20

RFID Information Value Chain and ETRI RFID Ecosystem: Value-added Environment Linking Physical and Virtual Worlds

Tae-Su Cheong[1] and Yong-Jun Lee[2]
[1]*School of Industrial and Systems Engineering, Georgia Institute of Technology,*
[1,2]*Electronics and Telecommunications Research Institute,*
[1]*USA*
[1,2]*Republic of Korea*

1. Introduction

Since MIT Auto-ID lab envisioned the concept of "a networked physical world" with tagged objects (Sarma et al., 2001), RFID technology has gained a lot of significant attentions from academia and industries, and its research on hardware issues as well as software platform has been actively studied so far. In fact, RFID technology has shown itself to be a promising technology to keep track of the items in real time and further enhance operational efficiency. As the RFID technology is being spread and applied to real world systems, the research focus is changed accordingly from hardware to business application areas and relevant software systems. Having considered the significance of software and business integration for (ideally) automated system, it is indispensable to have intelligent software platform designed for RFID technology to deal with large amounts of data and complex business contents in order to create *value* in the sense of business performance efficiency.

This chapter is divided into two folds: First, we introduce the concept of *RFID Information Value Chain* (RFID IVC), which reflects the evolution of RFID data semantics and investigate how RFID software systems should be organized in order to support such value chain. In this part, we focus on the information evolution activities converting raw RFID data to useful information which may even be used to lead to automated business process execution and further to knowledge to support decision making. We examine the elements and relevant activities on the information value chain.

Second, *ETRI RFID Ecosystem* which materializes RFID Information Value Chain is presented from architectural point of view. ETRI RFID Ecosystem consists of RFID middleware performing raw RFID data processing in the primitive level, rule engine generating business semantic events and orchestration engine coordinating automated business process. In this part, we explain how the software systems are implemented in association with RFID Information Value Chain and elaborate each individuals of ecosystem in the implementation perspective. We remark that the usual concerns of scalability, reliability, distributed management and interoperability are as relevant to RFID middleware as they are in other middleware domain. In addition, the extensibility for readers as well as user-defined functions, and global RFID standard compatibility such as EPCglobal and

Source: Development and Implementation of RFID Technology, Book edited by: Cristina TURCU,
ISBN 978-3-902613-54-7, pp. 554, February 2009, I-Tech, Vienna, Austria

ISO/IEC standards must be taken in consideration for better market positioning. Therefore, we analyze the requirements and highlight several architectural considerations in order to build a distributed, reliable and standard-compatible RFID software system. We present our efforts aimed at the RFID software platform which is distributed, reliable and global RFID standard-compatible.

2. EPC network and RFID middleware: representative reference model for RFID software platform

The EPC Network is a networked infrastructure for gathering, sharing and accessing EPC-related information about physical movement of each EPC-tagged items as it passes through supply chain. It was proposed and developed by Auto-ID Center (now, Auto-ID Labs[1]) at MIT and is currently maintained by EPCglobal[2]. It comprises of the following main components: (i) Electronic Product Code (EPC), an item identifier, (ii) RFID reader and tags, (iii) RFID middleware (*Filtering & Collection*; Its interface is named as Application Level Events (ALE)) (EPCglobal, 2005b; EPCglobal, 2008a) located in-between RFID readers and business applications and performing filtering and aggregation of EPCs, (iv) EPC Information Services (EPCIS; whose interfaces are *EPCIS Capture Interface* and *EPCIS Query Interface*) (EPCglobal, 2007b) which is a set of interfaces and repository for enabling EPC-related data sharing within or across the enterprises, and (v) Object Name Service (ONS) (EPCglobal, 2008b) and Discovery Services, authoritative directories of information sources or EPCISs associated with EPCs.

When it comes to the RFID middleware, the researches in EPCglobal have been continued with somewhat gradual progresses. In the initial specifications from Auto-ID Center (Oat Systems et al., 2002), the software called 'Savant' was to perform data routing operations for data capturing, monitoring, and transmission, and the specification focused on the functional analysis for RFID middleware software system.

In the later version of the specification about Savant from EPCglobal (Clark et al., 2003), a matter of concern moved from the specific processing features into a flexible container with the generalized external interfaces for outer service applications. Savant became a container of processing modules for specific features and may be customized to meet the needs for applications. Finally, in the latest version about the middleware, now called 'Application Level Events (ALE)' (EPCglobal, 2005b; EPCglobal, 2008a), it specified the external interfaces only for defining the control and delivery of the filtered and collected tag read data, and then outer applications have access to ALE in order to obtain the tag read data of interest.

We briefly presented EPC network architecture and one of its main software component, RFID middleware. We remark that most of currently available middleware software systems in the market follow EPC network architecture spanning not only middleware itself but also EPCIS (Sun Microsystems Inc., 2006; IBM, 2006; Oracle, 2008; Floerkemeier, 2007). However, EPC network architecture is mainly focused on supply chain management–oriented mechanism and associated information traceability. In this chapter, we view the RFID-based software architecture from a different viewpoint as the RFID data semantics evolution and propose RFID software framework supporting such the data transformation process as well as RFID-based automated business process execution in the end.

[1] http://www.autoidlabs.org/

[2] http://www.epcglobalinc.org/home

3. RFID information value chain

In this section, we discuss how to manage the information flow from RFID tag sensing to the delivery to RFID applications. In general, RFID system has its advantage on the automatic identification of individual objects without human intervention. Moreover, the structure of EPC Network is designed to support the information exchange among trading partners over the supply chain. However, in order to create value by utilizing RFID infrastructure, the following process must go through: information that is filtered and aggregated from RFID raw data is well-used for the enterprise applications as valuable resources, and later significant knowledge is derived from the accumulated information. In other words, low-value raw RFID data is transformed into useful information and further proper guidance for management and this implies that such transformation activities are in a value-added process. Therefore, we define this value-added data transformation process as '*RFID Information Value Chain* (RFID IVC)' and it pursues the maximization of the benefits from the RFID deployment. Figure 1 shows RFID IVC along with the RFID solution components and the correspondent activities for each step are described and, here after, we briefly present the key activities per each stage corresponding with RFID IVC.

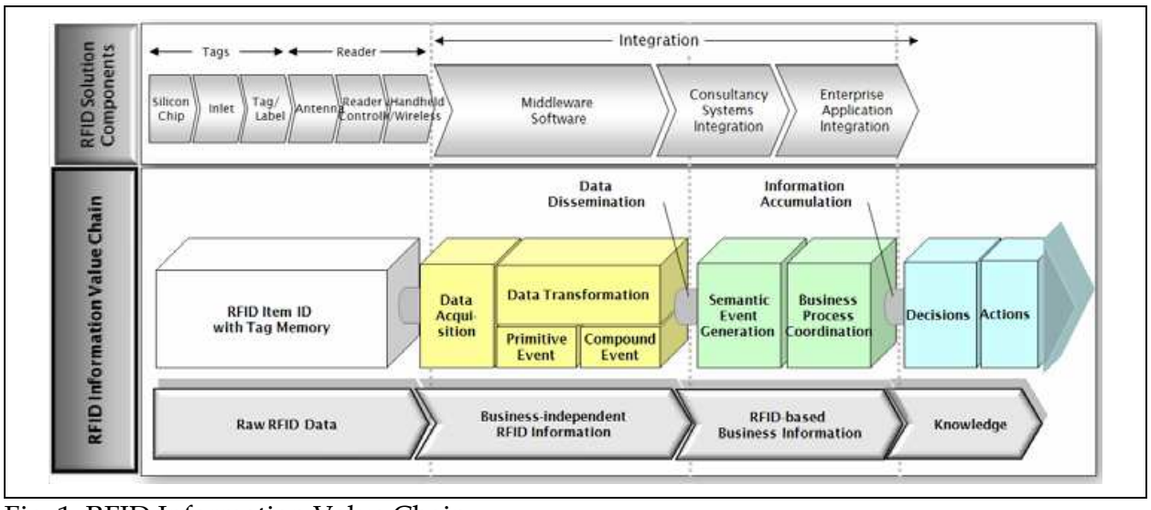

Fig. 1. RFID Information Value Chain.

3.1 Data acquisition

This stage is the first step to gather the RFID data from various types of RFID readers and other types of object identification devices like barcode system. At the present, there exist different kinds of RFID tag-reader protocols including EPCglobal Class 0, Class 1, and Class 1 Generation 2 tag-reader protocols (EPCglobal, 2005a), and ISO/IEC 18000-x series (ISO/IEC JT1/SC31/WG4, 2008). Moreover, the market-available reader systems use the different I/O interfaces. In addition, the heterogeneity among RFID reader systems must be dealt with in order to support the seamless data in-flow.

3.2 Data transformation

In general, RFID system generates a vast amount of data in the low level and such data right from RFID reader has little direct value to the enterprise applications. Hence, it is desirable to provide mechanism that the captured RFID raw data is transformed into the

correponding information in the business context so that such mechanism can endow the RFID captures with a significant value in the business sense. This stage, data transformation, primarily involves either filtering or aggregation of raw RFID data. Here, the meaning of 'filtering' is that not all data may be necessary to the applications, so signification data items are identified and the remainder discarded. With the consideration of the characteristics of RFID, the tag data captured by RFID readers have the technical limits as follows :

First, RFID readers cannot guarantee 100% accuracy of tag reading at the present because of interference, limited bandwidth, collision in the dense readers environment and high sensitivity by the surrounding environment (Floerkemeier, 2007). Therefore, it is necessary to resolve the physical limits and the smoothing process can be considered as one method (EPCglobal, 2006).

Second, RFID reader is physically non-contact to communicate with tags and the number of RFID tag data flowed from a RFID reader ranges from 10s of tag data per second up to more than 100s a second. This leads to bring the big burden to the host system which is responsible to process all the data, so the appropriate scheme for data volume reduction is required.

Third, as pointed out at the second, an RFID reader can read multiple RFID tags simultaneously and, sometimes, tags are unintentionally sensed from an area beyond one intended to be monitored by the reader due to many reasons including the reflection of radio wave. Therefore, among the captured tag data, not all the data are required to the applications that consume RFID data, so the tag data in which the applications are not interested should be filtered out.

This stage is responsible to handle the problems above and two levels of RFID event processing are considered: primitive and compound event processing.

Primitive Event Processing (Purification)

The data emerging directly from RFID readers may be regarded as a stream of records of the form (*r, n, g, u, t*), denoting the antenna *n* of the reader *r* reads tag *g* at time *t* (with tag memory *u*). In this stage, it is responsible for mapping the low-level data stream having the limited information to more manageable form suitable for application-level interactions.

An objective on this sub-stage is the volume reduction of data stream by discarding the redundant tag data. In general, when a tag appears present to a particular RFID reader for many read cycles, this generates a lot of data. Moreover, the tag might not appear every read cycle although the tag stays in the read range. As the methods to help overcome these problems, the event smoothing process is introduced (EPCglobal, 2006).

The meaning of 'smoothing' here is to pass the tag read only when something of interest happens such as when a tag is first captured by the reader or when the tag is no longer present. Figure 2 presents the state transition diagram used in the smoothing process.

As seen in Figure 2, there are three states per each tag – Unknown, Peeked, Captured – and four different events – eventPeeked, eventDisappered, eventCaptured and eventExpired, and the tag read can be passed to next stage only if the state transition between two adjacent states occurs in suitable situation. Initial state of a tag is 'Unknown' and, when a tag appears for the first time in the read range, the event 'eventPeeked' is generated and the current tag state is moved onto the state 'Peeked'. The event 'eventCaptured' is generated when the tag is seen for a certain period and the event 'eventExpired' occurs if current state of the tag is 'Captured' and has not seen for a while. The event 'eventDisappered' is generated when the tag hasn't seen for a time without subsequently generating 'eventCaptured' event. If a tag

generates 'eventPeeked' and then 'eventDisappered', it means that the tag enters the read range briefly and it can be regarded as a candidate for an inadvertent tag read.
What we can gain from this primitive event processing sub-stage is as follows: (a) reduction of data volume by reporting only if the state transition occurs, (b) establishment of the decision basis which is used to distinguish tags that remain for time interval enough to be regarded as tags that the reader should monitor, as compared to tags that enter the read range only too shortly or are reported unwillingly.

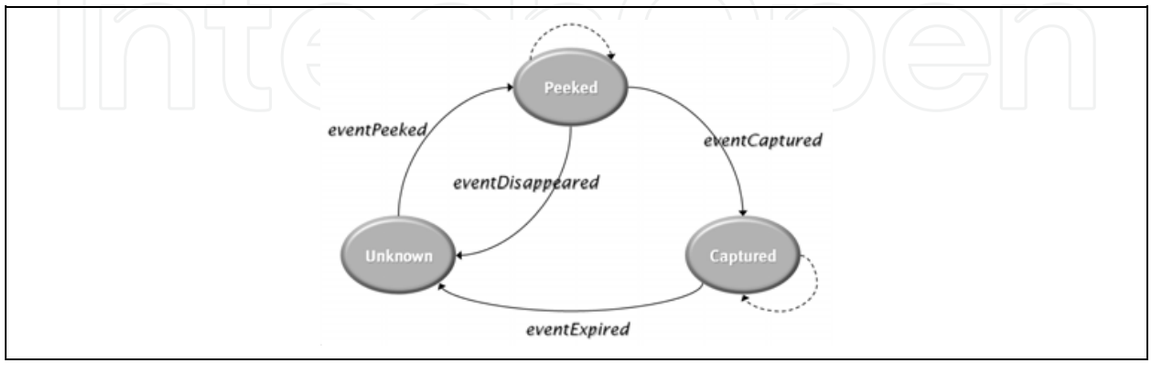

Fig. 2. Primitive Event Generation.

Compound Event Processing (Refinement)

The primitive event processing sub-stage mainly focuses on the duplicated data volume reduction. On the other hand, this stage - compound event processing - concentrates on the selection of tag data that are meaningful to the business domain (but, not dependent on business domain) while having the limited information delivered from RFID reader. There are several literatures dealing with complex event processing by utilizing existing complex event process techniquies (Ku et al., 2007; Dutta et al., 2007) and most of them describe the complex event with ECA rule-type algebraic notations and the corresponding algorithm for interpretation and exeuction. We also remark that Application Level Events (EPCglobal 2005, 2008) specification can falls into this category.
Data from readers follow the feature of record (*r, n, g, u, t)* and, as passing through the primitive event processing sub-stage, the event type *e* - which itself can be regarded as significant event information - is added as an additional field in the record as like the record (*r, n, g, u, t, e*) in exchange for the elimination of redundant data. At this stage, the information meaningful to business application has to be derived from the restricted information.
Basically, the event filtering can be applied to each field of the record or the combination of fields. For example, an application may want to select the tag reads which are captured by specific readers and another application may be interested in the tag reads that their IDs start with the specific pattern. Here, we have five categories to apply the filtering operation – reader ID, antenna number, tag ID, timestamp, event type and even tag memory (if possible) – and those are the basic filtering schemes. The basic filtering schemes based on the following fields – 'reader ID', 'antenna number', 'tag ID', 'event type' and 'tag memory' – are applied like this: the filtering operations applied to the associated field on the event record (*r, n, g, u, t, e*) forward the records that only fit within specific range or pattern; on the other hand, the basic filtering scheme based on the field 'timestamp' can be utilized as duration-based filtering since RFID data are highly temporal. The timestamp-based filtering

scheme usually combines with other basic filtering schemes to support more advanced filtering. In addition to the basic filtering schemes, many other filtering schemes are considered as follows:

Association In the case that several readers are associated with the same tag reads, it needs to allow a group of readers to be filtered for duplicate RFID tags so that the application which consumes the tag read seems as if the tag read is originated from the single reader. To provide the association, it needs to combine the event record (*r, n, g, u, t, e*) with the relationship (*r, l*) providing the logical location *l* of reader *r*. EPCglobal Application Level Event (ALE) specification handles association with `Logical Reader'.

Aggregation & Containments some cases, a number of tags is sensed by readers at the same time, then the group of sensed tags itself seems significant event to the applications. For example, suppose that readers located in the entrance of warehouse capture a number of cases and one pallet containing them within a short time interval. In order to derive the meaningful information to the applications, the sensed tag list including both cases and a pallet should be processed as a batch. That is, this batch filtering scheme clusters tag reads into groups of distinct tags based on when they have been observed during the given interval.

Read-Range In/Out For the situation of warehousing of goods or taking them out of the warehouse, reporting in-field and out-of-field events when tags move in and out of the read range is more significant to the corresponding business applications. This situation can be handled by taking advantage of the 'event type'-based basic filtering scheme; that is, use the event type 'eventCaptured' and 'event Expired' respectively among the fields of the record (*r, n, g, u, t, e*).

In general, the filtering schemes which applications want to apply vary from application to application and, we believe that the schemes discussed above are considered as the commonly used filtering schemes (for more detailed complex event process, see ALE specification or, Ku et al.(2007) and Dutta et al.(2007)).

3.3 Data dissemination

A critical task of any information system is to deliver the right information to the right people or applications at the right time. The dissemination activity involves moving the filtered and aggregated information from RFID readers to enterprise applications or other business integration applications. The purpose of data dissemination activities is to determine who needs what information and to deliver it to them on time.

We note that RFID-enabled business applications tend to have event-driven nature and, in order to meet this requirement, it must be required to provide the following types of data dissemination models at the same time; *the push model* (or *query/response model*) that RFID data captured by RFID readers keep flowing upward to the backend applications over pre-defined event pipeline, and *the pull model* (or *publish/subscribe model*) that applications which are interested in handling RFID data subscribe to the middleware system with additional information such as notification cycle, reader set which they are interested in listening to and so on, and the middleware plays a role of dispatching messages to the subscribers asynchronously.

3.4 Semantic event generation

At the second stage of 'data transformation', filtering and aggregation methods of raw RFID data are demonstrated. However, it may not be sufficient for the filtered data to become the meaningful information to enterprise applications. At this stage, the filtered data are

combined with different sources residing in the legacy system or external sources such as purchased data services so that semantic events which are significant in the business domain are generated. Here, the term "semantic event" means that, as it says, RFID data capturing events merely indicate raw observation and taking business actions based only on the event reports are somewhat limited; therefore, the raw observation events need to be combined with additional business context information in order to construct business events (here, we call 'semantic event') upon which legacy applications can play with. Furthermore, the reason why this stage is required is to enable sophisticated RFID-based data processing. As domain-specific information is integrated with RFID tag data, content-based filtering and routing become possible by mapping and combining tag data with corresponding object information and applying basic filtering scheme on the combined data.
In other way, Event-Condition-Action (or ECA) rules (Palmer, 2006) can be applied to generate the semantic events. When a primitive tag data are delivered from the previous stage as an event, the rule set associated with the event is evaluated and then appropriated actions are taken. For example, ECA rule mechanism can be useful when predefined action is taken by the comparison of the captured tag list with the scheduled information with little human intervention. If the sensed tag list mismatches the schedule information, another action to detect the problem can be taken. We may regard such the inference process as the semantic event generation process making use of RFID tag data and additional information stored in backend systems and supporting the conversion from raw RFID data to actionable information.

3.5 Business process coordination

Main objective in the stage is to enable business processes and solutions to leverage the real-time data captured by RFID infrastructure. The key benefit of RFID technology is automatic identification of individual objects coupled with automatic data capture. The employment of low-levels of process automation toward the process automation and efficiency improvement ultimately leads to the high return in terms of efficiency and cost reduction.

3.6 Decision / actions

One of the desired advantages adopting RFID technology is real-time information gathering, exchange and real-time item visibility. It means that real-time decision making could be realizable. For example, real-time inventory monitoring suggests optimum reorder points based on usage and improves inventory accuracy. Another purpose of this stage is to provide guidance for action to decision maker based on the accumulated information and ultimately produce the knowledge. As a lot of RFID data and related production information are accumulated, it is possible to elicit the valuable knowledge from them – for example, RFID data warehousing (Gonzalez et al., 2006). There are many methods to produce guidance for decision maker and further knowledge as following:
Views of current or historical information. This is the simple approach and the modeling usually consists of aggregation, summarization and filtering.
Forecast. This requires using a methodology like statistical regression based on the current and historical information.
Recommendation of the best and alternative decisions. To find the best recommendation, an optimization model searches among various alternatives and decides the best. Finding a reorder point in inventory problem through various optimization techniques is an example.
Inference through data mining. This is the process to elicit knowledge by searching for the pattern hidden within accumulated information.

4. ETRI RFID ecosystem

ETRI RFID Ecosystem is an RFID software platform that supports not only the presented capabilities that RFID middleware platform must provide but also the activities occurred on RFID IVC, and ultimately provides the seamless environment spanning from the edge of the enterprise network to the enterprise systems.

Figure 3 presents ETRI RFID Ecosystem in accordance with RFID IVC. ETRI RFID Ecosystem is a multi-layered middleware platform in Java environment. The first layer – RFID Event Management System (REMS[3]) – deals with primitive and compound event processing in order to obtain purified and refined RFID event while having less business context. The second layer – Real-time Business Process Triggering System (RBPTS) – is responsible for generating the semantic business event by utilizing refined RFID event. On the top layer, Orchestration Engine (OE) supports the autonomous business process execution. The top layer deals with the generation of invaluable knowledge. Additionally, Tagged Object Information Repository manages the tagged object information and makes them available to whatever the information are required for the purpose of interoperability and exchange within or among enterprises. It is designed to offer the seamless environment extending from RFID hardware infrastructure to backend software systems, and support the RFID IVC. In this section, we introduce the architectural considerations for RFID software platform implementation (mainly, RFID middleware implementation), and then, the functional features and the architecture of each individual that constitutes ETRI RFID Ecosystem is discussed in the following.

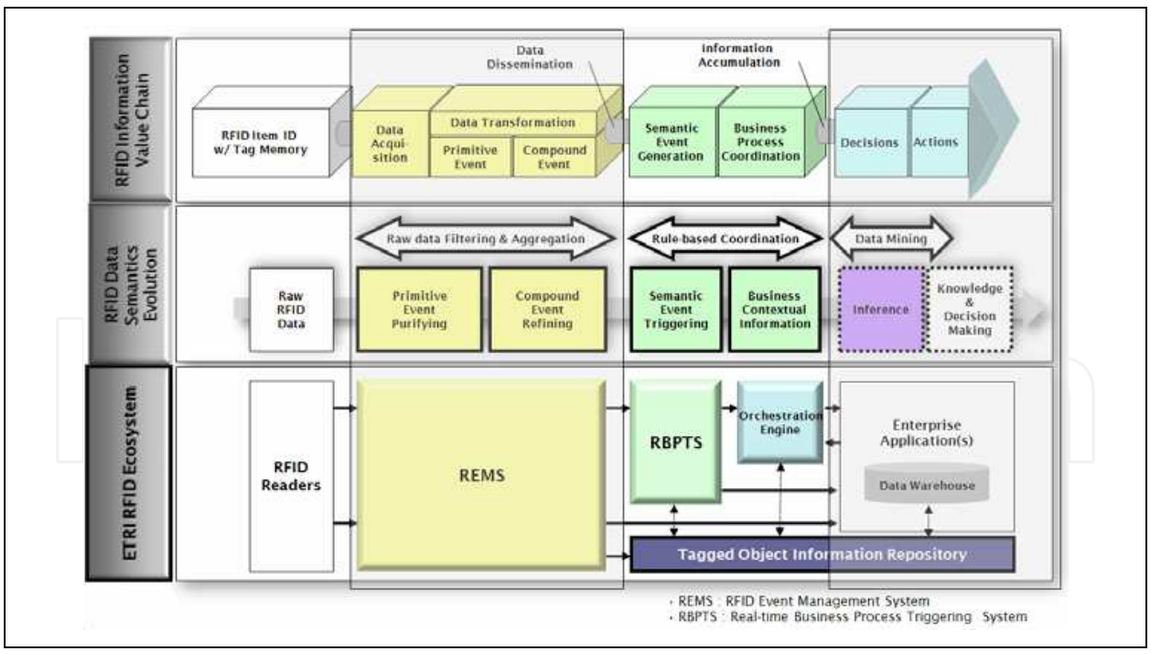

Fig. 3. ETRI RFID Ecosystem and its Correspondence with RFID IVC.

[3] ETRI REMS/EPCIS (including REMS) earned EPCglobal software certification (ALE/EPCIS). http://www.epcglobalinc.org/certification/sw_cert/

4.1 Architectural considerations for RFID middleware platform

We discuss the several architectural issues and our concerns toward REMS, RFID middleware since how well RFID middleware can perfom must be a decisive factor for overall system performance of ETRI RFID Ecosystem. We here deliver our approach to meet those requirements. There are several literatures which deal with concerns on RFID software requirements (Luo et al, 2006; Floerkemeier et al., 2007). Especially, Luo et al. (2006) proposed the following requirements for RFID middleware benchmarking: Streaming, Reactivity (event triggering features) and Integration. In the next, we present how ETRI RFID Ecosystem absorbs those requirements.

Component/Service-oriented Architecture

In general, many applications adopt the component or service-oriented frameworks – for example, Spring Framework[4] - in order to enhance the system flexibility and reusability. Among various component-based software architectures, we have chosen Avalon framework (Loritsch, 2001; SourceForge Inc., 2008a) to implement the RFID middleware servers. Using Avalon, it is straightforward to have the components of each server interact, to instantiate different instances of the components, and to reuse code while having light-weight and minimal features comparing with other application containers.

An Avalon applications, then, is composed of the Avalon infrastructure, the specified components, and a 'container' that reads the configuration files and starts the process running. The container reads the configuration files, loads the specified implementation classes, and invokes the Avalon interfaces in order. In this implementation, we use Phoenix (SourceForge Inc., 2008c) as a container.

Distributed System Architecture

The main role of RFID middleware is to filter and aggregate a lot amount of RFID tag events coming from RFID readers. If the situation of the item-level RFID tags attachment on individual items become realized in the near future, the single server which even equips with high computational capabilities may not control a great amount of RFID tag reads flowing into the system. Therefore, it is unlikely to handle a lot of data by a single server.

In this context, the term 'distributed' has somewhat different meaning from general sense. For business applications in general, enterprise-scale business applications are distributed and operated over the physically separated hardware in order to achieve the load balance and increase scalability. In this sense, it is applicable to the RFID middleware software as well. However, it has more than that: most of RFID readers can communicate with only one system at a time. Therefore, if a reader is deployed into a RFID software, then no other software except it can capture the tag reads by the reader. It implies that an application which wants to utilize the tag reads by specific RFID reader must cooperate with the software which have the connection with the reader.

In this implementation, we adopt the registry-based federated architecture. We name the software system that offers the resource locator service as 'Service Broker'. As shown in Figure 4, Service Broker acts as a 'federator' and other subsystem including our edgewares like 'Reader Management Subsystem' and 'Event Management Subsystem' are 'federatees'. When a subsystem connects into the network, it starts to send the predefined heartbeat messages including the server name and IP over the network. When the Service Broker

[4] http://www.springframework.org/

receives the message, it sends the acknowledge message back to the caller. If it is allowed to join the federation, it sends the registration information including the server name, the access information, and server type and so on, and asks for the download of common resources like RFID reader adapters if necessary.

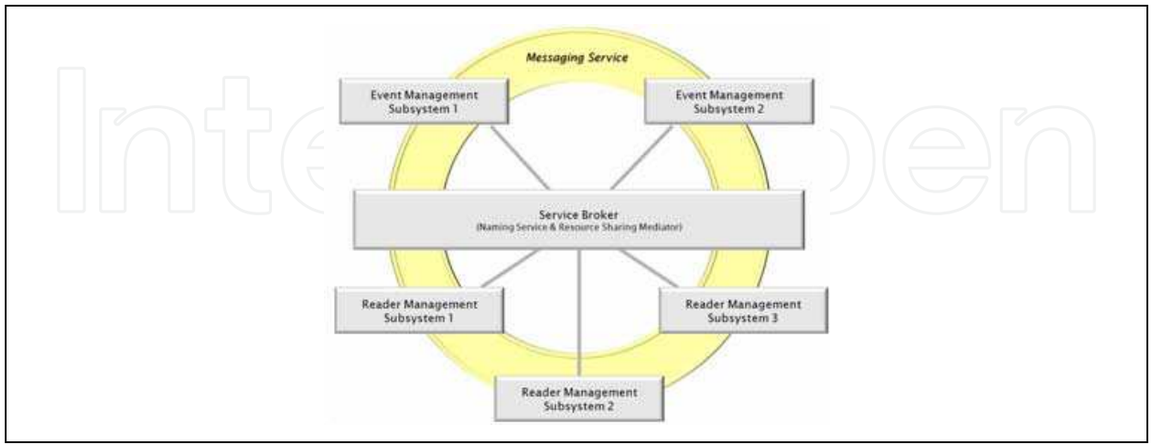

Fig. 4. Building Blocks for REMS, as a distributed system.

Reliability (Fault-Tolerant System Operation)

To ensure the reliable RFID data delivery to the desirable destinations, it is critical to eliminate any single points of failure over all the layers. For example, Sun Java System RFID Software (Gupta et al., 2004; Sun Microsystems Inc., 2006) adopts Jini technology (Sun Microsystems Inc., 2008) for this purpose. When we consider our distributed RFID middleware federation, there are three types of system failures that this RFID middleware platform must take into account.

Failure of RFID readers The failure of any single RFID reader may link directly with the loss of significant data in the business context. RFID middleware system only recognizes and tracks items within range of readers and the failure of a reader may result in the data missing for the items. To cope with this situation, our subsystem periodically checks the aliveness of the connection with readers. If a reader stays unconnected in a pre-specified period of time, the subsystem sends the failure notification by e-mail to the administrator in order for him to examine the reader and periodically keeps trying to reconnect to the reader until establishing the connection between the reader and the subsystem again.

Failure of the Individual Middleware Subsystem Each distributed subsystem may face the system down because of several reasons – for example, the increase of system overhead caused by the tremendous amount of data influx and failure of managing the system overflow. To avoid the subsystem failure, we adopt a simple solution: that is for the service broker acting as a control center to perform active monitoring, which consist of having individual subsystems periodically send keep-alive messages to inform the service broker of their aliveness. Thus, service broker always has an image about the health of their federation over the network. If it has not received the keep-alive information from a subsystem for a timeout, fault-detector module performs reachability test to the subsystem for conformation. It is certain that it has a problem that it generates the amount of traffic; however, an appropriate timeout period can be mediated with the consideration of the trade-off between alleviation of network traffic and responsiveness of failure occurrence.

When the service broker detects a subsystem's failure, it sends the failure notification by e-mail and then it packages all the operational resources related to the failed one and feeds them to the temporal running server in order to take over all the responsibilities which the failed subsystem has taken care of until then. At the startup stage of the service broker, the temporal subsystem also starts up for this case. This approach can be possible because the service broker keeps track of all the changes happened in the federation – that is, we adopt the centralized meta-information sharing in order to cope with such failure.

Failure of Service Broker It is important to guarantee the stable running of the service broker because it governs all the distributed individual subsystems as a control center. In order for the safe operation, the service broker operates as a dual mode and the secondary service broker maintains the redundant information with the primary one by periodically replicating the information the primary manages so that it makes sure the continuous and reliable operation of the service broker.

Various Passive/Active RFID Readers & External Sensors Support and their Management

RFID middleware should provide the means to handle the heterogeneity of RFID readers in terms of the vendors and versions. Most RFID middleware software systems can connect to the RFID devices via the reader adapters which play a role of managing communication in a standardized way between the reader and the middleware. In the implementation, eclipse-based adapter development toolkit is developed. Reader adapter programmers can write java codes for the reader adapter which they want to provide through this middleware and perform the Ant-based build process to generate the Jar package. The adaptor is designed to be compatible with EPCglobal RFID Reader Managment Protocol (EPCglobal, 2007a).

In addition, it is necessary to look at what custom configuration settings you may need to tune on the reader that you choose. Currently, many middleware vendors support varying levels of configuration on the RFID readers; however, some are limited in the amount of control, which means that you are not able to control key settings such as antenna power or antenna cycling. This may lead the users to manually configure the readers outside the middleware if a tunable parameter is not supported. This manual tuning process may make it difficult to manage the readers.

In order to alleviate such difficulties, we devise XML-based configuration for setting tunable parameters for each reader as shown in Figure 5. The adapter programmers can decide the scope of tunable parameters which are willing to be opened by simply exposing parameters in the XML documents like Figure 5 (b).

Lastly, for the applications which do not necessarily require continous RFID reading (Floerkemeier et al., 2007), it is preferable to have a mechnism to initiate tag reading by external sensors. At this time, the reader adaptor can also cover the registration of external sensors in the limited manner enough that the sensor-triggered RFID reader activation is achievable.

Global RFID Standards Compatibility

There are several RFID-related global standards which RFID middleware must concern: (a) interface between RFID tags and readers, (b) interface between RFID readers and host applications and (c) information exchange interface between RFID middleware and RFID applications.

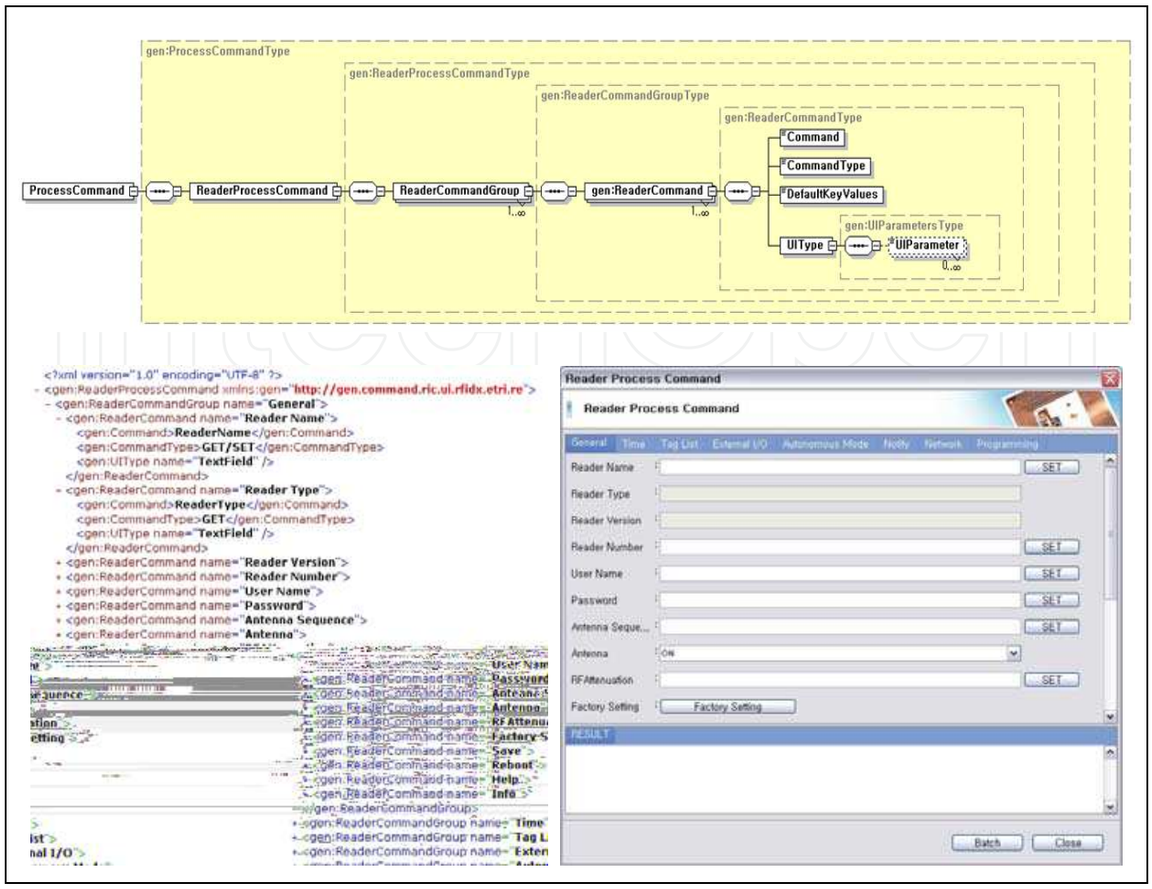

Fig. 5. Configuration for vendor-specific tunable parameters and commands: (a) XML schema for configuring tunable parameters for reader setting and commands (upper), (b) Example for Alien RFID reader[5] which is an instance of (a) (lower left), and (c) user interface rendering from (b) (lower right).

Tag/Reader Interface (Air Interface Protocol): EPCglobal UHF RFID Class 0, Class 1 Gen 1, Class 1 Gen 2 Protocol and ISO/IEC 18000 series

RFID Tag/Reader interface defines the physical interactions between RFID tags and readers. EPCglobal ratified the EPCglobal Class 1 Gen 2 protocol in 2004 and there are ISO 18000 series as a de jure air interface protocol standards in RFID system.

This interface has little concern with the RFID middleware implementation; however, it is required to consider the tag memory structures described in those standard specifications. The memory structure on the tags decides what types of information can flow in. Therefore, this tag memory structure affects the reader/host protocol design, which subsequently has an influence on the middleware implementation

Reader/Host Interface: EPCglobal Reader Protocol, Reader Management Protocol and ISO/IEC 15961, 15962

Reader/Host interface defines a set of commands for reader control, configuration and management, tag reading and writing. Each RFID reader vendor defines its own

[5] http://www.alientechnology.com

reader/host protocol and some vendors provide libraries to help programmers to develop RFID applications using their RFID readers in more convenient way. As part of resolving the diversities of reader/host protocol varying from vendors, the demand for developing general reader/host protocol leads to the development of EPCglobal Reader Protocol, and ISO/IEC 15961 and 15962. But, two protocols have different operations to have access to tag data due to the difference of tag memory structure and data organization in it. However, they define the general features which most RFID reader vendors also take into consideration, so most vendor-specific reader/host protocols can converge on either protocol.

When it comes to the middleware implementation, it is important to support as many market-available RFID readers as it can. Most middleware vendors provide toolkits to develop so-called 'reader adapter', which is a driver-like pieces that interfaces to actual RFID reader and provides a unified interface for RFID middleware to have access to readers. We also take the same approach to support various types of readers and devise the vendor-neutral reader/host API set in order to have access to those readers in a seamless way. The two global standard specifications play a critical role of deriving the common API set for the reader adapter while satisfying the diversities of vendor-specific protocols.

Middleware/Application Interface: EPCglobal Application Level Events

Application Level Events (ALE) is a software specification for the filtering and collection of RFID data being defined and ratified by EPCglobal. It defines a globally accepted method of filtering and collecting RFID information and it is the most representative standard interface between RFID middleware and external applications. As a result, this standard is expected to improve the interoperability between systems as it becomes widely accepted, so it is necessary to develop the middleware software that complies with this EPCglobal mandates, ALE. We fully implement ALE 1.0 (EPCglobal, 2005b) and currently extend it to meet ALE 1.1 (EPCglobal, 2008a) while reflecting ISO active tag features.

Application Integration

The ability to integrate RFID into the legacy systems or existing ones is absolutely critical to deliver the sensed tag events to the right applications in the right time. The simple approach is just to dispatch the captured events from readers to a series of applications at the low level; whereas, some form of enterprise application integration (EAI) is needed to get the full value from RFID events. Many major EAI solution providers like 'Tibco' try to integrate their solutions with RFID middleware and release to the market (Tibco Software Inc., 2006).

In particular, it is necessary to support various types of application integration methods including *push*, *pull* and *publish/subscribe* for application-level RFID information capture. For this, we take the following approach: our middleware is equipped with several standards-based adapters required to ensure connectivity to backend applications. In addition, it is necessary to allow application developers to register their own adapters to send the filtered RFID events to their legacy applications. In this implementation, there are six different protocols in order to let legacy applications receive the notification of RFID event messages – that is, HTTP, TCP/IP, JMS, File and Web Service (SOAP/HTTP). Users can subscribe to the middleware by entering URIs with specifying the protocol. Table 1 shows how to specify URI for each protocol.

Protocol	URI Template	Example
HTTP	http://<ip>:<port>/<web page>?pollingInterval=<millisecond>	http://129.254.238.16:8080/reports_log.jsp?pollingInterval=50000
TCP/IP	tcp://<ip>:<port>	tcp://129.254.238.16:7000
JMS	jms://<queue|topic>/<JMS Connection Factory>/<queue|topic name>?jndiInitialContextFactory=<Java class URI for JMS context factory} &jmsProviderURL=<URL of JMS Provider>	jms://topic/JmsTopicConnectionFactory/triggerTopic?jndiInitialContextFactory=org.exolab.jms.jndi.InitialContextFactory&jmsProviderURL=rmi://localhost:1099
File	file:///<directory>:/${SpecName}_${yyyy MMddhhmmss}.xml	file:///c:/sample_${SpecName}_${yyyyMMddhhmm}.xml
Web Service	axis://<end-point address>?optQName=<Operation QName> &optServiceName=<Service Name>	axis://localhost:8080/ECReportsService.asmx?optQName=escape('http://www.etri.re.kr')&optServiceName=OperationProcess

Table 1. URI definitions and their examples for RFID event subscription

Moreover, application programmers can develop their own event dispatch modules inheriting from designated Java interface we suggest and deploy them into the system in order to ensure the application-dependent protocol-based communication between the RFID middleware and their legacy applications.

4.2 ETRI RFID event management system (ETRI REMS)

RFID middleware is a software system that manages data communication between RFID readers and enterprise applications. In this section, we present the layered RFID middleware while considering architectural design aspects discussed in the previous section. We divide the RFID middleware into three layers – that is, 'device monitoring and management layer', 'data management layer' and 'business integration layer'. As given in Figure 6, the ETRI RFID middleware covers lower two layers and consists of two subsystems: Reader Device Abstraction & Management Subsystem for device monitoring, Event Management Subsystem for RFID data management and delivery. Also, there exists Service Broker for offering the name lookup service and resource sharing. The functional features and internal component architecture of each subsystem are described in the following.

Reader Device Abstraction and Management Subsystem (RMS)

The primary roles of RMS are to support the seamless integration between the middleware software and various kinds of RFID readers, and to monitor and manage the deployed readers. In order to handle the heterogeneity of readers and reader/host interface protocols, we abstract the reader/host interface APIs with help of the existing global reader/host interface standards mentioned in Section 4.1 and offer the eclipse[6]-based development toolkit for 'reader adapter'. Single reader adapter is developed per each vendor and version

[6] http://www.eclipse.org/

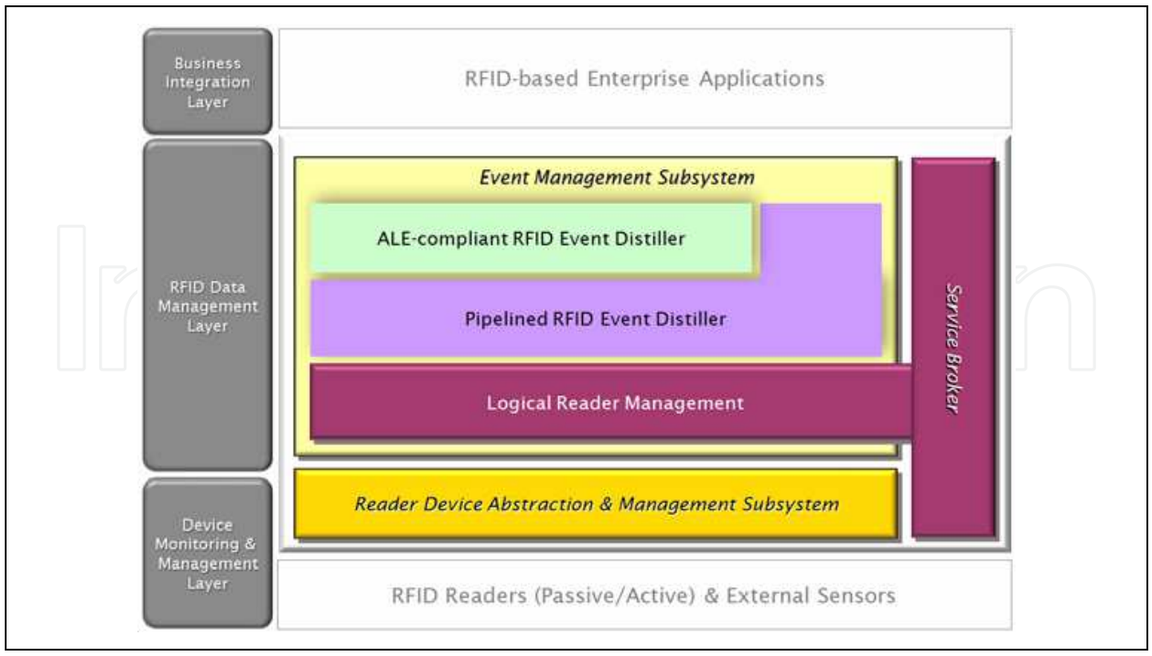

Fig. 6. Layered conceptual architecture of ETRI REMS.

if the vendor does not guarantee the backward compatibility of reader/host interface. Moreover, reader adapter developers take a responsibility of organizing the tunable reader-specific parameters by editing XML file shown in Figure 5 and implementing the proper execution codes for them. Those activities improve the extensibility for newly-introduced RFID readers at this middleware system.

In Figure 7(a), we show the component architecture of RMS using Avalon and the dependencies among the components. All the components are deployed and controlled by Phoenix. As shown in the Figure 7(a), there are three major components: Connection Manager, Monitor and Reader Agent Manager with Reader Agent.

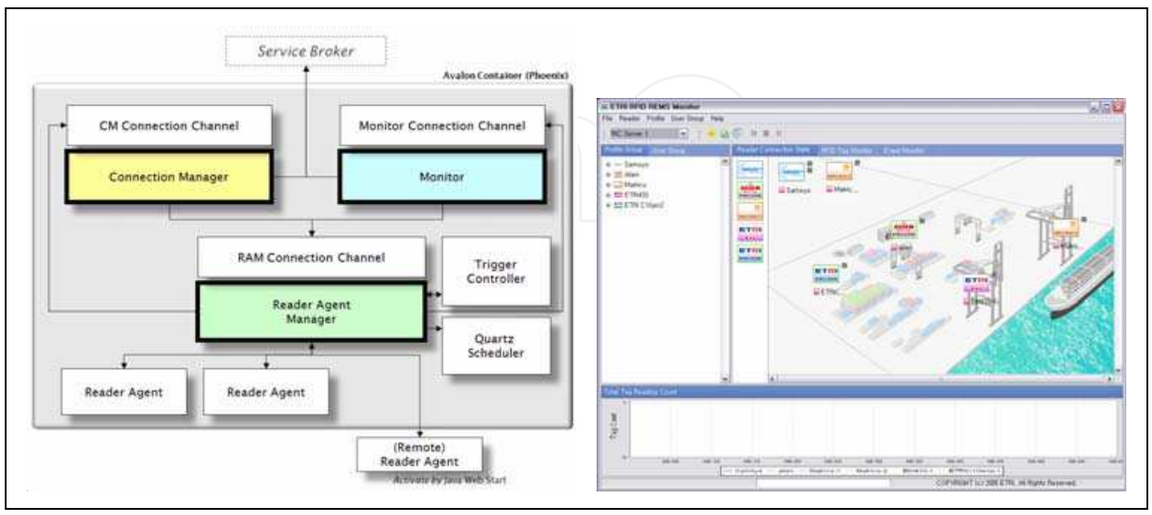

Fig. 7. Reader Device Abstraction & Management Subsystem: (a) component architecture (left) and (b) main user interface for reader configuration and monitoring (right).

The task of Connection Manager is to manage all the context information related to the deployed RFID readers and issue commands for reader management. All the commands exposed at the user interface are sent to this component. Then, this component validates and dispatches them to appropriate other components. The Monitor is responsible for monitoring the events occurred in the system. For example, it keeps track of the aliveness of individual active readers and notifies the erroneous events to the administrator or the reader management user interface in Figure 7(b).

The task of Reader Agent Manager is to manage the life cycle of Reader Agents representing the actively connected RFID reader or external sensors, and act like a container for Reader Agents. In general, a Reader Agent binds with a physical reader and handles all activities related with corresponding reader such as sending commands to a reader to control the reader and receiving tag data. Besides the reader-specific tunable parameter setting, each Reader Agent provides the following operations: (a) connect/disconnect reader, (b) suspend/resume reading tags (c) modify Reader Agent information, (d) delete Reader Agent and so on.

In addition, Reader Agent Manager has a dependency with Scheduler, Quartz, which is java-based open source scheduler. Basically, Reader Agent operates in a polling manner as a default operation mode in order to capture tag data in a reading zone; however, our implementation allows it to operate in the on-demand mode or user-specified schedule-based mode. For the latter case, the administrator specifies the Unix crontab-like expression with duration and Quartz scheduler awakes the Reader Agent based on the crontab expression and let it collect tag reads for the duration.

RFID Event Management Subsystem (EMS)

EMS is the core system in this middleware platform which filters data extracted from the RFID readers, aggregates the information and routes the data to the RFID-enabled applications (see Figure 8).

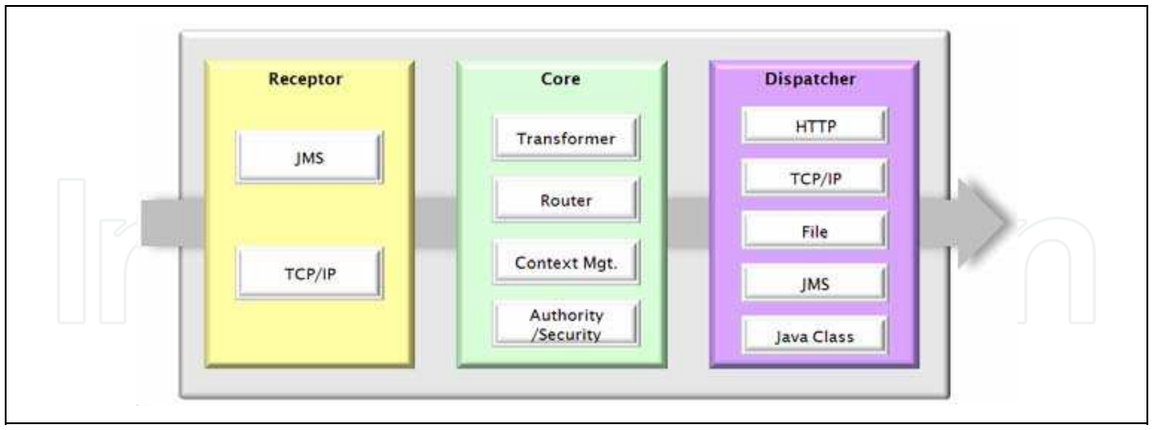

Fig. 8. Conceptual Representation of RFID Event Management Subsystem (EMS).

EMS enables ALE-based event queries and customizable event stream processing operations. Such the processes allow raw RFID data to be transformed into business information that can be leveraged by RFID-enabled external applications. In order to support reliable event processing and better performance, EMS adopts pipeline architecture as a basis for data processing. A pipeline consists of a set of primitive task processors called 'valves' and 'chain's connecting two valves. Pipelines categorize the influx data and process

those categories with a set of primitive tasks. By following the 'divide-and-conquer' like approach, it is expected to increase overall throughput and the average speed for high-volume data processing. Moreover, the XML-based event description named 'ECSpec' in ALE specification can be expressed in a pipeline way, so we stack ALE-specific processing modules over the pipeline-based processing modules in order to support ALE API as shown in Figure 9(a).

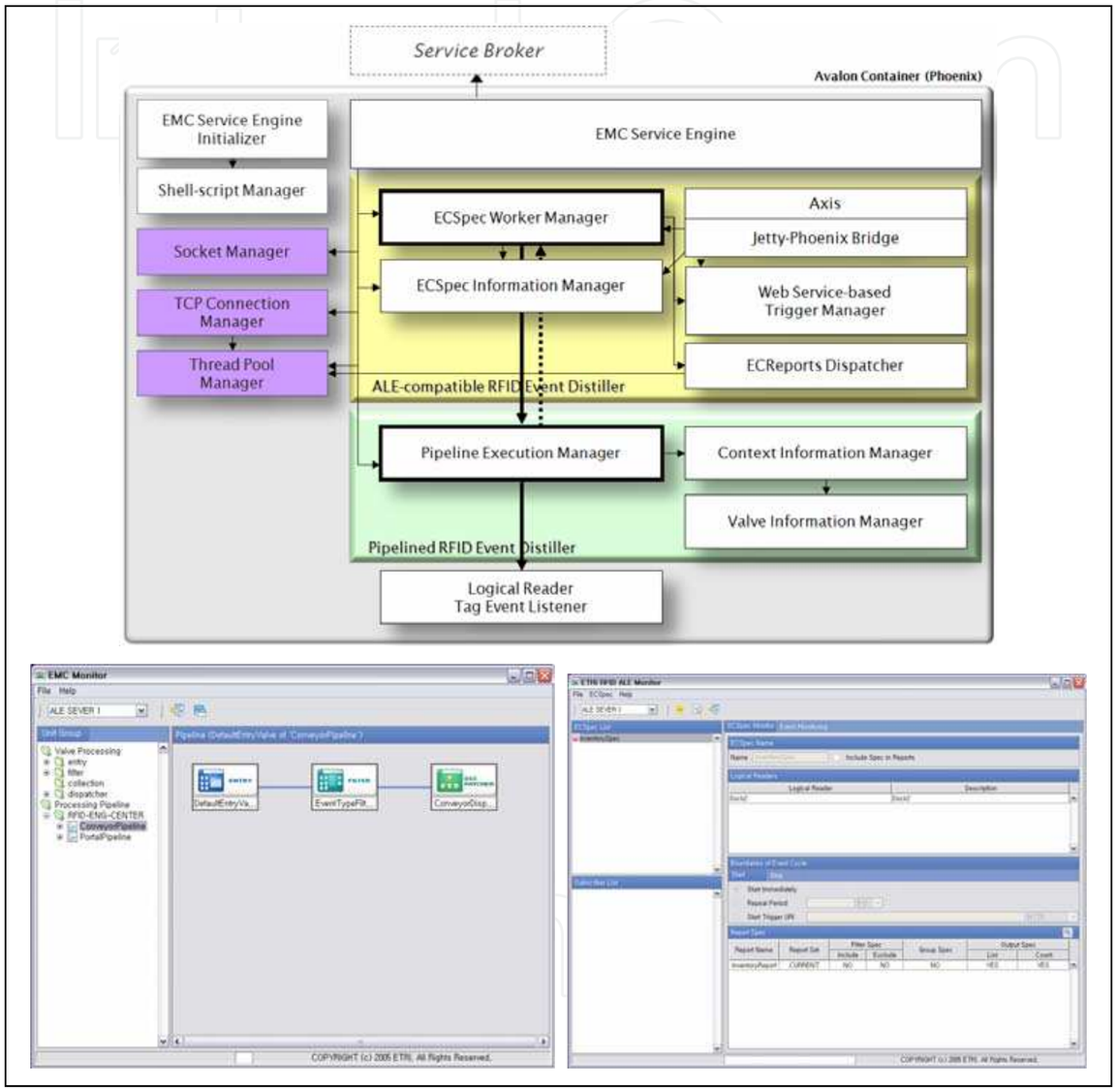

Fig. 9. Event Management Subsystem: (a) component architecture using Avalon (upper), (b) pipeline designer (lower left) and (c) ALE ECSpec designer and monitor (lower right).

Figure 9(a) shows the component architecture of EMS using Avalon and the dependencies among the components. As stated above, the building blocks for pipeline support reside in the bottom layer, followed by ALE-specific components. The major components are Valve Information Manager, Pipeline Execution Manager and ECSpec Worker Manager. Valve Information Manager is responsible for managing built-in or custom event processors.

Application programmers can create custom event processing valves as well as event dispatchers which pre-process filtered RFID data prior to propagating the information to external applications and use them while building up the event stream processing pipeline.
The task of Pipeline Execution Manager is to control the execution of pipelines and exception handling. We provide the user interface to define a pipeline instance as shown in Figure 9(b).
ECSpec Worker Manager takes a role of managing individual ECSpec Worker per each ECSpec given by outers. When the request for ECSpec is received, the ECSpec Worker Manager parses the request described in XML, transforms it into a pipeline and then asks Pipeline Execution Manager to execute the pipeline. The pipeline instance is linked with an ECSpec Worker internally so that the result of event processing by the pipeline is at first delivered to the ECSpec Worker. The role of ECSpec Worker is to manage the information of subscribers to the related ECSpec and handles the pipeline executions. Figure 9(c) shows the administration user interface for defining and monitoring ECSpec.
Lastly, we deploy the lightweight web server, Jetty (Mort Bay Consulting, 2008), with Axis in order to support the Web Service ALE API.

Service Broker

The Service Broker plays a role of a control center over the distributed network discussed in Section 4.1. It keeps track of the all the distributed subsystems and provides the name look-up service for them – especially, our user interface uses this look-up service to have access to individual subsystem.
Another major role of this service broker is to configure the logical readers. Generally, a single reader is not often sufficient to reliably cover the entire physical area relevant to a business process. For example, a loading dock may have to be equipped with several readers that should be exposed to client applications as a single logical reader. This enables EMSs to avoid the modifications from the change of readers in RMSs. Service broker offers methods to configure the logical readers and the relations between a logical reader and physical readers deployed to RMSs.

Fig. 10. Component Architecture of Service Broker.

Figure 10 shows the component architecture of Service Broker. We develop the Java servlet pseudo-container for Avalon components in order to reuse components we already

developed and keep consistency with other systems. As major components, Naming Service Manager maintains distributed system configuration information such as access information of all RMSs and EMSs. Heartbeat Message Monitor and Fault-Detector keep track of the aliveness of all subsystems and take reactions whenever any failure of them is captured. The task of Logical Reader Information Manager is to maintain the configuration of logical readers and the relations between logical readers and physical readers.

4.3 Real-time Business Process Triggering System (RBPTS)

Real-time Business Process Triggering System (RBPTS) is built on a rule-based inference engine which provides mechanism to extract semantic and business-context information from tag read events through ECA-type rules. Such semantic information is derived from domain-specific knowledge provided by domain experts or business collaboration partners and embedded in rule definitions. The semantic events – as discussed in Section 3.4 – are produced by associating the RFID primitive events with the domain-specific information residing in legacy system. This system receives a continuous stream of filtered and unfiltered RFID data from RFID middleware or RFID readers and produces the RFID-triggered business event by using set of rules. The produced semantic event is utilized for the query to execute collection of rules to perform various predefined actions ranging from one-time actions such as DB operation, the notification, alerts, actuator operations, or actions that involve the long term business process actions which require interoperation with workflow systems such as ebXML[7] engine and BPEL[8]-based workflow engine. The development of RBPTS is driven by the requirement of flexible way of incorporating RFID data with business applications; that is, to convert the data from lower RFID middleware layers to actionable semantic information for the upper layers.

In order to achieve the goal and be suitable for RFID environment, the rule engine of RBPTS adopts the backward chaining inference mechanism. As the physical RFID readers involve the specific business goals – for example, gate open/close, inventory check and so on – and the business actions triggered by the collected data fall into small number of categories, it is expected that possible conclusions can be chosen at the time that a set of tag data is collected by specified readers. The domain experts define a set of rules which are described as the 'If-condition(s)-then-action' pattern. The event message delivered by REMS includes the 'action' indicator called 'query' to be proved, so the inference process starts with a conclusion with the help of 'query'. The rule engine searches for the rule set which has the action clause that matches the action which the event message includes and then evaluate the associated condition clauses. The condition is described as not just a simple form like value matching but also complicated form like a predefined java class or access to database located in the legacy system.

Figure 11 shows the RBPTS components and the internal message flow. We note that we revised the open source java class library, MANDARAX (SourceForge Inc., 2008b), in order to implement ECA-type rule engine, and XML Schema for ECA rule definition is newly defined as presented in Figure 12.

[7] http://ebxml.org
[8] http://www.w3.org

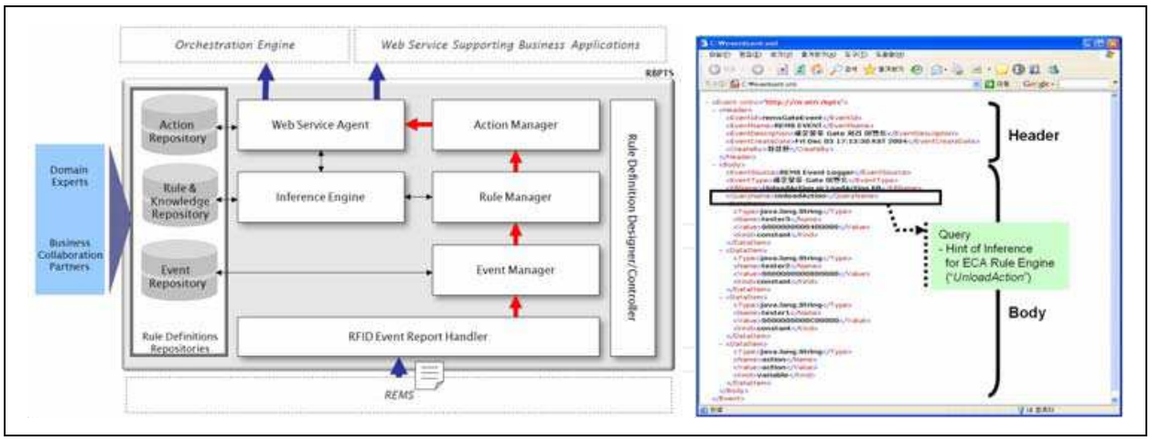

Fig. 11. Architecture of Real-time Business Process Triggering System and Sample Event XML Message over SOAP/HTTP delivered from REMS.

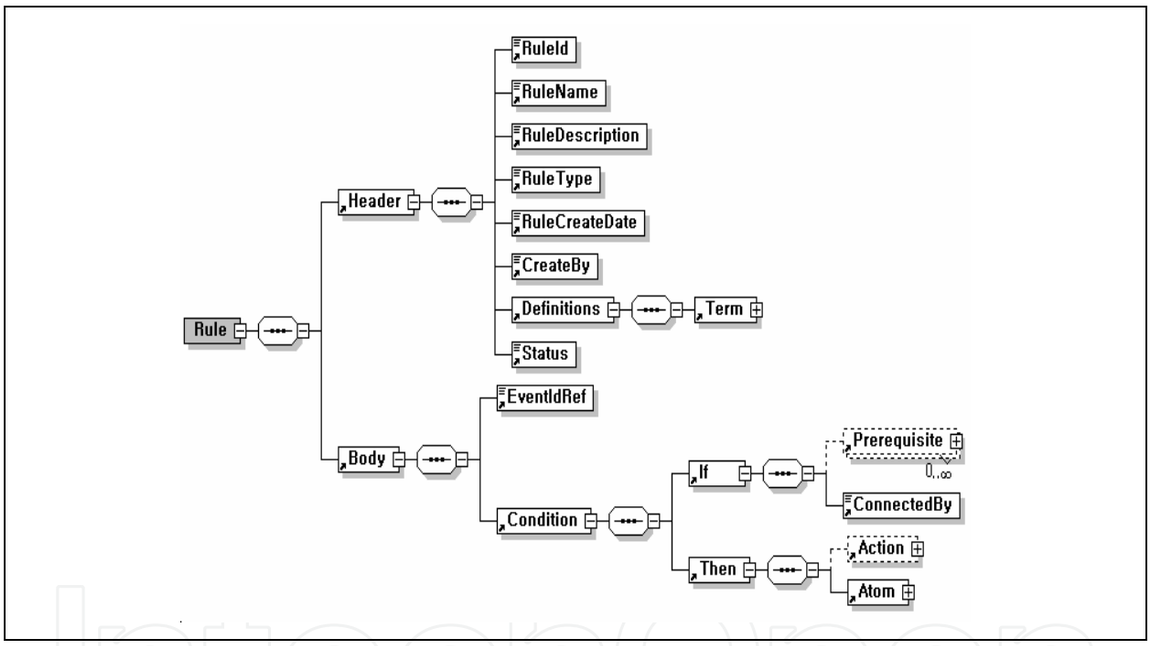

Fig. 12. XML Schema of Rule Definition.

REMS accumulates RFID tag data over intervals of time, filters to eliminate duplicate tag data and the tag data that are not of interest, and then reports in the XML/SOAP message form which follows the input format of RBPTS (see Figure 11). The RBPTS-specific event XML messages are generated by custom message dispatcher registered in REMS. RFID Event Report Handler accepts the SOAP message and then passes it to Event Manager in turn. Event Manager unmarshals the event message, checks whether the message is valid by looking up the event registry and checking its register status. Afterward, Event Manager reorganizes the valid event message into sort of event query message that is used for the next step - inference process - and delivers it to Rule Manager. Rule Manager inquires for the rule set associated with the event query and constitutes all the matters that are essential for the reasoning: database drivers that have access to the legacy database, repository

information and so on. Rule Manager feeds all the prepared materials into the Inference Engine and then this evaluates the conditions for each rule and generates the result set. This is during the process that business semantics are disclosed. Types of conditions span from simple forms including direct comparison to more complex ones such as inquiring to external applications. Based on the result set, Rule Manager organizes the action execution list and passes the list to Action Manager. Action Manager searches for the web service for each action execution and configures the information for the web service call. Action Manager asks for Web Service Agent to call the dynamic web service and records the execution result on the log database. RBPTS supports the application triggering via web service only.
In addition, RBPTS provides web-based user interface for rule design to model RFID event, related business rules and the detailed actions which in result provide more flexible way to adapt to the rapidly changing business environment.

4.4 Orchestration Engine (OE)

To build a concrete RFID middleware platform, it is desirable to orchestrate RFID-based end-to-end processes that associate with multiple applications or legacy systems so that it ultimately provides the RFID-enabled process automation environment.

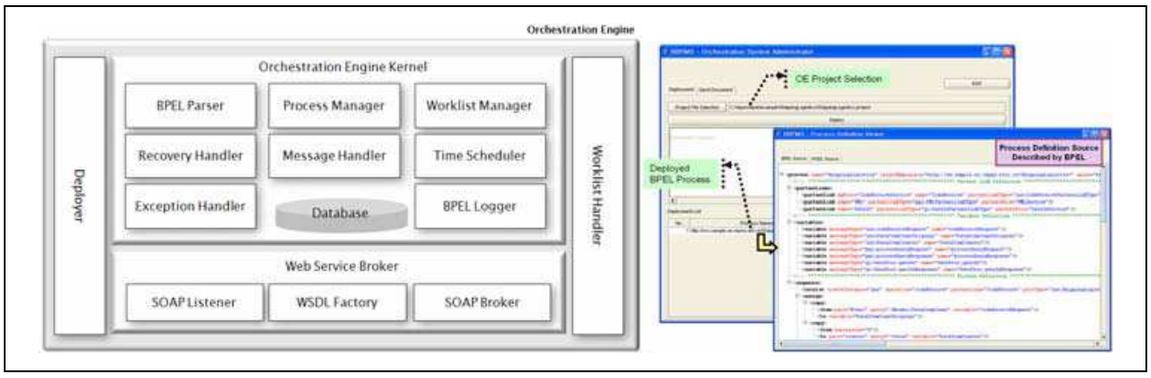

Fig. 13. Architecture of Orchestration Engine (OE) and GUI Window of OE Administrator.

For this purpose, the business process/workflow coordination engine called 'Orchestration Engine' is developed and the system architecture is shown in Figure 13. Currently, the most recent answer to the integration challenge is the Service Oriented Architecture (SOA) and the Web Service technologies. We suppose that we can access different functionalities of different legacy and other developed applications in a standard way through Web Services. Under the environment that all applications expose the functionalities via Web Services, we develop a business process definition and execution engine that provides a way to compose the Web Service-exposed functionalities in J2EE framework. Mostly, the business processes defined by Orchestration Engine are triggered by RFID-related events including not only the primitive event as the output of data transformation on the RFID IVC but also the semantic event generated by RBPTS.
In this implementation, we adopt BPEL (Business Process Execution Language for Web Services) (Andrews et al., 2003), an XML-based industry standard for business process management, as the definition language of business processes. BPEL builds on top of XML and Web Services, and BPEL process specifies the exact order in which participating Web

Services should be invoked. As the typical scenario we develop under the ETRI RFID Ecosystem, a BPEL business process receives a SOAP request from RBPTS when the raw observation by REMS are dispatched to RBPTS with XML event message and the rule instance, which is invoked by the message and contains the action clause of calling the BPEL process, is evaluated as true. Then, new instance is started and managed by Process Manager. It calls the external Web Services specified in the BPEL definition – for example, invoking EPCIS Web Service in order to get the prices of products and then invoking calculator Web Service to sum up the prices of items which are checked out – and returns the results when the process instance is done.

7. Conclusion

One of major issues in RFID software platform is how to (i) handle vast amount of RFID data efficiently and (ii) transform raw RFID tags to more valuable forms so that RFID-driven business applications can create value and achieve their ideal goals – automated business execution and integration. In this chapter, we introduce the concept of RFID Information Value Chain (RFID IVC), sequential activities for RFID data semantics evolution and value creation, and then discuss the several architectural considerations when we developed our RFID software platform called 'ETRI RFID Ecosystem ', whose middleware has features of component-oriented architecture, distributed and reliable operations, support of various types of RFID devices including external sensors, compatibility with existing RFID standards and business applications integration. We believe that we address the majority of the RFID software implementation concerns which other products should regard.

In addition, we demonstrate the RFID software platform with the internal service component architecture and the dependencies among components. It is presented how the discussed architectural considerations have been applied in order to construct such the RFID middleware. In fact, this artifact was tested in the field of several RFID pilot projects in South Korea including yard management project in harbour (Kim et al., 2006) and it showed the competitive advantages of ETRI RFID Ecosystem in the sense of the improvement in operational efficiency rather than partial deployment as desired.

8. References

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J. Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. & Weerawaran, S. (2003). Business Process Execution Language for Web Services version 1.1. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf

Clark, S., Traub, K., Anarkat, D. & Osinski, T. (2003). Auto-ID Savant Specification 1.0, *MIT Auto-ID Center working paper*

Dutta, K., Ramamritham, K., Karthik, B. & Laddhad, K. (2007). Real-time Event Handling in an RFID Middleware System, *Proceedings of Workshop on Databases in Networked Information Systems,* pp. 232-251, ISBN 978-3-540-75511-1, Japan, October 2007, Springer

EPCglobal (2005a). EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz Version 1.0.9. http://www.epcglobalinc.org

EPCglobal (2005b). The Application Level Events (ALE) Specification, Version 1.0. http://www.epcglobalinc.org

EPCglobal (2006). Reader Protocol Standard, Version 1.1. http://www.epcglobalinc.org

EPCglobal (2007a). Reader Management 1.0.1. http://www.epcglobalinc.org

EPCglobal (2007b). EPC Information Services (EPCIS) Version 1.0, Specification. http://www.epcglobalinc.org

EPCglobal (2008a). The Application Level Events (ALE) Specification, Version 1.1. http://www.epclobalinc.org

EPCglobal (2008b). EPCglobal Object Name Service (ONS) 1.0.1. http://www.epclobalinc.org

Floerkemeier, C., Roduner, C. & Lampe, M. (2007). RFID Application Development With the Accada Middleware Platform. *IEEE Systems Journal*, Vol. 1, No. 2, December 2007, pp. 82-94

Gonzalez, H., Han, J., Li, X. & Klabjan, D. (2006). Warehousing and Analyzing Massive RFID Data Sets, *Proceedings of the 22nd International Conference on Data Engineering*, pp. 83-92, ISBN 0-7695-2570-9, Atlanta, USA, April 2006, IEEE Computer Society, USA

Gupta, A. & Srivastava, M. (2004). Developing Auto-ID Solutions using Sun Java System RFID Software. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html

IBM. (2006). *IBM WebSphere RFID Handbook: A Solution Guide*. http://www.redbooks.ibm.com/redbooks/pdfs/sg247147.pdf

ISO/IEC JT1/SC31/WG4. http://usnet03.uc-council.org/sc31/sc31_wg5.cfm

Kim, Y., Yoo, J. & Park, N. (2006). RFID Based Business Process Automation for Harbor Operations in Container Depot, *Proceedings of Industrial & Manufacturing Engineering Graduate Research Symposium at Wayne State University*, pp. 213-226, Detroit, USA, April 2006

Ku, T., Zhu, Y. & Hu, K. (2007). Semantics-based Complex Event Processing for RFID Data Streams, *Proceedings of First International Symposium on Data, Privacy and E-Commerce*, pp. 32-34, ISBN 978-0-7695-3016-1, Chungdu, China, November 2007, IEEE Computer Society, USA

Loritsch, B. (2001). Developing with Apache Avalon. *Apache Software Foundation*

Luo, Z., Wong, E., Cheung, S.C., Lionel, M. & Chan, W.K. (2006). RFID Middleware Benchmarking, *Proceedings of the 3rd RFID Academic Convocation*, Shanghai, China, October 2006

Mort Bay Consulting. (2008). *Jetty – Java HTTP Servlet Server*. http://www.mortbay.org/jetty/.

Oracle. (2008). *Oracle RFID and Sensor-based Services.* http://www.oracle.com/technologies/rfid/index.html

Oat Systems & MIT Auto-ID Center. (2002). The Savant Version 0.1 Alpha, *MIT Auto-ID Center Technical Manual*

Palmer, M. (2006). RFID and Complex Event Processing, *RFID Today*, http://www.rfidtoday.co.uk/articles/objectsore_rfid.html

Sarma, S.; Brock, D.L. & Ashton, K. (2001). The networked physical world proposals for engineering the next generation computing, commerce & automatic-identification. *MIT Auto-ID Center white paper*

Sun Microsystems Inc. (2006). *Sun Java System RFID Software 3.0 Developer's Guide*. http://docs.sun.com/source/819-4686/ale-web-services.html
Sun Microsystems Inc. (2008). *Jini Network Technology*. http://www.sun.com/software/jini/
SourceForge Inc. (2008a). *Avalon – Framework branch*. http://freshmeat.net/projects/avalon/?branch_id=18263
SourceForge Inc. (2008b). *Mandarax Project*. http://mandarax.sourceforge.net/
Tibco Software Inc. (2006). RFID Solution. http://www.tibco.com/solutions/biztech/rfid.jsp

Development and Implementation of RFID Technology

Edited by Cristina Turcu

ISBN 978-3-902613-54-7
Hard cover, 450 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2009
Published in print edition January, 2009

The book generously covers a wide range of aspects and issues related to RFID systems, namely the design of RFID antennas, RFID readers and the variety of tags (e.g. UHF tags for sensing applications, surface acoustic wave RFID tags, smart RFID tags), complex RFID systems, security and privacy issues in RFID applications, as well as the selection of encryption algorithms. The book offers new insights, solutions and ideas for the design of efficient RFID architectures and applications. While not pretending to be comprehensive, its wide coverage may be appropriate not only for RFID novices but also for experienced technical professionals and RFID aficionados.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen