Reference Architecture for Event-Driven RFID Applications

Jürgen Dunkel and Ralf Bruns

Hannover University of Applied Sciences and Arts, Department of Computer Science
Ricklinger Stadtweg 120, 30459 Hannover, Germany

Abstract. RFID technology is usually not well integrated with business processes and backend enterprise information systems. Current enterprise information systems cannot deal with high volume of events and their complex dependencies. In this paper we propose a reference architecture for event-driven RFID applications. The key concept of the approach is complex event processing as the model for processing and managing of event-driven information systems. In order to prove the practical usefulness of the reference architecture we present a case study for tracking samples in large research laboratories using RFID.

1 Introduction

Radio Frequency Identification (RFID) has established itself as a significant technology in the automation of business processes. In recent years, RFID technology has been applied to a wide range of application areas such as supply chain management, health services, transportation, environmental systems [5].

One major obstacle for the success of RFID systems is how to process and manage RFID data and integrate it to drive the business processes. Current enterprise information systems cannot deal with events in order to drive the business processes and, consequently, are not sufficient for RFID-based applications. Due to the high volume of events and their complex dependencies it is not possible to have a fixed/predefined process flow on the business level. In recent years, event-driven architectures (EDA) have been proposed as a new architectural paradigm for event-based applications. In this paper we propose a reference architecture for event-driven RFID applications. The key concept of the approach is to use complex event processing (CEP) as the model for processing and managing of event-driven information systems. The reference architecture enables the seamless integration of RFID event streams in upper-level business processes in real-time. In order to prove the usefulness of our approach we selected the tracking of samples in large research laboratories using RFID.

The remainder of this paper is structured as follows. In the next section the basic concepts of event-driven architectures and complex event processing are introduced. Section 3 presents a reference architecture for handling events in RFID-based applications, allowing to integrate arbitrary RFID event sources with enterprise information systems. Laboratory logistics is selected as the application scenario for our approach, which is discussed in section 4. Finally, the last section summarizes the most

Dunkel J. and Bruns R. (2008).
Reference Architecture for Event-Driven RFID Applications.
In *Proceedings of the 2nd International Workshop on RFID Technology - Concepts, Applications, Challenges*, pages 129-135
DOI: 10.5220/0001729001290135

significant features of the approach and provides a brief outlook of future lines of research.

2 Event-Driven Architecture Basics

Event-driven architecture (EDA) provides an architectural concept for dealing with complex event streams. Complex Event Processing (CEP) is an event processing model to cope with a huge number of events [10]. One main idea is that events are not independent from each other, but correlated. The goal of CEP is to identify in a huge event cloud those patterns of events which are significant for a business domain. An example is a stock trading system which analyzes millions of tick data to discover patterns which signify buy or sell signals for certain shares. Pattern Matching and Event Processing are performed by so-called event-processing agents (EPAs), which monitor the stream of occurred events analyzing it for event patterns. Essentially they filter, split, aggregate, transform and enrich event – tasks, which are also well-known from application integration [4]. Furthermore, new complex events can be synthesized from simple events. This step takes correlations between occurred events into account and provides the real power of complex event processing. For instance, technical RFID events must be synthesized into meaningful business events A Complex Event Processing (CEP) system consists of the following building blocks:

Event Metadata. To automate event processing, a formalism based on metadata is required, which precisely specifies all possible events and their interdependencies. Each event contains general metadata: e.g. event ID, event type, event timestamp and event-specific information e.g. the ID of a RFID reader. Event Specifications are the base for event transformation, event generators and event subscribing.

Event Processing Rules define correlations between events for detecting event patterns, and determining corresponding actions. They can be expressed as ECA rules [17], [12] or as a SQL-based approach of querying event streams [1], [3]. Event Metadata is specified by appropriate event processing languages (EPLs) [12].

Event Processing. An *Event Processing Engine* evaluates and executes event processing rules. The *Event Data* manifests the occurred event and is a set of instances related to the event specification. Each event must be described in standard format, e.g. by an XML document. Because pattern matching takes also historical events into account, events must be permanently stored. Usually, not all events can be stored permanently, because the number of events can be nearly infinite. A practical solution is regarding so-called sliding windows, i.e. storing only the events occurred in a certain time period. It means that events have a certain lease and are deleted if it is expired. Of course, the appropriate lease times are domain and event-type specific. The more abstract or business oriented an event is, the longer might be its lease time.

Enterprise Integration Backbone. The enterprise integration backbone provides the middleware to access the backend systems. It provides event channels, event trans-

port, event subscription and is a mediator for accessing the enterprises backend systems. The enterprise integration backbone is used by the business components to propagate domain-specific events and by the rule engine to trigger event-driven actions provided by the backend systems.

3 Reference Architecture

The major objective of our approach is the development of a reference architecture for the integration of RFID events into existing enterprise information systems. The reference architecture is agent-based and should serve as an idealistic model of a class of event-driven architectures (EDA). Figure 1 provides an overview of the proposed reference architecture. On the top layer, the RFID sensors generate a continuous stream of RFID data event and serve as the event source. Moreover, the event stream can contain further types of events originating from other systems or event sources.

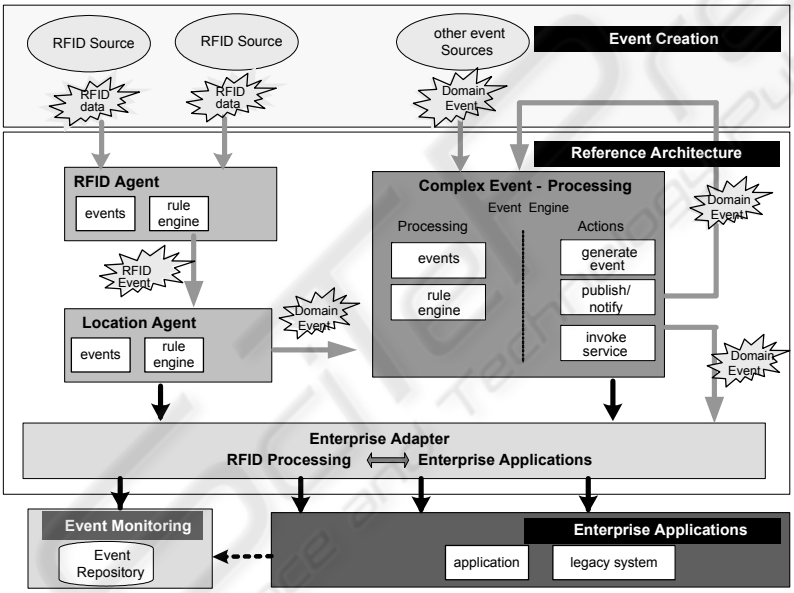

Fig. 1. Reference Architecture.

An **RFID Agent** serves as the interface to the different RFID sources and collects all incoming RFID data events. First, it provides the functionality of an adapter to integrate technically different, proprietary RFID systems. Furthermore, it transforms each RFID data event into a standard RFID event format and guarantees consistency, e.g. by filtering duplicates [9].Duplicates occur quite often, i.e. multiple RFID data events are emitted if a RFID tag passes a RFID reader.

A **Location Agent** transforms RFID events into application-specific domain events. To perform this task the Location Agent must query information from the enterprise information system via an adapter. The Location Agent has to provide two

different mappings: First, the technical position, i.e. the ID of the RFID reader must be related to a domain-specific location, e.g. a certain room. Secondly, the ID of an RFID tag must be assigned to a domain-specific item. For instance, the tag ID of a laboratory sample must be mapped to the sample ID in the laboratory information system. The data of new items is stored in the information system by manually entering it or by reading the tag data. There is a specialized workstation equipped with a RFID reader to determine the relation between RFID tag ID and item ID.

The key concept of the reference architecture is the **Complex Event Processing (CEP) component**, where the domain events are processed by application-specific rules. The CEP component contains a rule engine with the rule base and buffers the event data. It analyzes continuously the events and executes predefined rules if a specific event pattern is detected. Possible reactions are the generation of new events or the invocation of a service of the enterprise applications via an adapter.

The **Enterprise Adapter** component provides a facade giving access to the enterprise application systems for the Location Agent and the CEP component. The Location Agent requires the adapter to map technical RFID positions to domain-specific locations. The CEP component uses the Adapter to invoke business services for event handling, i.e. to make enterprise applications aware that certain events have occurred. Note, that the communication is unidirectional: the enterprise applications do not know anything about the Location Agent or the CEP component.

The **Event Monitoring** component is optional, and contains a repository to save the history of relevant events. It is the only CEP-specific part visible to the enterprise applications. It allows them to analyze event specific data and to visualize it in a Business Activity Monitoring application.

The reference architecture separates the RFID data processing tasks from application-specific ones. The RFID data processing is performed by the RFID Agent and the Location Agent, which transform RFID data into consistent domain events. The application-specific processing is performed by the CEP component where application-specific rules perform pattern matching and event processing tasks.

4 Application Scenario

In order to prove the practical usefulness of our reference architecture we selected the tracking and localization of samples in large research laboratories using RFID. In large research laboratories the incoming samples are usually divided in several subsamples which can be tested simultaneously in different departments of the laboratory. Therefore, the parts of one sample are spread over different laboratory rooms while being tested. Despite of organizational regulations, valuable time is wasted to search the samples for the upcoming tests. To solve this problem RFID can be applied. Each incoming sample gets an RFID tag. Laboratory locations are equipped with RFID readers, for example doors and testing equipment, in order to track individual samples.

The types of events in this specific domain, which are shown in the UML state diagram in figure 2.

– When a tagged sample passes an RFID reader the RFID data is transferred to the RFID Agent who transforms the proprietary format into an RFID-Event containing tag ID, reader ID and timestamp. If a sample tag stays longer in the area of an RFID reader, usually multiple RFID events from the same sample tag are emitted. In fact these events are logically identical duplicates, only their timestamps differ slightly. Therefore, the RFID Agent of figure 1 filters and aggregates events from the same RFID tag and the same RFID reader to one Location-Event, which contains the ID of the reader. If the reader is located in the room a Room-Event can be generated.

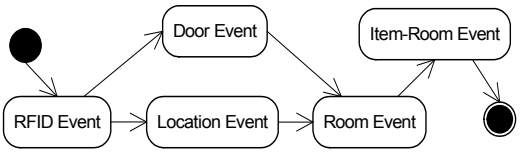

Fig. 2. Different types of Events.

– The Item-Room-Event is generated by content-enrichment of the Room-Event with the assignment of the tag ID to the domain item. The ID and type of the tagged sample are retrieved by accessing the laboratory management system. The Item-Room-Event contains all information to localize a sample and can now be considered as a domain event.

– The Door Events are originated by RFID readers located in doors. If such an event occurs it has to be decided whether a sample is entering or leaving a room for determining its location.

The CEP component of figure 1 contains a domain-specific rule base that is applied to all domain events. The domain events are transformed, classified, aggregated, evaluated, and appropriate actions are initiated according to the requirements of the application domain. For example, every Item-Room-Event is transferred to the laboratory management system in order to provide the information of the current location of the sample. Furthermore, all events of the same original sample are aggregated to a complex event representing the state of the testing procedure of the sample. This aggregated complex event can trigger upper-level business processes to take some actions if a certain state is reached.

5 Summary and Future Research

In this paper, we suggested a reference architecture for event-driven RFID applications. The key concept of the proposed approach is the use of complex event processing as the model for data processing. The flexibility and adaptability of the proposed architecture for RFID applications is achieved by distinguishing general RFID data processing tasks from application-specific ones. The presented event-driven architecture addresses two major challenges for the success of RFID technology: Firstly, the processing and managing raw RFID data; secondly, the integration of low-level RFID

data into the enterprise backend IT infrastructure, i.e. how to use RFID events to drive upper-level business processes. With the support of the logistics business processes in large research laboratories a new innovative application domain is investigated, in which RFID technology has not been applied so far.

A main research challenges, among others is the specification of event models [2] and of a standardized language for the definition of event patterns and event rules, a so-called Event Processing Language [11], [15]. Another important issue for the success of EDA is the need for real-world EDA applications in order to prove the practicability of the architectural paradigm [7]. Experiences with EDA for RFID applications are still rare. In [16] [14] and [6] first experiences with EDA for RFID are reported and event processing languages are proposed.

At the moment, no generally accepted standards exist for event definition and notation, event pattern specification, or rule languages and engines. Most rule engines have proprietary APIs, making them difficult to integrate with enterprise applications. This lack of standardization is the major obstacle in order to fully exploit the benefits of event-driven architectures. However, some notable advances in the standardization process can be recognized recently, e.g. the Java Rule Engine API [8] or the rule language RuleML (Rule Markup Language [13]).

References

1. Arasu, A., Babu, S., Widom, J., 2003. CQL A language for continous queries over streams and relations. 9th Intern. Conf. on Data Base Programming Languages (DBPL), pp. 1-19.
2. Adi, A., Botzer, D., Etizon, O., 2000. Semantic Event Model and its Implication on Situation Detection, 8th European Conference on Information Systems
3. Esper, 2008. http://esper.codehaus.org, retrieved 18. Feb.
4. Fowler, M., 2002. Enterprise Application Architecture. Addison-Wesley.
5. Garfinkel, S., Rosenberg, B., 2006. RFID: Applications, security and privacy, Addison-Wesley.
6. Gyllstrom, D., Wu, E., Chae, H.-J., Diao, Y., Stahlberg, P., Anderson, G., 2007. SASE: Complex Event Processing over Streams, 3rd. Biennal Conference on Innovative Systems Research (CIDR), pp. 407-411.
7. Hewlett Packard, 2002. Zero latency enterprise architecture. White paper.
8. Java Community Process, 2008: Java Specification Request JSR 94: Java Rules Engine API, http://www.jcp.org/en/jsr/detail?id=94, retrieved 18. Feb. 2008.
9. Jeffrey, S., Alonso, G., Franklin, M., Hong, W., Widom, J., 2006. A pipelined framework for online cleaning of sensor data streams, ICDE, pp. 140-142.
10. Luckham, D, 2002. Power of Events. Addison-Wesley.
11. Luckham, D., 2006. A View of Current State of Event Processing. First Workshop on Event Processing, New York,.
12. Paton, N., Díaz, O., 1999. Active Database Systems, ACM Computing Surveys, Vol. 31, 9, pp. 63-103.
13. The Rule Markup Initiative, 2008. www.ruleml.org, retrieved 18. Feb. 2008.
14. Rizvi, S., Jeffrey, S., Krishnamurthy, S., Franklin, M., Burkhart, N., Edakkunni, A., Liang, L., 2005. Events on the Edge. ACM SIGMOD International Conference on Management of Data, pp. 885-887.

15. Schiefer, J., Rozsnyai, S., Rauscher, C., and Saurer, G., 2007. Event-driven rules for sensing and responding to business situations. Inaugural International Conference on Distributed Event-Based Systems, pp. 198-205.
16. Wu, E., Diao, Y., Rizvi, S., 2006. High-performance complex event processing over streams. Proceedings of the 2006 ACM SIGMOD ICMD, pp. 407-418.
17. Zimmer, D., Unland, R., 1999. On the semantics of complex events in active database management systems. ICDE, pp. 392-399.