

rXPDL: An XML-based Policy Definition Language for Policy-driven RFID Event Management Agents

Hyun Ahn*, Bumjo Park*, Wonbo Shim*, Namgi Kim*, Heeyoul Kim*, Kwanghoon Kim*
Youngyoun Ahn**, Dongkeun Oh**, Eunsoo Kim**

*Dept. of Computer Science, KYONGGI UNIVERSITY, Suwonsi Kyonggido, Korea

**RFID Research Laboratory, IC & IT, LTD., Seoul, Korea

{hyun.ahn, parkbumjo}@kgu.ac.kr, miniyusi@hotmail.com, {ngkim, heeyoul.kim, kwang}@kgu.ac.kr
yyahn@icnit.co.kr, dkoh@icnit.co.kr, eskim@icnit.co.kr

Abstract—In this paper, we focus on RFID middleware's event management functionality that provides a means of specifying event-constraints for filtering a huge number of raw data caught from the associated RF readers. However, the current specifications are too circumstantial to be understood by the application developers who used to lack the professional and technological backgrounds of the RFID middleware functionality. As an impeccable solution, we propose the concept of event-constraint policy that is not only representing semantic contents of RFID domains but also providing transparencies with higher level abstractions for RFID applications. We also embody the proposed concept in an XML-based RFID event policy definition language, which is abbreviated to rXPDL. Finally, we expect that the application developers are able to easily specify their RFID applications' event-constraints through the proposed concept, and that the concept is also able to be extensively applied to other *ad hoc* sensor networks.

Keywords—RFID middleware, *ad hoc* sensor network middleware, RFID event constraints policy, RFID event policy XML schema

I. INTRODUCTION

The purposes of this paper is to enhance the understandability of RFID middleware for the application developers and is to allow them to easily define their RFID event-constraints by providing a high-level abstraction from their point of view, not from the RFID readers' and tags developers'. Also, in spite of that the RFID application arena has a vast published literature, there still exists a confusing array of terms and meanings. Therefore, in describing the RFID middleware and its applications, we would use the basic terminology of RFID application, RFID service and RFID task, and also we assume to follow the standards' terminology defined in the EPCglobal architectural framework[1][4][2][3], ISO/IEC's SSI¹ architectural framework[9][10], and others.

The conventional RFID middleware architectural frameworks provide a certain level of interfaces for RFID applications, which are embodied in an interfacing component between the RFID applications and the internal blocks of the frameworks, such as device interface block, data management block, and device management block. However, the embodied interfacing component and its related standards are

too circumstantial to be understood by the RFID application developers, even though the corresponding RFID applications do not need to specify and understand the details of them. To the application developers, actually it is much more important to provide a highly abstracted interfacing component for specifying and expressing their applications' semantic contents with some reasonable transparencies. A little more specifically speaking, it might be no doubt that the current EPCglobal's ALE (application level events) specification and the ISO/IEC 24791 SSI Part 4's AI (application interface) are unsatisfied with the crucial requirement, *providing a means for specifying applications' semantic contents with a reasonable transparency*. Therefore, in order to resolve the difficulties, several approaches[7][6][11][12][8] have been proposed in the literature. One of the approaches is the policy-driven RFID event management framework and its agents[8] that is able to reasonably embodying the interfacing component between the RFID applications and the internal blocks of the conventional RFID middlewares. In this paper, we particularly describe the details of the policy definition language and its XML schema, which is the essential buildtime component of the framework.

The remainders of this paper are organized as follows. The next section briefly describes about the motivation and scope of this paper. Section 3 works out the concept of policy and its definition language, and finalizes with an example applying the language. And in the section 4 we describe the literature survey related with the topic of the paper.

II. MOTIVATION AND THE SCOPE

As stated in the previous section, we already have a framework for the policy-driven interfacing component for the RFID applications. Out of the interfacing component's functions, the most essential one is the RFID event definition function that is to describe event-constraints filtering a huge number of raw data caught from the associated RF readers. So, the interfacing component being charged with the event-constraints definition functionality is so-called a RFID policy definition language formatted in XML, which is abbreviated to rXPDL. Therefore, the scope of the paper is to work out the rXPDL's schema and to show its usability and feasibility.

As illustrated in Fig. 1, the proposed language is based upon the policy-driven RFID event management agent that

¹ISO/IEC JTC 1/SC 31/WG 4 SSI (System Software Infrastructure)

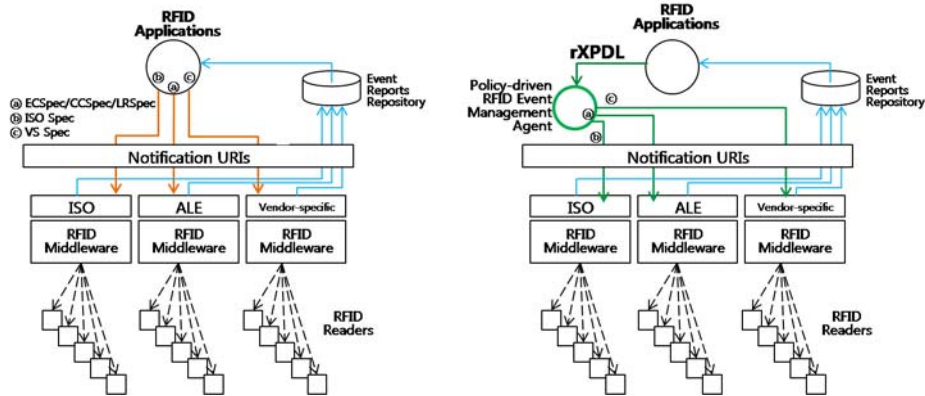


Fig. 1. The Basic Motivation of the Proposed Language

is to newly implement from the policy-driven RFID event management framework[8]. The agent takes complete charge of the interpretation of highly abstracted service policy written in rXPDL as well as the transformation of the received rXPDL-based policy into the corresponding RFID event specifications of the underlying systems like ALE and others. Conclusively, through the proposed language, it is reasonably possible for the application developers to be able to easily define their event-constraints services without acquainting with the details of the conventional approaches, and which is the main motivation of the paper.

III. THE PROPOSED LANGUAGE

Based upon the motivation stated in the previous section, we try to conceive an XML-based description language supporting the concept of RFID event management policy, and work out the details of the language, such as design principles, metamodel, schema, and its application, in this section.

A. Design Principles

The existing REID event-constraints descriptions supporting EPCglobal ALE², ISO/IEC 24791-2³ and the other vendor-specific (non-standards) specifications have been suffering the deficiency issues of easiness and transparency. In other words, these are becoming the design principles that the proposed language should satisfy. The easiness issue is directly related with the degree of complexity on the RFID event-constraints specifications. At present, it is hard to say that almost all RFID event-constraints definition tools provide an easy way of the event cycle definition interfaces so as for the novice users to be able to simply and conveniently use. Therefore, it is very important to provide a user-friendly definition methodology with intuitive interfaces.

And, the transparency issue has something to do with the differences among the conventional de facto standards for RFID event-constraints descriptions and specifications. That is to say, the RFID application developers and its managers should understand the details of each of the corresponding

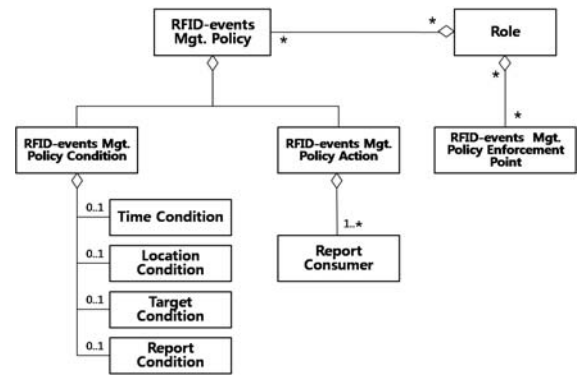


Fig. 2. A Meta-Model of rXPDL

standards, and it means, however, that understanding everything is understanding nothing. Therefore, it is very important to provide a certain level of transparencies to the RFID application developers and the RFID middleware managers in order for them to define and place their event cycles through a certain type of unified and middleware-independent interfaces.

B. rXPDL Metamodel

In designing a policy description language, it is important to have a metamodel that describes all possible components and their relationships (association, inclusion and grouping) that are needed to build a RFID event management policy definition model. Also, the buildtime system used to build the model is based on the metamodel. The meta-model is shown in Fig. 2. As you can see in the meta-model for rXPDL, a RFID event management policy model is defined by associating conditions, actions and roles that are represented by the condition types of time, location, target and report, the report consumer, and the enforcement point, respectively. In terms of the tool's functionality, basically it is necessary to be equipped with a policy repository, a relevant information repository and a policy verifier and analyzer.

²Application Level Events Spec.

³ISO/IEC JTC 1/SC 31/WG 4 SSI Part 2: Data Management

C. rXPDL Schema

Based upon the meta-model, we build an XML schema for the RFID event-constraints definition language. The schema is hierarchically structured and syntactically represented in the Backus Naur Form. A policy is defined by a set of actions and conditions. The actions are activated by one of the types of URI (Universal Resource Identifier) - HTTP, JMS, TCP and File, while the conditions are defined by four properties - time, location, target and report. Conclusively the followings are the whole details of the schema:

(1) Policy Definition Schema

```

0: < POLICY-STATEMENT >
1:   ::= Policy {
2:     < POLICY-NAME >
3:     < DESCRIPTION >
4:     < POLICY-ROLES >
5:     < POLICY-CONDITIONS >
6:     < POLICY-ACTIONS >
7:   }
8: < POLICY-CONDITIONS >
9:   ::= PolicyConditions {
10:    < CONDITION-OP-TYPE >
11:    < POLICY-CONDITIONS-DETAILS >
12:  }
13: < POLICY-ACTIONS >
14:   ::= PolicyActions {
15:    < ACTION-OP-TYPE >
16:    < POLICY-ACTIONS-DETAILS >
17:  }

```

(2) Policy-Conditions Definition Schema

```

0: < POLICY-CONDITIONS-DETAILS >
1:   ::= <> | < POLICY-CONDITION-STATEMENT >
2:   | < POLICY-CONDITIONS-DETAILS >
   < POLICY-CONDITION-STATEMENT >
3: < POLICY-CONDITION-STATEMENT >
4:   ::= PolicyCondition {
5:    < POLICY-CONDITION-NAME >
6:    < DESCRIPTION >
7:    < TIME-CONDITION-STATEMENT >
8:    < LOCATION-CONDITION-STATEMENT >
9:    < TARGET-CONDITION-STATEMENT >
10:   < REPORT-CONDITION-STATEMENT >
11:  }

```

(3) Policy-Actions Definition Schema

```

0: < POLICY-ACTIONS-DETAILS >
1:   ::= < POLICY-ACTION-STATEMENT > |
2:   < POLICY-ACTIONS-DETAILS >
   < POLICY-ACTION-STATEMENT >
3: < POLICY-ACTION-STATEMENT >
4:   ::= PolicyAction {
5:    < POLICY-ACTION-NAME >
6:    < DESCRIPTION >
7:    < REPORT-CONSUMERS >
8:    < ACTION-NOTI-URI >
9:    < ACTION-FUNCTION >
10:  }
10: < POLICY-ACTION-NAME > ::= policyActionName =
<NAME>;
11: < REPORT-CONSUMERS > ::= reportConsumer = <
CONSUMERS >;
12: < ACTION-NOTI-URI > ::= actionNotiURI = <
URI-TYPE >;
13: < ACTION-URI > ::= actionURI = < URI >;
14: < CONSUMERS > ::= <> | < CONSUMER >
| < CONSUMERS >, < CONSUMER >
15: < URI-TYPE > ::= < HTTP-RECEIVER-URI > |
< JMS-RECEIVER-URI > |
< TCP-RECEIVER-URI > |
< FILE-RECEIVER-URI >

```

(4) Time Condition Definition Schema

```

0: < TIME-CONDITION-STATEMENT >
1:   ::= <> | PolicyTimeCondition {
2:    < TIME-CONDITION-NAME >
3:    < DESCRIPTION >
4:    < START-TIME >
5:    < FINISH-TIME >
6:    < EVENT-CYCLE >
7:  }

```

```

8: < TIME-CONDITION-NAME > ::=
timeConditionName=<NAME>;
9: < START-TIME > ::= <> | startTime = <DATETIME>;
10: < FINISH-TIME > ::= <> | finishTime =
<DATETIME>;
11: < EVENT-CYCLE > ::= <UNSIGN-INTEGER>

```

(5) Location Condition Definition Schema

```

0: < LOCATION-CONDITION-STATEMENT >
1:   ::= <> | PolicyLocationCondition {
2:    < LOCATION-CONDITION-NAME >
3:    < DESCRIPTION >
4:    < INCLUDE-TYPE >
5:    < LOCATIONS >
6:  }
7: < LOCATION-CONDITION-NAME >
   ::= locationConditionName = < NAME
>;
8: < INCLUDE-TYPE > ::= includeType =
<INCLUDE-EXCLUDE>;
9: < INCLUDE-EXCLUDE > ::= INCLUDE | EXCLUDE
10: < LOCATIONS > ::= <>
| locationSet = {<LOCATION-SET>};
11: < LOCATION-SET > ::= <> | < LOCATION >
| < LOCATION-SET >, < LOCATION >
12: < LOCATION > ::= any sequence of unicode
characters starting with a letter of location
name

```

(6) Target Condition Definition Schema

```

0: < TARGET-CONDITION-STATEMENT >
1:   ::= <> | PolicyTargetCondition {
2:    < TARGET-CONDITION-NAME >
3:    < DESCRIPTION >
4:    < INCLUDE-TYPE >
5:    < TARGETS >
6:  }
7: < TARGET-CONDITION-NAME >
   ::= targetConditionName = < NAME >;
8: < INCLUDE-TYPE > ::= includeType =
<INCLUDE-EXCLUDE>;
9: < INCLUDE-EXCLUDE > ::= INCLUDE | EXCLUDE
10: < TARGETS > ::= <>
| targetSet = {<TARGET-SET>};
11: < TARGET-SET > ::= <> | < TARGET >
| < TARGET-SET >, < TARGET >
12: < TARGET > ::= any sequence of unicode
characters starting with a letter of target
name

```

(7) Report Condition Definition Schema

```

0: < REPORT-CONDITION-STATEMENT >
1:   ::= <> | PolicyReportCondition {
2:    < REPORT-CONDITION-NAME >
3:    < DESCRIPTION >
4:    < REFINEMENT-TYPE >
5:    < ACCOUNT-TYPE >
6:  }
7: < REPORT-CONDITION-NAME >
   ::= reportConditionName = < NAME >;
8: < REFINEMENT-TYPE > ::= refinementType =
<REFINEMENT-VALUE>;
9: < REFINEMENT-VALUE > ::= CURRENTS | ADDITIONS |
DELETIONS
10: < ACCOUNT-TYPE > ::= <>
| accountType = {< ACCOUNT-VALUE >};
11: < ACCOUNT-VALUE > ::= ACCOUNT | UNACCOUNT

```

D. rXPDL Example

In order to prove the soundness of the proposed language, we try to give a possible example with a hypothetical scenario. The scenario assumes that we implement a security service for managing facilities in three laboratories by using RFID technology, and through this scenario we expect to give much more detailed comprehension of the rXPDL schema and its usage. Let's imagine a research institute consisting of three laboratories (LAB1, LAB2 and LAB3).

Then, an administrator in the institute should define a relevant policy specification for deploying the security service to the facilities in the three buildings. By using a RFID event

management policy modeling tool, he/she is able to easily define a policy specification of the security service without recognizing the details of RFID resources like RF-tags, devices, RFID middlewares and even the application's domain knowledge. The following is an example of the security policy specification according to the schema/meta-model of rXPDL:

```

0: Policy {
1:   policyRoles = 'Facility Security Service';
2:   policyName = 'SecurityPolicy';
3:   PolicyConditions {
4:     PolicyCondition {
5:       PolicyConditionName = 'SecurityPolicyCondition';
6:       PolicyTimeCondition {
7:         PTiCName = 'securityPolicyTimeCondition';
8:         eventCycle = '1000'; }
9:     PolicyLocationCondition {
10:      PLCName = 'securityPolicyLocationCondition';
11:      locationSet = { 'LAB1', 'LAB2', 'LAB3' }; }
12:     PolicyTargetCondition {
13:      PTaCName = 'securityPolicyTargetCondition';
14:      includeType = 'INCLUDE';
15:      targetSet = { 'ALL' }; }
16:     PolicyReportCondition {
17:      PRCName = 'securityPolicyReportCondition';
18:      refinementType = 'DELETIONS'; } }
19: }
20: PolicyActions {
21:   PolicyAction {
22:     policyActionName = 'securityPolicyAction';
23:     reportConsumer = 'ADMINISTRATOR';
24:     actionNotiURI = 'HTTP-RECEIVER-URI';
25:     actionURI = 'http://www.securityServiceSystem.net/
        FSS2009//CheckLaboratory.php?
        id=180&password=fa9f95'; }
26: }
27: }

```

After defining the security policy specification, the administrator registers the policy into an policy resource repository of the RFID-based facilities security management system. The snapshot of the implemented policy specification for the example is shown in Fig. 3.

IV. RXPDL-BASED POLICING AGENT

As described in the scenario of the RFID-based facilities security management system, once the policy is defined and then it will be enacted by a runtime framework. Fig. 4 illustrates a possible configuration of the runtime framework. That is, in order to process the scenario's policy specification, a RFID event policy enactment agent is assigned, and the corresponding policy enforcement point compiles the policy and transform into one of the conventional RFID event management specifications like EPCglobal ALE, ISO/IEC SSI Part 2 standard or other vendor-specific specifications. Of course, the RFID event policy enactment services' server in the runtime framework communicates with the policy enforcement points through a certain type of message exchanging protocol, which however is not handled in this paper. Finally, the transformed scenario's specification forms a policy message and is transmitted into a policy client through the policy message exchange protocol, and eventually the policy is enforced into the corresponding RFID middleware (the policy client).

Summarily, the buildtime component (the RFID event policy modeling system) processes policy specifications formatted in rXPDL, while the runtime components (the RFID event policy enactment agents and the enforcement points) communicate each other by a policy-message exchanging protocol. The language, rXPDL, is a means to provide an unified interface

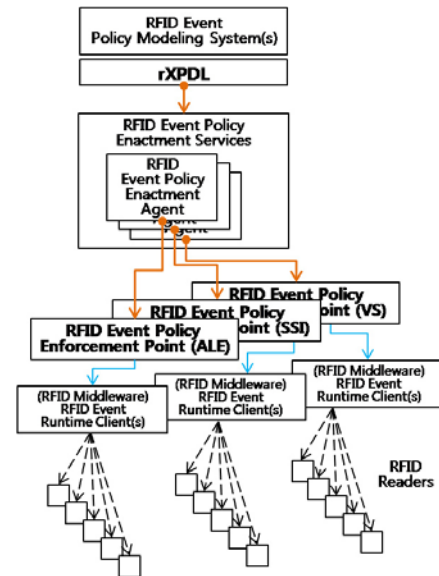


Fig. 4. A rXPDL-based Policing Agent

to the conventional RFID events processing systems, each of which complies with its own RFID events management specifications including various international and de facto standards, and it also is characterized by an abstracted and graphical specification language coping with EPCglobal ALE, ISO/IEC 24791-2 and other vendor-specific specifications. Based on this proposed language, the application developers are able to easily define and automatically deploy RFID event management policies through an intuitive and graphical modeling methodology.

V. RELATED WORKS

So far, there exist several researches related with this topic for easily specifying RFID event-constraints by furnishing with a high-level abstraction, which is generally called the rule-based RFID service management frameworks and systems. That is, Wang[7] proposed the rule condition and method of action definition for RFID service integrations according to the specific events obtained from the context elements and event refinement declarations with respect to time, location and target. His approach also supported the negative condition syntax related to the tense constraints. However, the declaration part of the event refinement method takes into account just time, location and target, and did not support the method of report conditions that are considered in this paper. Therefore, his approach may be unsuitable for the service domains possibly generating a huge amount of events. Xu[6] suggested the operation method. The method detects from several RFID readers by applying the Manchester coding algorithm after allocating a collection pattern about a large amount of low events, and then it makes running through query retrievals for the corresponding business rules that are registered previously after analyzing some derivations between patterns of relevant events. However, the proposed method has some disadvantages in terms of that it needs a certain amount

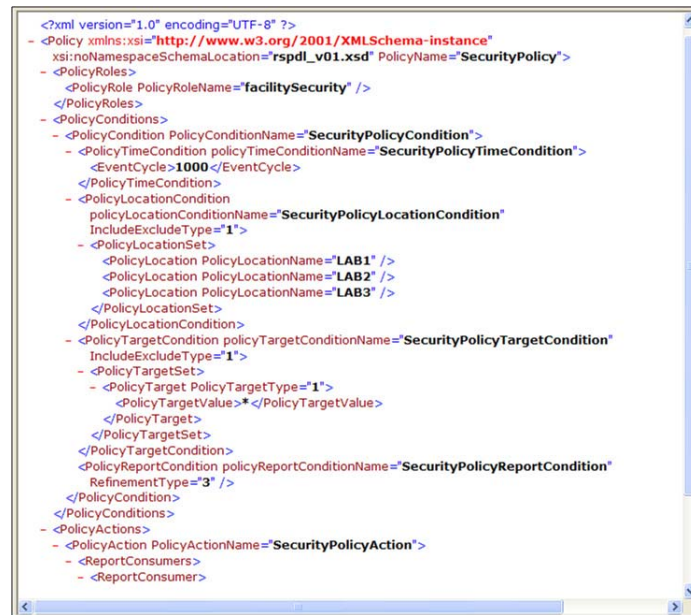


Fig. 3. A Snapshot of the Implemented Policy Specification

of high-capacity storage in order to analyzing patterns in a large amount of low events.

Additionally, Moon[11] proposed the CEL (Contextual Event Language) description language to define RFID events, which is satisfied with the registered event cycle specifications such as the event refinement specifications and rules. However, the proposed method may have some limitations in terms of that the users ought to completely understand the structures of the event cycle specifications in order to define the rule conditions. Finally, Kim[12] proposed the description method that can define not only the condition syntaxes of business rules as BESpec (Business Event Specification), but also the corresponding event refinement specifications according to the previously registered event cycle specifications. And it can integrate the performance results of BESpec into a composite web service defined in BEReport (Business Event Report). However, the users should understand the structures of ECSpecs and ECREports in order to define the BESpec.

VI. CONCLUSIONS

So far, this paper has presented a highly abstracted RFID event description language, which is called rXPDL, and also its meta-model and XML schema. Additionally, we have given the description of a runtime framework to show the language's feasibility and usability. Particularly, in this paper we pointed out the limitations of the current RFID event management processing frameworks and standardized specifications, and proposed a feasible solution for resolving the limitations through the concept of policy and its related details.

Conclusively, the XML-based RFID event policy language is able to convey more convenient, intelligent and broad-ranged RFID services to the RFID application developers and managers. It also is strongly believed that the proposed language and its runtime framework should be an impeccable

solution for realizing the useful, effective, flexible, and extensible RFID events management systems.

VII. ACKNOWLEDGMENTS

This work was funded by the TTA's standardization promotion project, 2010-P1-40E, supported by the Ministry of Knowledge and Economy, South Korea.

REFERENCES

- [1] EPCglobal. Inc, "EPC Information Services (EPCIS) Version 1.0.1 Specification", 2007.
- [2] EPCglobal. Inc, "The Application-Level Events(ALE) Specification, version 1.1 - Part 1: Core Specification", 2008.
- [3] EPCglobal. Inc, "The Application-Level Events(ALE) Specification, version 1.1 - Part 2: XML Bindings", 2008.
- [4] EPCglobal. Inc, "EPCglobal Object Name Service (ONS) 1.0.1, Ratified Standard Specification with Approved", 2008.
- [5] S. Wamba, L. A. Lefebvre, E. Lefebvre, "Enabling intelligent B-to-B e-Commerce supply chain management using RFID and the EPC network: a case study in the retail industry", ACM International Conference Proceeding Series Vol. 156, 2006.
- [6] J. Xu, W. Cheng, W. Liu, W. Xu, "XML Based RFID Event Management Framework", The Proceedings of IEEE Region 10th Conference TENCON 2006, Vol. 1, pp.1-4, 2006.
- [7] F. Wang, S. Liu, R. Liu, Y. Bai, "Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams", The Proceedings of Advances in Database Technology - EDBT 2006, Lecture Notes in Computer Science, Vol. 3896, pp.588-607, 2006.
- [8] H. Ahn, et. al, "A Policy-driven RFID Event Management Framework", The Proceedings of ICUT2009, IEEE Computer Society, 2009.
- [9] ISO/IEC JTC 1/SC 31/WG 4 24791-2, "System Software Infrastructure: Data Management Specification", 2008.
- [10] ISO/IEC JTC 1/SC 31/WG 4 24791-4, "System Software Infrastructure: Application Interface Requirements", 2008.
- [11] M. Moon and et al., "Contextual Events Framework in RFID System," The Proceedings of 3th International Conference on Information Technology: New Generations, pp. 586-587, IEEE Computer Society, 2006.
- [12] S. Kim and et al., "RFID Business Aware Framework for Business Process in the EPC Network," The Proceedings of 5th International Conference on Software Engineering Research, Management and Applications, pp. 468-475, IEEE Computer Society, 2007.