Privacy," Addison-Wesley, 2005.
[7] M. Langheinrich, "Personal Privacy in Ubiquitous Computing - Tools and System Support." Zurich, Switzerland: ETH Zurich, 2005.
[8] M. Langheinrich, "RFID and Privacy," in *Security, Privacy and Trust in Modern Data Management*, M. Petkovic and W. Jonker, Eds.: Springer-Verlag, 2006.
[9] F. Wang and P. Liu, "Temporal Management of RFID Data," presented at 31st VLDB Conference, Trondheim, Norway, 2005.
[10] C. Floerkemeier, M. Lampe, and C. Roduner, "Accada EPC Network Platform," 2006, www.accada.org.
[11] M. Lampe, M. Strassner, and E. Fleisch, "A Ubiquitous Computing Environment for Aircraft Maintenance," presented at ACM Symposium on Applied Computing, Nicosia, Cyprus, March, 2004.
[12] M. Lampe, S. Hinske, and S. Brockmann, "Mobile Device based Interaction Patterns in Augmented Toy Environments," presented at Third International Workshop on Pervasive Gaming Applications, PerGames 2006, Dublin, Ireland, May 2006.
[13] S. Sarma, D. L. Brock, and K. Ashton, "The Networked Physical World - Proposals for Engineering The Next Generation of Computing, Commerce & Automatic Identification," 2000, www.autoidcenter.org/research/MIT-AUTOID-WH-001.pdf.
[14] "The EPCglobal Architecture Framework," Architecture Review Committee. EPCglobal Jul. 2005.
[15] "Oracle Sensor Edge Server," http://www.oracle.com/technology/products/iaswe/edge_server.
[16] "WebSphere RFID Premises Server," Dec. 2004, http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/.
[17] A. Gupta and M. Srivastava, "Developing Auto-ID Solutions using Sun Java System RFID Software," Oct. 2005, http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html.
[18] OATsystems, "OAT C4 Architecture," 2006, www.oatsystems.com.
[19] B. S. Prabhu, X. Su, H. Ramamurthy, C.-C. Chu, and R. Gadh, "WinRFID: A Middleware for the Enablement of Radiofrequency Identification (RFID)-Based Applications," in *Mobile, Wireless, and Sensor Networks: Technology, Applications, and Future Directions*, R. Shorey, A. L. Ananda, M. C. Chan, and W. T. Ooi, Eds.: John Wiley & Sons, Inc., 2006, pp. 331-336.