Abstract—With the proliferation of smart computing equipment, the range of providing highly intelligent services is widely expanding in an attempt to include the present condition of each individual data source. Furthermore, broadly adopted RFID (Radio Frequency Identification) technology has led to the widespread application of automatic identification in numerous pervasive computing services including logistics, emergency control, and medical services. Another remarkable phenomenon is the explosively growing number of smartphone users, which allows them to stay connected to the Internet. The environment created by these factors, increases the need to make a decision out of the current context which can be inferred from various sensors and mobile devices. In response to the growing demand for the need to make a decision in the current context, the new XOnt Agent has been integrated into our collaboration middleware framework, the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware). All the collected information should be examined too see if they meet any condition that triggers correspondent actions of a specific rule through reasoning process of a rule engine. We also suggest the context-aware inference (CAI) model to effectively support the reasoning and handling processes.

Keywords-Pervasive/Smart Computing; XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware); CAI (Context-Aware Inference) model; RFID (Radio Frequency Identification); USN (Universal Sensor Network); Collaboration; Context-Awareness; Ontology; Rule engine

I. INTRODUCTION

Rapid progress in wireless telecommunication and sensing facilities based on RFID/USN technology encourages us to apply the advanced smart computing capabilities to the existing IT solutions. It also allows us to introduce more elaborate and advanced services to both associated users and developers. Under this environment, the current IT systems are more likely to collaborate with each other and maximize the synergy effect from the related service systems. To do this, it is required to resolve complicated issues raised by many interested parties and suggest effective ways to make the parties interact with each other, which can be done by making direct connections among numerous parties. But creating direct connections imposes a heavy burden on developers because they have to maintain the mesh of connections between different parties and the infrastructure system itself.

In order to provide end users with highly sophisticated services of interconnected applications, we need to consider a new type of orchestrating service framework, which collects a large amount of sensor data from many data sources and delivers real-time data to each application service. The framework must be a robust and reliable infrastructure and be able to handle a huge amount of tag and sensor data and propagate them to the appropriate services according to predefined scenarios. These requirements led us to develop a scenario-based collaborative framework, the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware) framework, which seamlessly integrates numerous heterogeneous application services and plays an organizing role in today's pervasive computing environment [10], [11].

The XCREAM is placed between RFID/USN middlewares and application services as in “Fig. 1.” It collects the ECReports from RFID/USN middlewares [14], interprets them and finally delivers them to the correspondent user-defined business application services in order to establish an integrated cooperative working environment.
The XCREAM framework allows us to develop RFID/USN-enabled applications easily as it acts like a bridge between physical sensors and application services. By using the XCREAM as a bridge, most of sensor specific details are migrated to the XCREAM framework from individual application services.

Furthermore, smartphones allow the users to stay connected to the Internet almost every minute and the pervasiveness of the sensor devices gives them the opportunity to use the information from the sensor devices. These factors increase the need to process a huge number of data transactions and infer the present situation out of the current context. The factors that are included in the current context are the owner’s current location, status values of various sensors, media preferences, community accessibility, and so on.

The agent observes all incoming events, finds certain context and triggers the appropriate service(s). For example: when a handicapped passenger with RFID-embedded boarding pass is approaching a boarding gate, the airline employee is already aware of his/her arrival and is ready for the passenger.

This paper is composed of 5 chapters in total. Chapters 1 and 2 present the introduction and the related research. Chapter 3 describes the overall architecture of our extended framework and the internal design specification of the XOnt agent. In Chapter 4, the CAI model is described based on situations that might arise in an airport. Chapter 5 concludes with future research direction.

II. RELATED RESEARCH

A. RFID Technology

There have been many researches related to the proposed system: RFID-enabled auto-identification technology for the data collection and automatic recognition of the numerous events at the front-end of the framework; continuous event and query processing technology for the massive events stream; and USN middleware technology, which many similar systems adopted in the fields [5], [7].

A lot of research efforts made during the past decade aimed at defining the RFID tag specification and handling the RFID tag data. Past research also focused on formalizing the protocols required for seamless communication between related components and enhancing the performance and correctness of the system itself.

Initially, RFID-enabled network infrastructure was proposed by the Auto-ID Labs [12] at MIT. The labs were dedicated to creating the Internet of Things using RFID and Wireless Sensor Networks. The labs evolved into EPCglobal which developed ubiquitous automated identification technologies based on the networked physical world by proposing the EPC Network. The EPC network identifies goods using a unique numbering system called an Electronic Product Code (EPC) number by enabling all objects in the world to be linked via the Internet [5], [14]. EPCglobal also suggested an open architecture system which is composed of the EPC, EPC tags and readers, local networking technology, and RFID middleware [14], [15]. Local networking technology allows readers and sensors to be connected via local databases. RFID middleware collects and filters massive tag data, aggregates and processes them into meaningful information, and delivers them to the pertinent applications. The middleware may use the Object Name Service (ONS) similar to the Domain Name Service (DNS) in the Internet for location lookups of specific items [4].

B. Context-Awareness

Many researchers have defined “context,” on their own. The term, “context-aware,” was introduced by Schilit and Theimer (1994) for the first time. The authors described context as location, identities of nearby people, objects and changes to those objects. After three years, Ryan (1997) referred to context as the user’s location, environment, identity, and time. One of the most accurate definitions was given by...
Dey and Abowd in 2000, who defined context as “any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves” [2].

As various kinds of event data and services interact with each other, it is natural that a single event relates to numerous services and several different events are relevant to a specific context. This means that the framework can face many-to-many connections between physical devices and cyber application services.

The XOnt Agent, has been added to the XCREAM framework in order to identify valid context information from numerous incoming events. Once the XOnt Agent receives various events from multiple sources, a tightly coupled rule engine, called Drools [13], decides whether a certain context has been met by evaluating the pre-registered rules with the newly received events. Drools can also make this decision by combining the newly received events with the existing events. [3].

XCREAM increases the overall flexibility of the framework, by allowing new physical sensors and its related application services to be integrated seamlessly.

C. Ontology

The current computing environment contains lots of smart devices working with its corresponding application services. This situation requires us to devise a communication scheme between them that includes a flexible interface and common vocabulary.

The XOnt Agent includes the XOntology [8] as a context representation scheme which is shared by related application services. The original events and the resulting context within the framework should be understandable by related parties and to achieve this, an ontology scheme is applied as a mutual communication method in the XCREAM framework in combination with the context-awareness scheme. The integration of ontology scheme not only allows applications to share contextual information based on the common ontology specifications, but also increases the collaboration rate between application services.

First of all, the underlying XCREAM framework needs to understand various kinds of context information according to a common context description scheme in order to enhance the effectiveness in representing and sharing the information.

Second, RDF (Resource Description Framework) and OWL Web Ontology Language of the Semantic Web standards needs to be applied to the common context description scheme of the framework in order to represent context information. According to the World Wide Web Consortium (W3C), “The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.” The Semantic Web is to make automated semantic agents access the Web more intelligently and carry out sophisticated tasks based on the semantics encoded into the Web page on behalf of the users [9].

In the Semantic Web, we regard things in the world as resources: resources can be anything that someone might want to talk about and is easily understood as a thing or an entity [1]. In this situation, a user’s current location, real-time traffic information, environmental information like temperature, bioinformation like one’s blood pressure, and a RFID tag attached to a traveler’s luggage are all examples of resources.

<table>
<thead>
<tr>
<th>Rule Engine</th>
<th>Drools</th>
<th>Jess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>RETE Algorithm</td>
<td>RETE Algorithm</td>
</tr>
<tr>
<td>OWL-DL Entailment</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Java Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rule Support</td>
<td>DRL (Own Rule Format)</td>
<td>SWRL</td>
</tr>
<tr>
<td>Version</td>
<td>5.4</td>
<td>7.1</td>
</tr>
<tr>
<td>Licensing</td>
<td>Free / Open source</td>
<td>Academic use only</td>
</tr>
</tbody>
</table>

D. Rule-based Inference Engines

There are many kinds of rule engines available. We chose to study Drools and Jess and summarized their key features in the following table. The table gives a comparative chart for the selected inference engines measured on different performance metrics.

- Drools

Drools is a java-based object-oriented rule engine, which is an open-source. Therefore, we can freely use and modify the codes in java. Drools uses an optimized version of the RETE Algorithm [6], called RETE-Object-Oriented algorithm to give high performance. It has its own writing rules called DRL (Drools Resource Language) and is flexible enough to match the semantics of a problem domain with DSLs (Domain Specific Languages), graphical editing tools, web based tools and developer productivity tools. It has useful features which include rule debugging and rule authoring tools like IDE plug-in [13].

- Jess

Jess is also a java-based rule engine. It uses an enhanced version of the RETE Algorithm to process the rules. It can also directly manipulate and analyze Java objects. It provides a powerful Java scripting environment in which you can create Java objects, call Java methods, and implement Java interfaces without compiling any Java code. It supports SWRL (Semantic Web Rule Language) and the rules can be expressed in XML or Lisp languages. Jess is not an open-source, therefore it won’t let us alter its source code. But it is available with no cost for research purposes [16].

We chose the Drools engine as an inference engine and interfaced it with the XOnt Agent because it is a java based
open source software that allows embedding java classes and methods within its rule definitions. The rule syntax for the Drools engine is shown in “Fig. 3.” The method(s) for the action(s) triggered by the matched condition(s) should be placed within “then” clause.

```
rule "<name>"
  <attribute>*
  when
    <conditional element>*
  then
    <action>*
end
```

Figure 3. Drools Rule Syntax

Generally, an inference engine, also called a rule engine, tries to derive answers or new knowledge from a knowledge base which is composed of rules and facts as shown in “Fig. 4” [13]. The main function of an inference engine is to analyze patterns by matching facts and rules defined in a knowledge base and execute their corresponding actions. An action is likely to generate new knowledge or carry out specific method(s) of a specific object. The pattern matching process of the rule engines uses the RETE Algorithm, which infers conditions from facts and rules in a match-resolve-act fashion [6], [13].

Figure 4. Drools Inference Engine

III. THE EXTENDED XCREAM FRAMEWORK

The extended XCREAM framework includes the XOnt Agent as well as other existing agents such as the Collector Agent, the Proxy Agent, the Event Activation Agent, the Web Application Service (WAS) Agent, and the Event Handler [11]. The XOnt Agent has context awareness capabilities which enables a certain condition to be matched with incoming events based on known facts and rules and actions executed [3], [8].

A. The XOnt Agent

The context-aware system plays an important role in the current pervasive computing environment, as it recognizes and propagates environmental information in a certain situation, such as the current location of a person or object, available network connection, room temperature, and air pollution level in a specific region. In our framework, a certain context is recognized by a rule engine that is integrated into the newly extended XOnt Agent. The rule engine analyzes all the incoming events, reasons the relationship between them depending on the registered rules, and determines the best-fit context situation. The framework is designed to make the recognized context information trigger appropriate action(s) and perform service(s) according to the pre-registered scenarios. This approach is expected to increase the usability of identified or sensor data as the XOnt agent triggers several services of interest and enable service providers to develop complicated services with the combination of different events.

In order to describe the reasoning process of the framework containing the XOnt Agent in detail, the event flow of a rule matching process which is initiated by the Event Handler, is shown in “Fig. 5.”

1) XOnt Agent Implementation: The XOnt Agent Implementation component is in charge of managing the Fact Generator and the Knowledge Controller and communicating with the Event Handler. The component usually extracts unique identification information from “ReportsArrivedEvents,” which is delivered by the Event Handler. When “ReportsArrivedEvents” arrives, the component sends the event to the Fact Generator.

2) Fact Generator: The Fact Generator is responsible for converting events in its event queue into facts and forwarding them to the Drools Rule Engine.

3) Rule Engine: A rule engine is required in the XOnt Agent to help the XCREAM framework recognize a special situation by combining contexts, i.e. the various current statuses of an environment. The Drools Rule Engine is chosen as an inference engine in our framework. This rule engine

Figure 5. Event Flow of a Rule Matching Process
generates new knowledge through a pattern matching process and then sends the knowledge to the Knowledge Controller.

4) Knowledge Controller: The Knowledge Controller not only receives newly inferred knowledge from the rule engine, but also transforms the knowledge into “ReportsXontKnowledgeEvents.” Once the event has been generated, the controller forwards the event to the Event Handler.

IV. CONTEXT-AWARE INFEERENCE MODEL

The context-aware inference (CAI) model suggested in our research uses rule engine that matches the pre-registered rules in its rule base with the incoming context information which is referred to as facts. The CAI model allows time-varying combination of facts to be evaluated according to their possible availability or feasibility. We started the work by modeling an airport as an application environment. Then we analyzed different situations by deriving assumptions and constraints from expected situation(s). In the next phase, we built the CAI model by registering rules to the rule base on the basis of the assumptions and constraints.

A. Model Environment: Airport

An airport is chosen as an example environment to apply the CAI model. Firstly, we enumerated the major entities of the airport management system, including passengers, air cargoes, and smart airport facilities, along with their associated assumptions and constraints. Secondly, we included RFID/USN-enabled application services required in the airport management system and described the 3rd party systems interfaced to the framework. Lastly, we formalized the services in terms of rules which are expected to be fired when the facts from the entities meet the appropriate conditions.

1) Entities:

a) Passengers with smartphones: We assume that the passengers are all aware that they are being traced through their smartphones or RFID-enabled boarding passes. We also assume that their consent has been obtained. The smartphone acts as an information provider and a consumer in this framework, depending on the situation.

b) Air Cargo: Air cargo ranges from business air cargo to passengers’ luggages that are to be checked in. All of them are supposed to be traced with RFID tag.

c) Smart Airport Facilities: RFID readers are mounted on the primary hotspot(s) of a smart airport with motion sensors, noise detection sensors, thermo sensors, light sensor, air pollution detector, etc.

2) RFID/USN-Enabled Application Services: The services include Check-in, Security Check, Automated Immigration, Passenger Tracking, Air Cargo Tracking, Flight Information, and Airport Monitoring.

3) 3rd-Parties for System Integration: The airport framework is used to interface with Air Cargo Companies, Airline Companies, and Catering Companies.

B. Situation Analysis

Prior to building the CAI model, we listed the assumptions and constraints which are necessary to reason the expected context in the example environment and categorized them in the form of the assumption and constraint tables within the context database that contains Existence Availability table; Expected Transit Time table; and Sensor Range table.

1) Existence Availability: This is used for validating identification results. More specifically, it is used to verify whether a person or an object is to be recognized at different places within short time intervals or at the same time.

<table>
<thead>
<tr>
<th>Place1</th>
<th>Place2</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check-in Desk</td>
<td>Security Check</td>
<td>False</td>
</tr>
<tr>
<td>Security Check</td>
<td>Immigration Desk</td>
<td>True</td>
</tr>
<tr>
<td>Boarding Gate</td>
<td>Parking Lot</td>
<td>False</td>
</tr>
<tr>
<td>Restaurant</td>
<td>Store</td>
<td>True</td>
</tr>
</tbody>
</table>

2) Expected Transit Time: This is used to determine the min/max time of delay from one location to another, where RFID readers are installed.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check-in Desk</td>
<td>Security Check</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>Security Check</td>
<td>Immigration Desk</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Immigration Desk</td>
<td>Gate1</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>Gate1</td>
<td>Restroom</td>
<td>0</td>
<td>20</td>
</tr>
</tbody>
</table>

3) Sensor Range: This shows recommended range of each sensor device. An abnormal level of measurement value should be notified to the associated parties.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>-10</td>
<td>25</td>
</tr>
<tr>
<td>Illumination</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Noise</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>CO</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

C. Building the CAI Model

In order to resolve hundreds of complicated scenarios of the application services, we need to develop an effective way to control the execution of conflicting rules. When registered rules become fully matched in Drools rule engine, they are transferred to the Agenda, which controls the execution order of rules depending on its conflict resolution strategy, salience (or priority) and LIFO (last in, first out) [13]. The strategy is
applied to the XOnt Agent by assigning different priorities to the rules depending on the preference of a rule.

The demand for the standardized scenario execution scheme in RFID/USN-enabled applications encouraged us to suggest the CAI model. The CAI model basically requires situation analysis, the results of which are used as the basic criteria for essential inquiries which include read suitability model and event supervising model.

1) Read Suitability Model: This model validates whether the recognized person or object is supposed to be at a specific place, during a certain period of time. The model plays a basic role in reasoning complex scenarios which arise from a combination of variety of information from many sensing equipments.

```prolog
rule "Determine if the object is supposed to be read here"
  salience 20
  when
    IsPossible($objectid : objectid, $readerid : readerid)
  then
    Notify($objectid, $readerid);
end
```

Figure 6. Read Suitability

2) Event Supervising Model: This is also a general reasoning model with a variety of sensors. Usually, this model causes the alarm signal to be delivered to their corresponding application services, for example- Airport Monitoring Service.

```prolog
rule "Determine if emergency situation happens"
  salience 100
  when
    AbnormalSituation($gateid : gateid)
  then
    Notify($gateid);
end
```

Figure 7. Event Supervising

The two fundamental inquiries listed above trigger the methods defined as actions, when the conditions have been matched. The rule file can be maintained separately from the framework and altered when needed. Furthermore, we can develop more complex rules by combining the above basic inquiries.

V. CONCLUSION AND FUTURE RESEARCH DIRECTION

In this research, we have extended the XCREAM framework by adding the XOnt Agent. Doing so enhances the framework’s context awareness capability which enables hundreds of application services to immediately respond to complex situations that arise from various kinds of events of RFID/USN environment. In today’s world, events from individual sensor devices are increasing, and the various contexts derived from their relationships allow us to use them for decision-making and figuring out current situations. In addition, we have included the ontology system as the conceptualization method of events, objects and context. The research is focused on applying the framework to a real world environment such as an airport, and drawing distinct characteristics of the environment in terms of the RFID/USN-enabled application services. This resulted in setting up the context-aware inference (CAI) model, which can be widely applicable to the environments that are similar to an airport’s. As this model is based on the rule production system, the fundamental rules are used to derive new knowledge. This context-aware collaborative framework will be advanced in that it will allow us to carry out high level rules that are demanded by more complicated scenarios of the new services.

REFERENCES