

**TOWSON UNIVERSITY  
OFFICE OF GRADUATE STUDIES**

**COLLABORATION FRAMEWORK FOR A CONTEXT-AWARE ENVIRONMENT**

**WITH MULTI-DEVICE ENABLED APPLICATIONS**

**by**

**Kyung Eun Park**

**A Dissertation**

**Presented to the faculty of**

**Towson University**

**in partial fulfillment**

**of the requirements for the degree**

**Doctor of Science in Information Technology**

**Department of Computer and Information Sciences**

**Towson University  
Towson, Maryland 21252**

**December 2012**

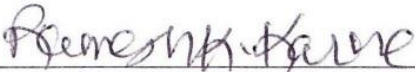
**TOWSON UNIVERSITY  
OFFICE OF GRADUATE STUDIES**

**DISSERTATION APPROVAL PAGE**

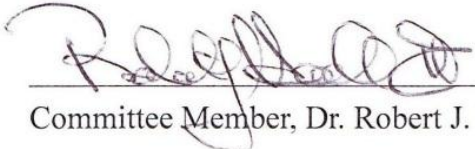
This is to certify that the dissertation prepared by Kyung Eun Park, entitled "Collaboration Framework for a Context-Aware Environment with Multi-Device Enabled Applications)," has been approved by the dissertation committee as satisfactory completing the dissertation requirements for the degree of Doctor of Science in Information Technology.

  
\_\_\_\_\_  
Chair, Dissertation Committee, Dr. Yanggon Kim

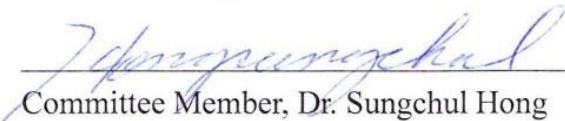
Nov. 2, 2012  
\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Committee Member, Dr. Ramesh K. Karne

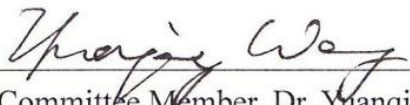
11/2/2012  
\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Committee Member, Dr. Robert J. Hammell II

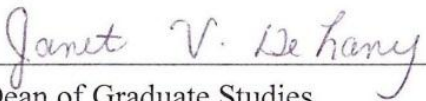
11/2/2012  
\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Committee Member, Dr. Sungchul Hong

11/2/2012  
\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Committee Member, Dr. Yuanqiong Wang

11/2/2012  
\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Dean of Graduate Studies

12/12/12  
\_\_\_\_\_  
Date

© 2012 By Kyung Eun Park  
All Rights Reserved

## ACKNOWLEDGEMENTS

There have been a number of people without whom this dissertation could not have been completed. I would like to deeply express my gratitude to Dr. Yanggon Kim who has broadened the horizon of my research terrain through the considerate support and constant propulsion. He helped me develop an earnest interest in the research process and encouraged me to complete this dissertation. I also would like to thank the committee members: Dr. Ramesh K. Karne, Dr. Robert J. Hammell II, Dr. Sungchul Hong, and Dr. Yuanqiong Wang for their invaluable advice, their perceptive comments, and their ongoing efforts to help me be a better researcher. I also would like to thank Dr. Chao Lu, Chair of Department of Computer and Information Sciences, Dr. Marius Zimand, and Dr. Yeong-Tae Song for their thoughtful support.

I would like to express my undying gratitude to my parents, my mother-in-law and my father-in-law who passed away, for their love, and support and for modeling what a person should be like. I also would like to thank my sisters, my brothers and the rest of my family for their love and encouragement. I would like to give my special thanks to my wonderful kids, Ian and Eugena for sharing the most precious time with me in the U.S.A. Most importantly, I wish to thank my beloved husband, Dr. Juno Chang, from the bottom of my heart for his endless faith, support, and love which allow me to complete this work.

*I dedicate this dissertation to my family,  
my husband, my son, and my daughter  
for their faithful support and everlasting love.  
I deeply love you all.*

## **ABSTRACT**

# **COLLABORATION FRAMEWORK FOR A CONTEXT-AWARE ENVIRONMENT WITH MULTI-DEVICE ENABLED APPLICATIONS**

**KYUNG EUN PARK**

With the proliferation of smart computing equipment, the range of intelligent application services is widely expanding with continuous attempts to make the best use of the contextual information from individual data sources. Accordingly, RFID (Radio Frequency Identification) or GPS (Global Positioning System) technologies are combined with the contextual information to enhance the accuracy and availability of many smart services. Another remarkable phenomenon is the explosively growing number of smartphone users, who stay connected to the Internet and interact with service providers at all times. This environment requires us to recognize a certain situation when receiving contextual information from various sensors and mobile communication devices. Additionally, the situation needs to make any related services activated and collaborated.

First, this research attempts to present an intelligently coordinating framework maintaining RFID-enabled applications. According to technology development, the framework has been evolved to the collaboration framework for a context-aware

environment with multi-device enabled applications. A variety of sensor data such as RFID tag events, sensing data from USN (Universal Sensor Network) sensor devices, and location information or instantaneous service requests from a wide range of mobile communication devices can be managed within the framework, the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware). It enables us to develop various applications including an emergency rescue system, a smart facility management system, a frequent mobility supporting system, a multi-agent collaboration system, and a personalized mobile security/safety system.

Upon receiving an event, the framework classifies the event based on the XOntology model, analyzes its contextual status with the existing facts (events) and knowledge (rules), and relates the contextual information to the appropriate actions (services). The event's associated service(s) is registered to the framework through the common interface of the XLogic script language, which increases the flexibility and interoperability across the extended framework environment, encompassing multiple application services.

In order to support the context-awareness scheme, the XOnt agent has been integrated into the existing XCREAM framework. It examines all the collected events to see if they correspond to any conditions of certain rule(s) that could trigger associated actions of the rule(s). The XOnt agent will be seamlessly integrated with the XOntology in the Phase II XOnt agent.

This research introduces the Context-Aware Inference (CAI) model based on general situation analysis. The situation analysis is used as the context-aware mobile security option which is applicable when developing mobile security-focused applications.

The performance and collaboration validity tests have been performed on the simulation environment of the XCREAM framework. The test results show that the framework works well in a collaborative service environment in which many heterogeneous application services and multiple event sources are complicatedly related with each other.

The major contribution of this research is the construction of the scenario-based collaboration framework for a context-aware environment. Most importantly, the framework communicates with multi-device enabled applications through the XLogic script language, which is developed as a multi-device access scheme and used to compose service scenarios. Further the simulation has been accomplished to prove the performance and the collaboration validity of the framework. In order to support context-awareness, the rule-based context awareness scheme has been introduced within the XOnt agent. In addition, the application of an ontology scheme is expected to enhance the availability of the contextual information and support future development of the finely customized services by combining the ontology scheme with the rule-based scheme of the framework.



## TABLE OF CONTENTS

LIST OF TABLES.....	xi
LIST OF FIGURES.....	xii
LIST OF ABBREVIATIONS .....	xiv
Chapter 1. Introduction .....	1
Chapter 2. Literature Review .....	9
2.1 RFID and its Standardization.....	9
2.2 Events Stream Handling .....	13
2.3 Continuous Query Processing.....	14
2.4 USN Middleware .....	15
2.5 Rule Engines .....	17
2.6 Context-Awareness .....	22
2.7 Inter-Application Context Sharing.....	24
2.8 Context Representation with Ontologies .....	25
2.8.1 Ontologies .....	26
2.8.2 Reasoning out of Formal Ontologies .....	28
2.9 Problem Statements .....	28
2.10 Solution: Proposed Ideas with the Collaboration Framework .....	29
Chapter 3. The XCREAM Framework .....	32
3.1 Research Strategy.....	32
3.2 The XCREAM Components .....	34
3.2.1 The Event Handler .....	34
3.2.2 The Collector Agent.....	35
3.2.3 The Proxy Agent .....	36
3.2.4 The Event Activation Agent.....	36
3.2.5 The Web Application Service (WAS) Agent.....	37
3.2.6 The Phase-I XOnt Agent.....	37
3.3 XOntology.....	41
3.3.1 Ontology Adoption.....	41

3.3.2 XOntology Components .....	41
3.3.3 XOntology Modeling .....	44
3.3.4 Phase-II XOnt Agent .....	46
3.4 XLogic Script Language .....	48
3.5 XLogic Script Execution Modes .....	50
3.5.1 Trigger Mode .....	51
3.5.2 Immediate Mode .....	51
3.6 Context-Aware Inference (CAI) Scheme .....	52
3.6.1 Smart Airport Environment .....	52
3.6.2 Situation Analysis Rules .....	53
3.6.3 Context-Aware Mobile Security Option .....	61
Chapter 4. Simulation .....	64
4.1 Simulation Environment .....	64
4.2 Performance Tests .....	65
4.2.1 Parsing Efficiency Tests .....	65
4.2.2 Event Processing Tests .....	67
4.3 Collaboration Validity Tests .....	69
4.3.1 <i>I</i> Event Generator vs. <i>N</i> Applications .....	69
4.3.2 <i>N</i> Event Generators vs. <i>I</i> Application .....	71
4.3.3 <i>M</i> Event Generators vs. <i>N</i> Applications .....	73
Chapter 5. Conclusion .....	77
APPENDICES .....	80
APPENDIX A: THE XCREAM APPLICATION, SPORTS COMPLEX MANAGEMENT SYSTEM (SCMS) .....	81
APPENDIX B: THE EXTENDED COLLABORATION MODEL WITH THE XCREAM FRAMEWORK .....	92
APPENDIX C: XLOGIC SCRIPT LANGUAGE AND ITS EXTENSION FOR XONTOLOGY ..	95
APPENDIX D: XLOGIC SCRIPT LANGUAGE SPECIFICATIONS .....	98
REFERENCES .....	107
CURRICULUM VITAE .....	121

## LIST OF TABLES

Table 1. Rule Engines: Drools vs. Jess .....	18
Table 2. Ontology Classes.....	43
Table 3. Ontology Properties .....	44
Table 4. Existence Availability Rules .....	54
Table 5. Access Availability Rule .....	56
Table 6. Expected Transit Time Rule .....	58
Table 7. Sensor Range Rule .....	60
Table 8. Average Parsing Time depending on the Number of Statements of the XLogic Scripts.....	66
Table 9. Average Event Processing Time depending on the Number of Events in Trigger Mode.....	67
Table 10. $I$ EG vs. $N$ Apps.....	70
Table 11. $N$ EGs vs. $I$ App .....	72
Table 12. $M$ EGs vs. $N$ Apps.....	74
Table 13. XLogic Script Tags.....	96

## LIST OF FIGURES

Figure 1. The XCREAM Framework Configuration .....	2
Figure 2. Service-Oriented Architecture (SOA) .....	3
Figure 3. Event Driven Architecture (EDA) .....	4
Figure 4. The XCREAM Framework .....	7
Figure 5. EPC Network.....	11
Figure 6. Drools Rule Syntax.....	20
Figure 7. Syntax for Rule Pattern .....	20
Figure 8. Drools Inference Scheme .....	21
Figure 9. The XCREAM Framework .....	35
Figure 10. The XEM Interface (Server Setting) .....	37
Figure 11. The XEM Interface (Service Scenario Management) .....	38
Figure 12. Rule Matching Process of the Phase-I XOnt Agent .....	40
Figure 13. The XOntology of Smart Airport Mobile Security Control .....	42
Figure 14. XOntology Class Hierarchy on Protégé .....	45
Figure 15. XOntology Property Hierarchy on Protégé .....	46
Figure 16. Phase-II XOnt Agent with Jena API.....	48
Figure 17. XLogic Script for “PassengerInfoService” .....	50
Figure 18. Trigger Mode Execution.....	51
Figure 19. Immediate Mode Execution .....	52
Figure 20. Existence Availability Rule .....	55
Figure 21. Matching Result of Existence Availability Rule.....	55
Figure 22. Access Availability Rule.....	57
Figure 23. Matching Result of Access Availability Rule .....	57
Figure 24. Expected Transit Time Rule .....	59
Figure 25. Matching Result of Expected Transit Time Rule .....	59
Figure 26. Sensor Range Rule .....	60
Figure 27. Matching Result of Sensor Range Rule.....	61
Figure 28. Read Suitability Model.....	62

Figure 29. Event Supervising Model .....	63
Figure 30. Simulation Environment.....	64
Figure 31. Parsing Efficiency Test Model.....	65
Figure 32. Average Parsing Time .....	66
Figure 33. Event Processing Test Model .....	67
Figure 34. Average Event Processing Time .....	68
Figure 35. 1 EG vs. N Apps Test Model .....	70
Figure 36. Average Time for 1 EG vs. N Apps .....	71
Figure 37. N EGs vs. 1 App Test Model .....	71
Figure 38. Average Time for N EGs vs. 1 App .....	72
Figure 39. M EGs vs. N Apps Test Model .....	73
Figure 40. Average Time for M EGs vs. N Apps .....	75
Figure 41. The SCMS Configuration.....	81
Figure 42. Event Flow of the SCMS.....	82
Figure 43. Network Configuration of the SCMS.....	83
Figure 44. DFD of the SCMS .....	83
Figure 45. Athlete Registration .....	84
Figure 46. Athlete Management.....	85
Figure 47. Room Management .....	85
Figure 48. Gym Management .....	86
Figure 49. Food Management .....	86
Figure 50. Collaboration Scenario of the SCMS .....	88
Figure 51. The Extended Collaboration Framework .....	92

## LIST OF ABBREVIATIONS

CAI	Context-Aware Inference
DBMS	Database Management System
EPC	Electronic Product Code
EPCIS	EPC Information Service
DL	Description Logic
DRL	Drools Rule Language
GPS	Global Position System
JEXL	Java Expression Language
KIF	Knowledge Interchange Format
NIST	National Institute of Standards and Technology
OWL	Web Ontology Language
RDF	Resource Description Framework
RFID	Radio Frequency Identification
SCMS	Sports Complex Management System
SPARQL	SPARQL Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USN	Universal Sensor Network
W3C	World Wide Web Consortium
XCREAM	XLogic Collaborative RFID/USN-Enabled Adaptive Middleware
XEM	XCREAM Enterprise Manager
XLogic	eXtensible Logic
XML	Extensible Markup Language
XSD	XML Schema Definition

## Chapter 1. Introduction

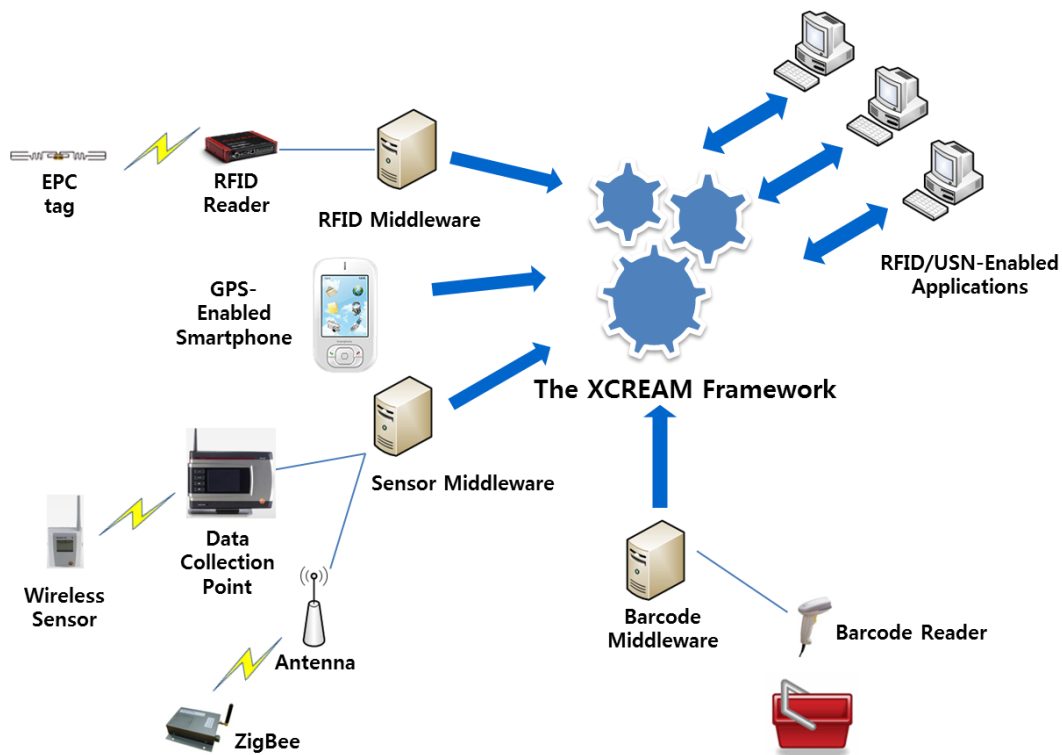
Rapid progress in wireless telecommunication and multi-device (RFID/USN) sensor technology encourages us to apply this advanced smart computing power to existing IT solutions. Developers and end-users are also motivated by the demand for elaborate services which use the latest technology.

Additionally, current IT systems are more likely to collaborate with each other and maximize the synergy effect from their inter-related systems. Accordingly, sharing of information across many involved parties should be resolved to support the collaboration effectively. As a solution to this requirement, an intermediate framework rather than individually paired connections is considered. Because numerous direct interfaces cause a heavy burden on developers and the target environment, a middleware framework will be responsible for processing minute details.

In order to provide end-users with the 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 this real-time data to the related application services. The framework must have a robust and reliable infrastructure, able to handle a huge amount of tag and sensor data and propagate them to the appropriate services according to predefined scenarios. The new framework will be responsible for triggering services by monitoring incoming events, which represent the current state of environment, and performing pre-registered action plans according to the identified events.

These requirements first 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 as described in the previous researches [49]–[51].

The XCREAM is placed between multi-device middlewares and application services as in “Figure 1.” It collects the ECREports format sensing data from multi-device middlewares [67], [68], interprets them, and delivers them to the correspondent business application services, establishing an integrated collaborative working environment.



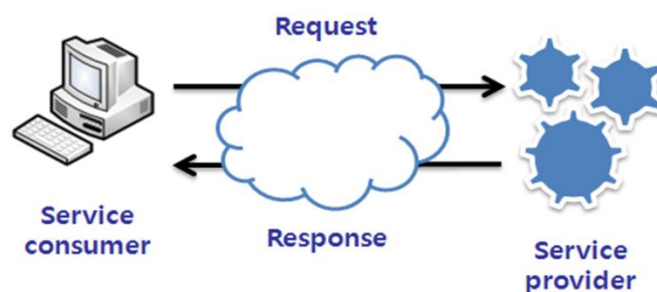
**Figure 1. The XCREAM Framework Configuration**



The XCREAM framework allows us to easily develop multi-device enabled applications because it acts like a bridge between physical sensors and application services. By using the XCREAM framework as a bridge, most sensor specific details are migrated to the XCREAM framework from individual application services, and individual connections between autonomous solutions of institutions or agencies are replaced by the framework-to-service connections.

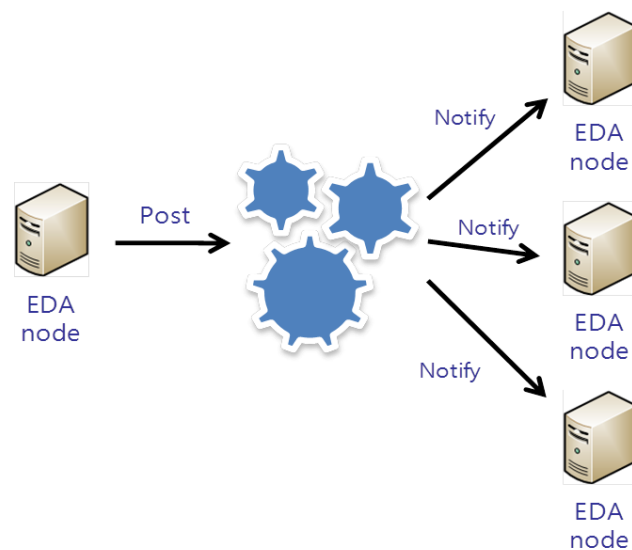
The framework not only conforms to the EPCglobal's ALE interface [68] but also provides its own XML infrastructure language, called "XLogic (eXtensible Logic)." XLogic script is registered to the framework and is triggered whenever a specific event is identified or a certain measurement value reaches its threshold. In addition, a script caching mechanism is integrated into the framework to immediately process the XLogic script, saving time spent on parsing the script.

The XCREAM framework adopts the service-oriented architecture (SOA) as an interface scheme for multi-device enabled application services. It adopts a conventional request/reply mechanism as described in "Figure 2".



**Figure 2. Service-Oriented Architecture (SOA)**

This not only guarantees the independence of individual services, but it also gains flexibility when extending the framework infrastructure, enabling new collaborative services to work with the XCREAM or other application services. The SOA makes it possible to increase the reusability of the business services. Easily applicable service functionalities through the SOA also provide the foundation for stable provision of composite services. The additional advantage of using SOA consists in retaining a loose coupling of the service building blocks across the whole service framework. Further, SOA technology allows a business application to request any services which are exposed to others with SOA interfaces, regardless of the service platform environment. Moreover, the SOA has been focused as a flexible integration technology in parallel with the demands for the fast development of multi-device enabled applications as well as the sharp increase in the numerous device-oriented services [17], [18], [25].



**Figure 3. Event Driven Architecture (EDA)**

The interface scheme of the XCREAM follows an event-driven architecture (EDA) among the constituent agents of the XCREAM and the external application services. The EDA introduces long-running asynchronous process and an EDA node transmits events and does not care about the availability of others based on post/notify mechanism (see “Figure 3”). The internal agents composing the XCREAM framework are designed according to this architecture, immediately propagating processed events to others and selectively processing received events [56], [57]. The applications work independently with individual multi-device middlewares and execute service scenarios only when the triggering events from the multi-device middleware are recognized.

The adoption of the SOA and the EDA to the XCREAM framework plays a cooperative role in interfacing event generators with their consumers such as application services. These services cope actively with the changes and update their scope more flexibly by easily establishing or withdrawing interfaces with numerous event generators [24].

Generally, multi-device enabled services such as an emergency rescue system, a smart facility management system, or a supply chain management system not only provide their own services but also perform more advanced collaborative services by interfacing within the XCREAM framework.

Furthermore, smartphones allow the users to stay connected to the Internet at all times, and the pervasiveness of the sensor devices gives them the opportunity to use the information from the devices. These factors increase the

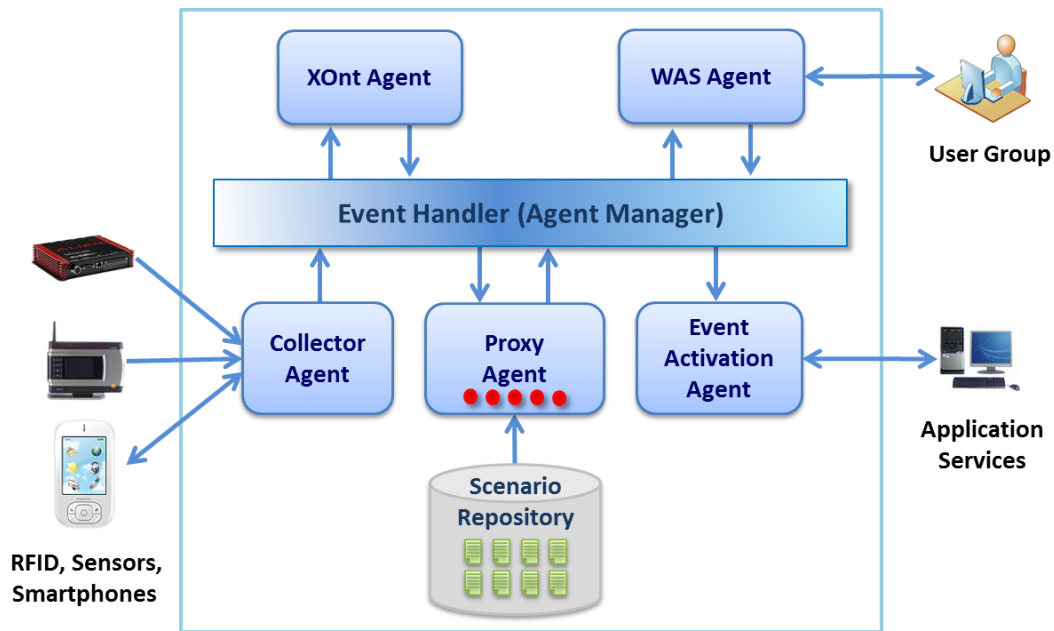
need to process a huge number of data transactions and infer the present situation from the current context. The factors that are included in the current context are the owner's current location, status values of various sensors, media preferences, community accessibility, and so on. This new environment also encourages the development of personalized context-aware services.

The need to analyze the factors present in the current context encouraged us to try to integrate the new XOnt agent into the XCREAM collaboration framework through the ontology based context-aware scheme [27]. Analyzing incoming events, the XOnt agent promotes applications to share the contextual information in a highly networked pervasive computing environment. In particular, we focused on the facts that smartphones work either as information providers or as information consumers, and the location information of smartphone users is to be combined with sensor information. This situation results in the extension of the range of contextual information and the need to intelligent analysis of current state [63]. All the collected sensor information should be examined if it meets any condition that triggers a correspondent command or rule through the reasoning process of a rule engine with the XCREAM framework [11].

As illustrated in "Figure 4," the extension of the XCREAM has been accomplished by adding a new agent called the XOnt agent, which analyzes a particular situation among the numerous event data. The XOnt agent recognizes a certain condition by comparing various sensor inputs with its pre-

constructed knowledge base. It then activates related services (actions) depending on the rule matching mechanism of the XOnt agent.

With the introduction of the XOnt agent, the applications work more efficiently and comprehensively than before because each service is defined by a finer rule-based scenario. The XOnt agent observes all incoming events, finds certain context, and triggers the appropriate service(s). For example: when a handicapped passenger with a 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. In addition, the XOnt agent combined with the XOntology scheme is discussed as the Phase II XOnt agent scheme.



**Figure 4. The XCREAM Framework**

In addition, the XOntology has been introduced as the ontology specification of the XCREAM framework. It defines the entities of real-world objects and their properties in order for the application services to share the concepts and relationship information. The adoption of ontology into the framework results in enhancing the interoperability and system reusability. The XOntology is composed of core and extended ontology sets. The core ontology includes person, place, and time information. The extended ontology covers high-level applied concepts such as inventory management, logistics, surveillance function, environmental monitoring, and so on.

This research also includes the Context-Aware Inference (CAI) model specialized to the context-aware mobile security option [47]. This option is made on the basis of the general situation analysis in combination with the XOntology specification classifying the real-world entities.

The performance and collaboration validity tests have been performed on the simulation environment of the XCREAM framework. The test result shows that the framework works well in a collaborative service environment in which many heterogeneous application services and multiple event sources are complicatedly related with each other [48].

## Chapter 2. Literature Review

This overviews various related researches, starting with an auto-identification with RFID technology and ending with rule-based reasoning methodology for collaboration framework for a context-aware environment. RFID-enabled auto-identification technology is related with the data collection and automatic recognition of the numerous events at the front-end of the framework. To effectively manage the massively incoming events, continuous event and query processing technology is considered as a filtering scheme. And also, USN middleware technology is examined when devising the framework [17], [18], [25]. The second half of this chapter includes rule engines and context-awareness capability for systematic reasoning, and ontologies for inter-application context sharing.

Each research topic includes problem statements, research strategies, and adopted solutions of the XCREAM framework.

### 2.1 RFID and its Standardization

A lot of research efforts in the past decade are aimed at defining RFID tag specification and handling the RFID tag data. Past research also focused on formalizing seamless communication protocols and enhancing system performance and correctness.

RFID-enabled network infrastructure was initially proposed by the Auto-ID Labs [54] at MIT. The labs were dedicated to creating the Internet of Things using RFID and Wireless Sensor Networks. The labs, evolved into

EPCglobal, developed ubiquitous automated identification technologies. It is based on the networked physical world by proposing the EPC (Electronic Product Code) Network. EPCglobal also suggested an open architecture system composed of the EPC, EPC tags and readers, local networking technology, and RFID middleware [67], [68]. The EPC network identifies goods using a unique numbering system called an Electronic Product Code (EPC), enabling all objects in the world to be linked via the Internet [18], [67].

An EPC number is a universal identifier for any physical object and works. EPC can be termed a “Pure Identity EPC URI” because it identifies computer systems including electronic documents, database, and electronic messages in the form of an Internet Uniform Resource Identifier (URI).

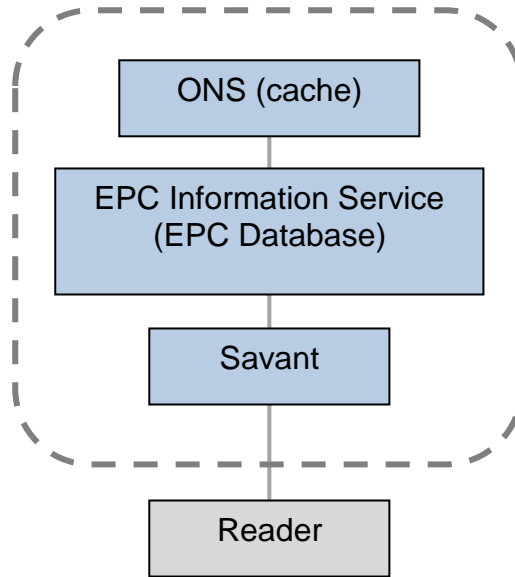
The EPC Tag Data Standard [66] defines the Electronic Product Code and specifies the memory contents of Gen 2 RFID Tags.

```
urn:epc:id:sgtin:0614141.112345.400
```

In addition, memory limitations of RFID tag causes the EPC not to be stored in URI form but in an encoded form with “EPC Binary Encoding” on the tag.

The EPC Network consists of three major components: the Savant, the EPC Information Service (EPCIS), and the Object Name Service (ONS). The reader may or may not be regarded as part of the EPC Network. The following “Figure 5” shows the structure of an overall EPC Network [32].





**Figure 5. EPC Network**

The above EPC Network is known as a local EPC Network, which is valid within a company or a private network. Each local EPC Network can be linked together through the Internet, so the information can be globally reachable and sharable. In this case, a global public ONS system may be used to connect EPC Networks in a similar way how the Domain Name System (DNS) supports on the Internet [32], [67].

Generally, the Savant contains two interfaces, one for readers and the other for applications. In this research, the XCREAM plays the Savant's role by receiving the EPC numbers of tags and forwarding them to the appropriate multi-device applications. In addition to the tag information, The XCREAM also handles the sensor data from the USN devices. The tag information is directly stored in the EPC database or sent to the EPCIS which processes the tag data. By querying the EPC database, the ONS in an EPC Network

recognizes the location information of data needed by an application. If the tag number is not resolved into a Uniform Resource Locator (URL) within the local EPC Network, the ONS searches for the external global public ONS across the Internet to get the location of the information associated with the EPC number. The EPC Information Service (EPCIS) is a gateway between any enterprise applications which request information and the EPC internal database which returns corresponding results [32], [67].

The XCREAM framework follows the interface mechanism of the Application Level Events (ALE) Specification of EPCglobal [68]. This mechanism not only enhances independence between the front-end data collector and the applications, but also increases flexibility when adding additional readers or sensors and applications to the framework. Especially, the request/reply method of the XCREAM corresponds to the main API of the ALE, including define, undefine, getECSpec, getECSpecNames, subscribe, unsubscribe, poll, immediate, getSubscribers, etc. Moreover, the XCREAM recognizes ECSpec and ECREports [68].

The latest ALE Specification (ALE Ver. 1.1.1) defines the *reading* and *writing* activities between the physical data sources and client applications [68]. In the reading activity, the ALE Specification receives EPCs and related data from data sources, accumulates and filters the data, and reports them to the recipients in various forms. On the other hand, the writing activity involves isolating the individual data channels, reading and writing the operations of the data, and generating the reports in various forms. The writing activity allows

applications to exercise specific operations such as “*lock*” and “*kill*” to enhance the level of security when using the RFID technology. This feature is added to make up for the lack of safety in the first version.

## **2.2 Events Stream Handling**

Regarding the handling of events, a lot of research efforts have been made in the last decade to formalize the event, the behavior of the events, and the relationship between them.

The study on event streams was derived from the concern for handling time-varying data within a time-series management system. Then, the active database rose to the surface of event-based temporal reasoning [41]. Active database has been frequently compared with the data stream management system because active database uses triggers to support automatic responses to events. The action may change the value of the database, whereas, the rapid materialization of ubiquitous computing environments revitalize the events stream handling issue. Data stream management system demands complex events processing schemes to treat the inflow of events from their sources. In addition, the process supports the continuous queries over the running streams [36], [52].

Many researchers have analyzed events and distinguished the nature of the event queries over event streams from the traditional queries on the relations [4], [36], [41], [64].

The most important difference between event query and ordinary query used in the traditional DBMS is that the former is initiated by an event, while the latter is self-activated [12]. In particular, time-varying event query needs to keep the event history in event repository. As a solution to this, the sliding window mechanism has been chosen to constrain the number of event at a particular point of time [21].

### **2.3 Continuous Query Processing**

Under highly networked environments, a variety of automatic identification devices generate huge amounts of event streams. This means that a new querying scheme is required to recognize arriving events and to evaluate corresponding queries. In order to do so, users need to issue a continuous query (CQ) which is invoked once and runs continuously over the event streams and database. In addition, the system is notified whenever the data matches the query [60]. The continuous query processing system must be capable of monitoring incoming events during a certain predefined time interval and dynamically performing query evaluations [12], [13]. Whereas, query transactions of traditional database management system (DBMS) are generally processed over the present relations [4], [29], [36], [41], [64]. The XCREAM interprets the XLogic script, which contains immediate (single) or trigger (continuous) mode event query request, into a Java runnable object, transforming the written script into an executable form. The runnable objects are activated by a specific event as in the event driven systems.

Many data stream management systems (DSMS), such as TelegraphCQ [12]; NiagaraCQ [13]; OpenCQ [34]; and STREAM [83] have been developed as solutions to multiple continuous, high-volume, and possible time-varying data streams. The TelegraphCQ adopts an adaptive query engine to process queries efficiently in volatile and unpredictable environments. The NiagaraCQ keeps scalability by grouping CQs. The OpenCQ uses a query processing algorithm based on the incremental view maintenance [7]. The STREAM maintains an adaptive and dynamic central scheduler of query operators over a shared stream queue which holds high-volumes of data.

These systems are primarily focused on CQ optimization, by allowing traditional DBMS or their own data stream management system to support CQs over data streams and conventional relations. The XCREAM supports CQs over time-varying RFID tags or sensor data found in trigger mode. It also attempts to enhance the performance by caching the runnable objects of the XLogic scripts correspondent to them.

## **2.4 USN Middleware**

The requirements of early USN middleware were comparatively simple due to its direct connection to a specific application service. As various computing applications, such as e-Healthcare, e-Transportation, e-911, and e-Eco, are rapidly introduced to our daily life, it is expected that the increase of information sharing from variable multi-device data sources will stimulate the development of composite services [28]–[30]. Several middleware researches

are conducted on the following examples: MiLAN middleware [25], developed to guarantee QoS (Quality of Service) as its top priority; event-based DSWare middleware [33], which uses a continuous stream of sensor data to send information on to USN application system; Impala middleware [35], which can dynamically change functions of sensor node middleware according to changes in USN application services and changes in the surrounding environment of sensor networks through wireless communication; TinyDB middleware [37] and Cougar middleware [62], which carry out the requirements of USN application systems, using distributed processing by considering sensor data from the sensor network.

In COSMOS (Common System for Middleware of Sensor Network), a system developed by ETRI, key functions of middleware, jointly needed for various types of USN application services, were extracted. In addition, technology development and standardization to provide these in a standardized way were carried out. The main functions of COSMOS are as follows: query support (temporary, permanent, and event), support for simultaneous processing of lots of queries in large-capacity sensor network environments, and support for abstraction of heterogeneous sensor networks [46].

Oracle Complex Event Processing (Oracle CEP) combines distributed caching to publish output events to a replicated, distributed cache, thereby making the results of its computation highly available. Distributed caching technology can also enhance the performance and scalability of Oracle CEP applications [76].

The XCREAM also supports the caching mechanism of the runnable objects of the XLogic Scripts. Further, incoming events are transmitted to the Proxy agent, triggering the runnable object of the associated scenario to be executed.

## **2.5 Rule Engines**

Out of the many kinds of rule engines available, the Drools [72] and the Jess [74] are picked to be candidates of the XCREAM framework for its popularity and continuous technical support. The range of the XCREAM has been become wider by allowing additional sensors or mobile devices like smartphones to interact with the associated applications. As a result, the need to systematize the process of controlling the information and performing the rules of the action plan increases.

The following “Table 1” summarizes the key features of both rule engines. It gives a comparative chart for the selected inference engines measured on different performance metrics, including their main algorithms, OWL-DL entailment, Java and rule support, version, and licensing issues. Most importantly, both rule engines are based on the RETE algorithm, which matches data tuples (“facts”) against productions (“rules”) in typical pattern-matching production system interpreters [19], [40].

**Table 1. Rule Engines: Drools vs. Jess**

<b>Rule Engine</b>	<b>Drools</b>	<b>Jess</b>
<b>Algorithm</b>	RETE Algorithm	RETE Algorithm
<b>OWL-DL Entailment</b>	No	Yes
<b>Java Support</b>	Yes	Yes
<b>Rule Support</b>	DRL (Drools Resource Language, Own Rule Format)	SWRL
<b>Version</b>	5.4 & 5.5 Beta	7.1
<b>Licensing</b>	Free / Open source	Academic use only

- Drools

The Drools rule engine is a business logic integration platform with Java-based object-oriented rule engine, providing a unified and integrated platform for rules, workflow, and event processing. The Drools 5 adopts an optimized version of the RETE algorithm [19], [73], called RETE-Object-Oriented algorithm, in order to give high performance. It has its own writing rules called DRL (Drools Rule 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, including rule debugging and rule authoring tools like the IDE plug-in [72]. As a rule engine, the Drools Expert allows us to do declarative programming, explaining the source of the solution and the decision-making process [73]. Particularly, dynamic configuration of various sensor devices in present environment is effectively supported by separating rules from internal controlling structure of the framework.



- Jess

The Jess is also a Java-based rule engine and scripting environment. It uses an enhanced version of the RETE algorithm to process rules. It can also directly manipulate and analyze Java objects, as it is entirely in Java language. It provides a powerful Java scripting environment in which you can create Java objects, invoke Java methods, and implement Java interfaces without compiling any Java code. It supports SWRL (Semantic Web Rule Language) [2] and the rules can be expressed in XML or Lisp languages. The Jess is not an open-source project, therefore, it will not allow us to alter its source code. But it is available with no cost for research purposes [74].

We chose the Drools engine as the rule engine of the XCREAM framework and integrated it with the XOnt agent for its open license policy and its easy-to-use interface mechanism, especially simple DRL scheme. As it is Java-based open source software, Java classes and methods within its rule definitions are readily embedded in the existing framework. The rule syntax for the Drools engine and a simple DRL rule for a fire alarm system are shown in “Figure 6.” The method(s) for the action(s) triggered by the matched condition(s) should be placed within “*then*” clause. Among conditional elements, a pattern element is the most important conditional element and matches on each fact that is inserted in the working memory of a session [73].

A pattern contains zero or more constraints and has an optional pattern binding. The following “Figure 7” diagram shows the syntax for a rule pattern of the Drools rule engine.

```

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

rule "When the fire is gone turn off the sprinkler"
when
  $room : Room( )
  $sprinkler : Sprinkler( room == $room, on == true)
  Not Fire( room == $room)
then
  modify( $sprinkler) { setOn(false) };
  System.out.println( "Turn off the sprinkler for room " +
    $room.getName() );
end

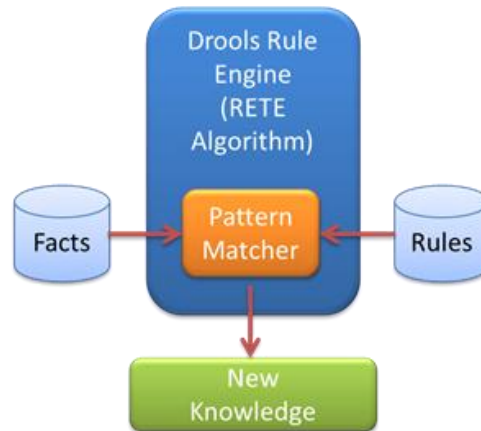
```

**Figure 6. Drools Rule Syntax**



**Figure 7. Syntax for Rule Pattern**

A rule file can contain multiple rules, queries, and functions, as well as some resource declarations like imports, globals and attributes that are assigned and used by the rules and queries [73].



**Figure 8. Drools Inference Scheme**

Generally, a rule engine tries to derive answers or new knowledge from previously known rules and facts as shown in “Figure 8” [72]. The main function of a rule engine is to analyze patterns by matching facts and rules and executing their corresponding actions. An action is likely to generate new knowledge or carry out specific method(s) of a specific object.

This pattern matching process of the production system uses the RETE algorithm, which infers conditions from facts and rules in a match-resolve-act fashion [19], [72], [73]. The RETE algorithm is an efficient pattern matching algorithm designed by Dr. Charles L. Forgy [19]. It applies the matching algorithm only to the changed facts and enhances the matching performance. He created a production system program consisting of an unordered collection of if-then statements called productions. In this system, the data operated on by the productions is held in a global data base called working memory. An

individual production is composed of LHS (Left Hand Side), a sequence of patterns, and RHS (Right Hand Side), an unconditional sequence of actions. The interpreter executes a production system by performing the following typical cycle: (1) Match; (2) Conflict resolution; and (3) Act [19].

## **2.6 Context-Awareness**

In the earlier stage of context-aware technology, many people wanted to define their own meaning of “context,” and many types of variation of “context” were found. The term, “context-aware,” was introduced in Schilit and Theimer (1994) [55] for the first time. The authors describe context as location, identities of nearby people and objects, and changes to those objects. Three years after, Ryan (1997) [53] referred to context as the user’s location, environment, identity, and time. After that, one of the most accurate definitions is given by Dey and Abowd (2000) [14], who regard 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” [8].

As various kinds of event data and services interact with each other in an environment, a single event can relate to numerous services, and several different events may have to do with the same specific context. This means that the framework can link interconnected environments between physical devices and cyber application services. Also, we realized that original events

and the identified context data which exist within the framework should be mutually understandable by related parties. As a result, an ontology scheme is additionally applied as mutual communication method in the XCREAM framework in combination with a context-awareness scheme. This method not only allows applications to share contextual information based on the common ontology specifications but also provides a powerful way to develop adaptively collaborative services.

The XOnt agent has been added to the XCREAM framework in order to sort out valid contextual information among numerous incoming events. Once the XOnt agent receives various events from multiple sources, a tightly coupled rule engine, called Drools [72], makes a decision regarding whether a certain context has been met. This is accomplished by evaluating the pre-registered rules with the newly received events and the current as-is condition [11].

The XOnt agent includes the XOntology [27] 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. 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 an 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. We will discuss context representation with ontology in the following section in more detail.

## **2.7 Inter-Application Context Sharing**

Since an Internet connection is available at any time and any place, we have faced an increasing demand to take full advantage of this highly intelligent environment where multiple data sources are available. Particularly, the demand leads us to develop a ubiquitous infrastructure framework in which all the context data collected from various sources is managed and shared across the correlated applications of all interested parties.

First of all, the underlying framework needs to understand various kinds of context information according to a common context description script. This is done in order to enhance the effectiveness in representing and sharing the information. Most importantly, the common context description scheme needs to be clear and general enough to be accepted by all related parties.

As a solution to representing contextual information, RDF (Resource Description Framework) [43], [85]–[87] and OWL Web Ontology Language [3], [23], [43], [84] of the Semantic Web standards are applied to the common context description scheme of the framework. 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 ultimate vision of the Semantic Web is to make automated semantic agents access the Web on their own and carry out tasks

only by referring to the semantics encoded into the Web page on behalf of users [31].

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

## **2.8 Context Representation with Ontologies**

The current ubiquitous computing environment contains lots of smart objects working with their corresponding application services. This phenomenon requires us to recognize the current situation out of the complicated combination of contextual information. In addition, the application services need commonly understandable scheme to share the contextual information in this environment. First of all, the sharing of current context begins with the clear definition of common vocabularies. Once we build the dictionary of the common vocabularies, we further proceed to detect a situation determined by the individual or combined contextual information. This process is treated by adopting an ontology scheme as a tool for denoting the contextual information and describing the relationships among the individual entities.

### 2.8.1 Ontologies

Ontologies have been described as “a specification of a conceptualization,” by Gruber in 1993 [23]. He mentions that: “what ‘exists’ is exactly that which can be represented”. In other words, all the intelligent equipment and objects in the current computing environment should be represented in a certain ontology language and shared by various parties. In the real world, especially Semantic Web world, ontology is about “the exact description of things and their relationships” [88].

A general ontology model describes concepts of target domain, properties and attributes of these concepts, constraints on properties and attributes. Additionally, it optionally examines individuals, defining a common vocabulary and encouraging a shared understanding [44].

On the other hand, Neches et al emphasize the importance of ontology as a method to enhance knowledge sharing and reusability across different applications [10], [42]. Once ontologies for a specific domain have been built, it can be shared and reused for other domains by adding new concepts into the existing one.

As a domain modeling tool, ontology languages such as RDF (Resource Description Framework), RDFS (RDF Schema) and OWL Web Ontology Language of the Semantic Web standards describe the common vocabularies and their relationships within the framework.

RDF is a primary representation language for domain models or ontologies. It allows us to define things, namely resources as well as their



properties, the relationships between resources [14], [15], [80]. RDF is composed of triples (statement), subject-predicate-object, and the formation of a graph of the resources [45]. RDF serves as a general-purpose language for representing information in the Web; whereas, RDFS includes the vocabulary of RDF [87]. RDFS is introduced to strengthen the semantic capability of RDF. In other words, it provides a syntactic specification mechanism to allow us to describe resources as classes and their relationships as properties.

In OWL, a class denotes a group of individuals characterized by a certain common attribute, and it contains the set of objects. While modeling a domain of interest, inter-class relationships are described as a set of subsumption relations, a collection of superclass-subclass relationships [59]. These relationships are represented as a class hierarchy. In a similar way, a hierarchy of properties, attributes, or relationships are formed to a property hierarchy in OWL scheme [3], [84].

A semantic model of a domain of interest and its associated rules need to be built by a domain expert who may not be familiar with expert system based on artificial intelligence or formal logic theory. Semantic/context representation languages such as RDF or OWL are too complicated to use at ease, especially while modeling a domain. As a result, the Semantic Application Design Language (SADL) is defined as “a language for building semantic models and expressing rules that capture additional domain knowledge” [69]. In addition, a model defined in SADL for an application domain can be mapped into OWL ontologies, too.

In chapter three, the ontologies of a smart airport mobile security option are developed with the semantic representation of the domain by using the Protégé [78] ontology development tool.

### **2.8.2 Reasoning out of Formal Ontologies**

Knowledge is represented in the form of ontologies since machines interpret and express concepts, relations, and specifications. The knowledge base is extended by deriving new facts from existing ontologies. Knowledge can be formalized by using Knowledge Interchange Format (KIF) [20], [22], Conceptual Graphs [58], [65], or Description logics (DL) [6], [61], [85]. Among them, Description Logic is picked up to formally represent concepts because it is expressive enough to describe logics of realistic applications. Usually, a reasoner checks consistency of the T-Box of terminology which describes terminology and rules and the A-Box of assertions, which looks at assertions about individual concepts. It also checks subsumption relationship across the existing relations [5], [6], [26], [61]. This scheme will be applied to the Phase-II XOnt agent.

## **2.9 Problem Statements**

The research has focused on establishing a collaboration framework for a context-aware environment with multi-device enabled applications. In addition, a simple and expressive interface mechanism between the framework and the applications is expected to provide a flexible and dynamic system

configuration within the framework. As the applications increasingly correlate with each other in the ubiquitous computing environment, a single event can make a great impact on almost every application service. Once a single event appears, it should be forwarded to the associated service(s), according to the automatic identification of the related services within the framework.

The identification procedure makes the target framework immediately recognize the context changes that result from the occurrence of an event. Within the target framework, the associated application services should be identified according to a systematic analyzing method such as an ontology modeling system that is based on contextual information classification and a rule-based inference scheme.

## **2.10 Solution: Proposed Ideas with the Collaboration Framework**

As a solution to the identified problem in this research, the need for a collaboration framework for a context-aware environment with multi-device enabled application, the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware) has been proposed. The framework mediates between smart devices (physical objects) and collaborative application services (cyber services) by collecting massive events from a variety of event origins and distributing them to the appropriate service parties depending on the predefined application business scenarios.

The application services are connected to the framework through the Web interfaces of the XCREAM Enterprise Manager (XEM). Actual

interaction is made by using the XLogic script language as a unified interface scheme between the framework and individual application services. In doing so, the script provides flexibility and interoperability for newly extended services in the future.

The XCREAM framework has been tested to examine its performance by checking the processing time for interpreting the XLogic statements and handling RFID tag events from the data origin to the individual application agents. Additionally, a validation of the framework in terms of collaboration feasibility has been accomplished.

The Context-Aware Inference (CAI) scheme has been suggested to present a widely applicable collaboration model for a fast-growing pervasive computing environment. The CAI scheme is applied to the design of the context-aware mobile security option.

Accordingly, the XCREAM is extended to help the framework recognize a special situation, specifically an airport domain that combines contextual information. The most typical contextual information includes the current status of an airport environment or the passengers' current location. The contextual information is combined with the XOntology specification, classifying the real-world entities of an airport. Within this environment, general situation analysis rules are applied to the acquired facts in order to recognize a violation of the rules, indicating an occurrence of any abnormal situation. The CAI scheme is advanced to suggest the common mobile security option for the customized multi-device enabled applications.

The extension of the XCREAM has been accomplished by adding a new agent called the XOnt agent, which reasons a particular situation among the numerous event data. The XOnt agent recognizes a certain condition by comparing various sensor inputs with a pre-constructed knowledge base and activates related services (actions) depending on the rule matching mechanism of the XOnt agent. In addition, the XOnt agent deploys the XOntology scheme, allowing the application services to share the concepts and relationship information while enhancing interoperability and system reusability.

## Chapter 3. The XCREAM Framework

This chapter begins by stating the individual agents' role from the standpoint of an event driven system. Individual agents collect events from physical data sources and transport them to their associated application services. The following sections explain the execution process of service scenarios written in the XLogic script language, introducing a unified interface between the framework and individual application services. In addition, the context-aware inference (CAI) scheme has been suggested to present a widely applicable context-aware mobile security option in a fast-growing pervasive computing environment.

### 3.1 Research Strategy

The XCREAM was first implemented as a scenario-based adaptive service platform, and it managed more adaptive service scenarios among highly correlated business applications [51]. By registering scripts written in a XML infrastructure language called the XLogic, individual services are seamlessly integrated into the XCREAM. The XLogic scripts describe specific service scenarios related to events such as automatic identification or remote measurement data which come from separate multi-device middlewares.

The XCREAM has evolved into a scenario-based collaborative framework, playing as a mediator between smart physical objects and collaborative cyber services, gathering massive events from a variety of event origins, and distributing them to the appropriate service parties depending on

predefined business scenarios. The sports complex management system (SCMS) [50] (see Appendix A), a prototype system based on the XCREAM, was presented in order to show how the framework flexibly helps applications collaborate with each other by allowing them share events through the unified XLogic script language [48], [49].

A rapidly growing demand for collaboration among numerous and heterogeneous business applications encourages the applicability written in a XML infrastructure language called the XLogic of the XCREAM and gives the framework the opportunity to extend its range. This research was followed by validating the framework in terms of collaboration as well as testing its performance [48]. This framework is also applied to build a mobile real-time tracking system by integrating smartphone applications with the framework [63].

Recently, the core XCREAM framework has been extended with the XOnt agent in order to support context-aware collaboration in the complex sensing network environment [11]. Because events from individual sensor devices are increasing, the demands to handle various contextual events also increased, encouraging us to use them as the basis of decision-making for figuring out a current situation. In addition, we have included an ontology scheme as the conceptualization method of events, objects and contextual information [27]. Furthermore, our research focused on applying the framework to a real world example, an airport, and using the multi-device application services to understand the distinct characteristics of the

environment. This resulted in setting up the context-aware inference (CAI) scheme, which can be extensively applicable to similar environments. As this model is based on the rule production system, the fundamental rules are to be used as basis knowledge in order to derive new knowledge. This context-aware collaborative framework will be advanced, so it can carry out the high-level rules which are demanded for more complicated scenarios in the new services [11], [47].

## **3.2 The XCREAM Components**

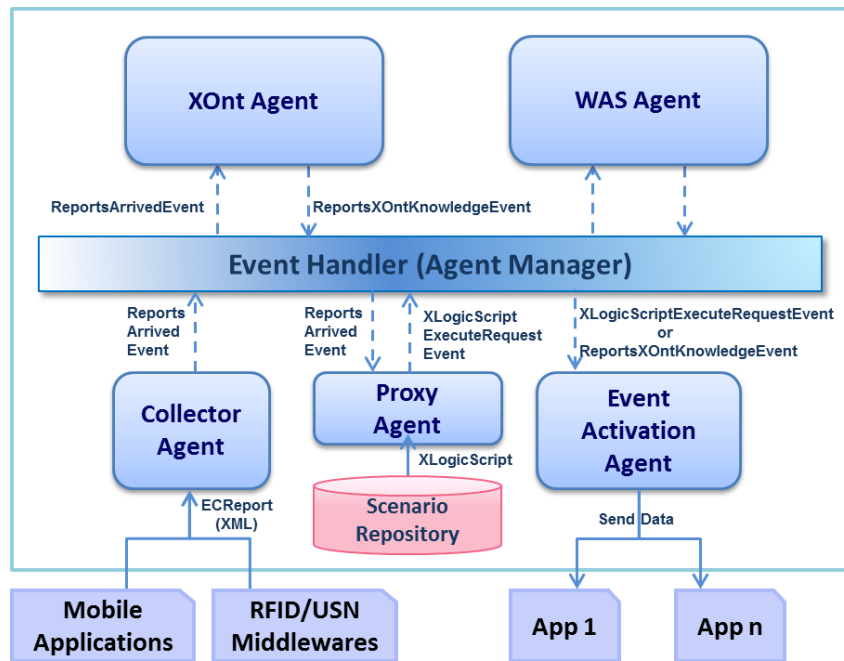
The XCREAM framework is composed of five agents including the Collector agent, the Proxy agent, the Event Activation agent, the XOnt agent, the Web Application Service (WAS) agent, and the Event Handler as shown in “Figure 9.”

### **3.2.1 The Event Handler**

The Event Handler maintains the runnable objects of the XLogic scripts, which invoke the corresponding web services of the application services.

The Event Handler is in charge of managing the remaining agents and delivering events received from each agent to the appropriate parties. Internal events just pass through the Event Handler. The Event Handler propagates events to all the agents, and the agents accept only events of interest for further processing and ignore the remaining events. The following “Figure 9” shows the overall flow of events across the XCREAM framework.





**Figure 9. The XCREAM Framework**

### 3.2.2 The Collector Agent

Events originated from multi-device middlewares are collected by application services through the following two types of queries to the XCREAM framework: a snapshot query and a continuous query. The former requests the framework immediate real-time identification or sensor data. Whereas, the latter requires the framework to keep the registered query in an active state and send the matching results to the application services during a specified period.

The Collector agent is connected to various multi-device middlewares through multi-threaded socket connections on the pre-allocated port and is in charge of forwarding the identification or sensor data to the Event Handler after converting them to the corresponding Java objects, called “ReportsArrivedEvent” (see “Figure 9”).

### **3.2.3 The Proxy Agent**

The Proxy agent is responsible for shortening the overall-event processing time. An XLogic script, which has been executed, is in the memory as an executable Java object rather than the XML script itself saved in the repository. This buffering scheme remarkably reduces parsing time compared to the fetch-and-execution scheme from the repository. This makes it possible to effectively manage the system resources throughout the whole lifetime of the XLogic script.

The Proxy agent usually extracts the unique identification information of “ReportsArrivedEvent,” which is forwarded by the Event Handler. If the value equals to one of the XLogic script residents in memory, then the XLogic is converted into “XLogicScriptExecuteRequestEvent” and sent back to the Event Handler. If it is not found in memory, then the XLogic in XML form is queried and then parsed to be placed in memory as an executable Java object. Accordingly, the event is delivered to the Event Handler.

### **3.2.4 The Event Activation Agent**

The Event Activation agent not only executes the XLogic script but also keeps the statistics of the active XLogic, including the success and failure ratio, the last completion time, and the current status. This information is presented online in real-time through a web interface, the XCREAM Enterprise Manager (XEM) and helps the user establish an execution strategy based on the acquired statistical data.

### 3.2.5 The Web Application Service (WAS) Agent

The WAS agent exposes the useful functions to the service providers by providing users with Web interfaces. It acts as the web server of the XEM through which service scenarios are registered to the XCREAM framework. The following two figures are the XEM interfaces for showing server configuration (“Figure 10”) and service scenario management (“Figure 11”) through XLogic script language.

The screenshot shows a web interface titled "Server Status : Server Settings". At the top, there are navigation tabs: "Server Status", "Object ID Provider", "XLogic Script", and "User Account". Below the title, there is a "Description" section with a red star icon and the text: "This page is a 'Server Settings' page. Shows the status of the server." To the right of the description are two buttons: "Do Garbage Collection" and "Refresh". Below this is a table showing memory usage:

Used Memory	Free Memory	Total Memory	Max Memory
22716984 byte	10640840 byte	33357824 byte	133234688 byte

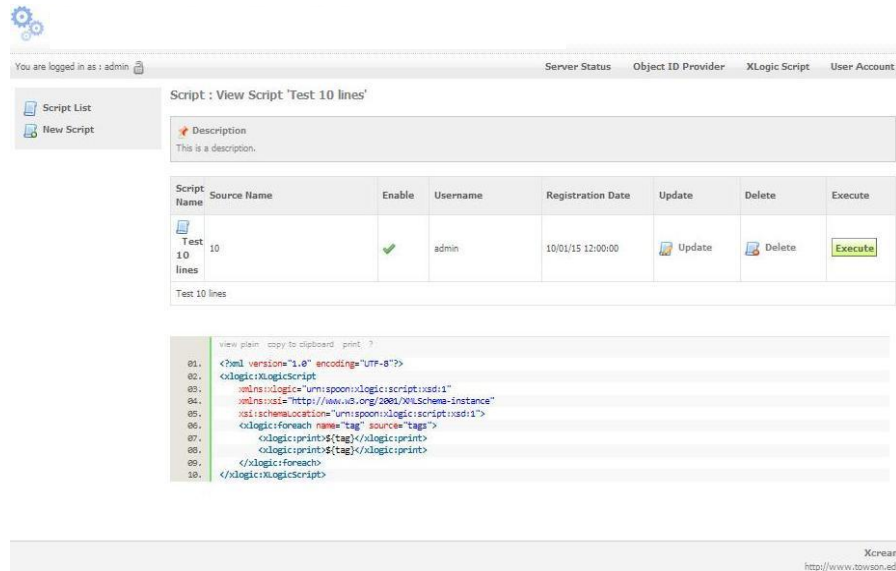
Below the memory table is another table titled "XLogic Middleware Platform Property" with the following data:

XLogic Middleware Platform Property	Value
xlogic.agent.proxy.maxSize	70
xlogic.agent.webserver.webappPath	C:/Documents and Settings/master/workspace/xem/WebRoot
xlogic.agent.manager.eventBroadcasterPoolSize	700
xlogic.agent.subscriber.poolSize	50
xlogic.agent.proxy.loadFactor	7.5
xlogic.agent.webserver.port	8010
xlogic.database.path	./db
xlogic.agent.subscriber.port	5000
xlogic.agent.instancemanage.maxExecutionThreadPoolSize	700

**Figure 10. The XEM Interface (Server Setting)**

### 3.2.6 The Phase-I XOnt Agent

The context-aware system plays an important role in the current pervasive computing environment because it recognizes and propagates contextual information in certain situations, such as a person’s or object’s current location, available network connection, room temperature, and air pollution level in a specific region.



**Figure 11. The XEM Interface (Service Scenario Management)**

In the XCREAM framework, a certain context is recognized by a rule engine that is integrated into the XOnt agent. The rule engine analyzes all the incoming events, reasons the relationship between them according to the registered rules, and determines the best-fit context situation. The framework is designed to trigger appropriate action(s) associated with service scenarios.

This approach is expected to increase the usability of contextual information including identification information or sensor data because the XOnt agent triggers related services and enables service providers to develop context-aware services. The XOnt agent consists of the following components:

### 1) XOnt Agent Implementation

The XOnt Agent Implementation component is in charge of managing the Fact Generator and the Knowledge Controller. In addition, it communicates with the Event Handler. This component usually extracts the unique identification

information from the “ReportsArrivedEvents,” which are 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 the events in its queue into facts and forwarding them to the rule engine of the XOnt agent.

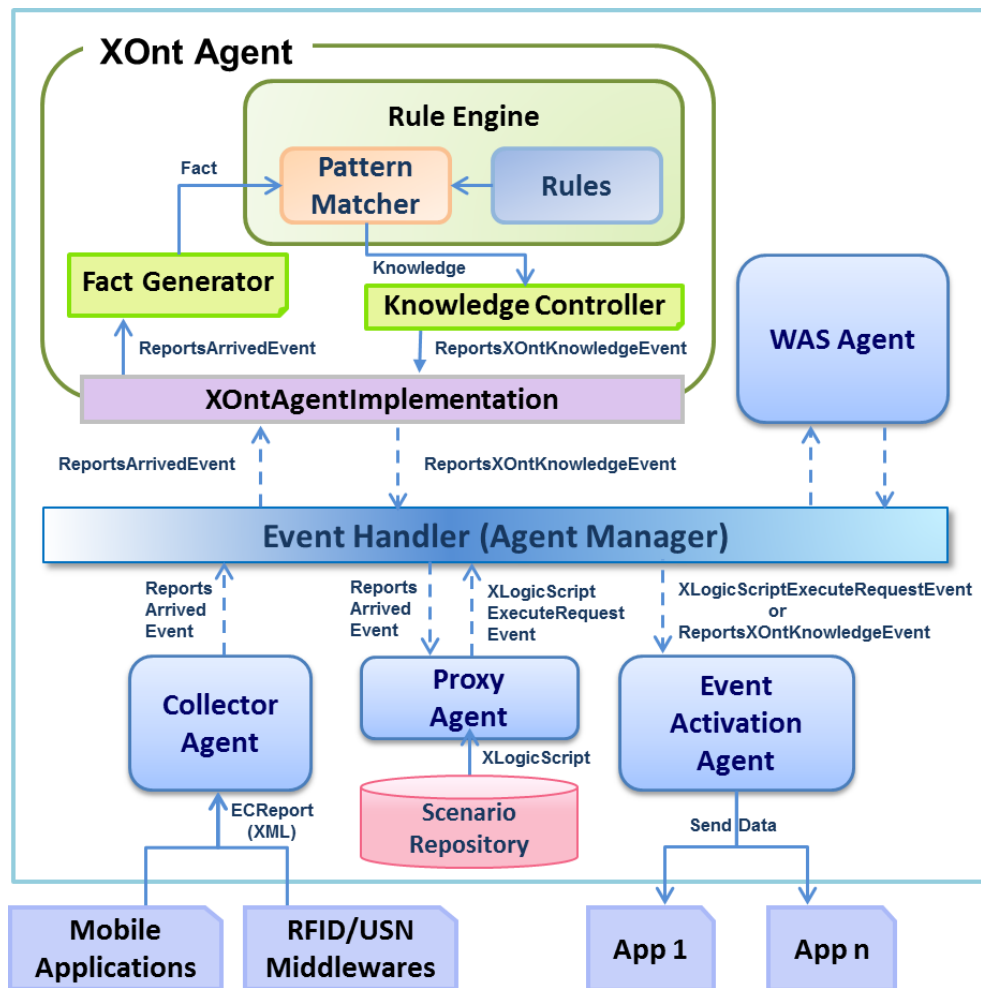
## **3) Rule Engine**

The rule engine is integrated into the XOnt agent to help the XCREAM framework recognize a special situation (context) by combining contextual information of an environment. The Drools rule engine is chosen as a rule engine in the framework. This rule engine 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 it also transforms the knowledge into its corresponding “ReportsXOntKnowledgeEvents” event. Once the event has been generated, the controller forwards it to the Event Handler via the XOnt Agent Implementation.

In order to describe the reasoning process of the framework in detail, the event flow of a rule matching process which is initiated by the Event Handler, is shown in “Figure 12”.



**Figure 12. Rule Matching Process of the Phase-I XOnt Agent**

The Drools [72] rule engine has been integrated into the Phase-I XOnt agent [11], [47]. The Phase-I agent validates the feasibility of the reasoning option within the XCREAM framework. In the Phase-II agent, this option will be further extended and combined with ontology features.

### **3.3 XOntology**

As described earlier, the current computing environment contains lots of smart devices working with corresponding application services. This situation requires us to introduce a communication scheme between them which includes a flexible interface with common vocabularies.

#### **3.3.1 Ontology Adoption**

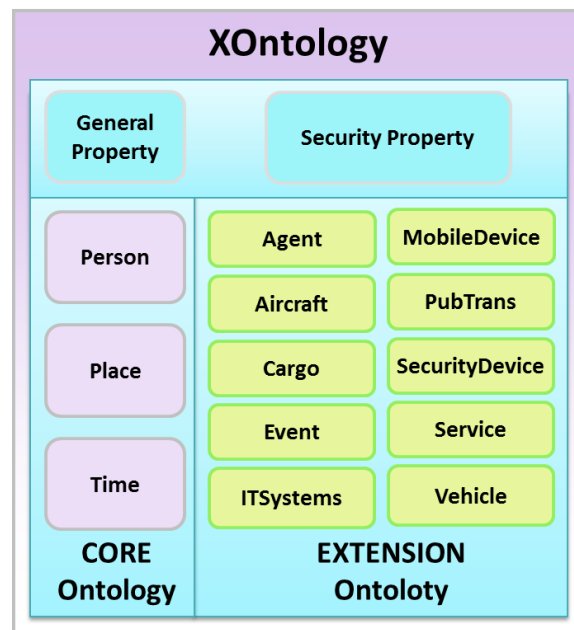
The XCREAM framework needs to understand various kinds of context information according to a common context description scheme in order to enhance the effectiveness in representing and sharing information. The context representation of the framework is described by the XOntology scheme, which is composed of the core and extension ontologies. By defining the common ontologies, identified context is understood and shared between the related parties of the framework. In addition to defining the common concepts as XOntology classes, the relationships between the ontology classes are also defined as properties.

#### **3.3.2 XOntology Components**

The XOntology is devised as an extensible structure for the XCREAM framework. It is composed of the core and extension ontology sets. The relation between ontology sets are defined as properties, which includes general and domain specific properties like the security property. The

following “Figure 13” introduces the hierarchical structure of the XOntology of the smart airport mobile security control system.

The core ontology classes encompass person, place, and time concepts, and the extension ontology describes high-level concepts that depend on domain specific concepts or entities. The example ontology describes the ontology hierarchy of an application domain of the smart airport mobile security control system, which includes agent, aircraft, cargo, event, IT systems, mobile device, public transportation, security device, service, and vehicle. The individual classes are not only drawn by analyzing real-world airport management rules, regulations [77], and the latest airport IT solutions [81], but they are also derived by combining general security issues in the computer security handbook of NIST (National Institute of Standards and Technology) [75].



**Figure 13. The XOntology of Smart Airport Mobile Security Control**



**Table 2. Ontology Classes**

<b>Classes</b>		<b>Subclasses</b>
<b>Core Ontology Classes</b>	Person	Passenger, Employee, FlightCrew, Contractor, etc.
	Place	Terminal, CheckIn, SecurityCheck, PassportControl, Gate, ImmigrationCheck, BaggageClaim, Custom, SecurityIDArea, Lounge, ParkingLot, AirCargoHandlingArea, AirOperationsArea, Clinic, FireStation, PoliceStation, QuarantineStation, etc.
	Time	TimeInterval, ExactTime, etc.
<b>Extension Ontology Classes (for Smart Airport Mobile Security Control)</b>	Event	USNInfo, LocationInfo, Behavior, etc.
	Service	Security, Business, Administrative, etc.
	Vehicle	CargoDolly, CargoLoader, Trailer, Truck, Bus, FuelingVehicle, PassengerStair, RampEquipment, etc
	Cargo	Freight, Baggage, Animal, etc.
	Aircraft	Airplane, Helicopter, etc.
	Agent	Airlines, Logistics, Police, Firefighting, Hospital, Construction, Catering, Cleaning, etc.
	SecurityDevice	RFIDReader, USNSensor, CCTV, WirelessAP, etc.
	MobileDevice	Smartphone, Laptop, HandheldDevice, Camera, etc.
	ITSystems	AerialControl, PassengerMonitoring, FacilityManagement, BuildingOperation, ImmigrationControl, CargoHandling, SecurityControl, FlightSchedule, AgentITSystems, etc.
PubTrans	Rail, GroundBus, etc.	

The constituent subclasses of the core and extension ontology classes in the XOntology scheme are described in “Table 2.”

In addition, relationships between classes and attributes of a member of a class form property hierarchy in the XOntology scheme. The properties are classified into the General Properties, commonly applied properties across the core ontologies, and the Extension Properties, domain specific properties (see “Table 3”).

The XOntology hierarchy, subclass-superclass hierarchy, is defined by using the Protégé ontology development tool, where the ontologies written in the Protégé are exported into a variety of formats including the RDF(s), the OWL, and the XML Schema [15], [78], [79].

**Table 3. Ontology Properties**

Property Category	Subproperties
General Properties	hasID, hasOwner, hasLocation, hasExactTime, hasInterval, etc.
Extension Properties (for Smart Airport Mobile Security Control)	hasAccess, hasConnection, hasSecurityLevel, hasAlert, hasThreat, etc.

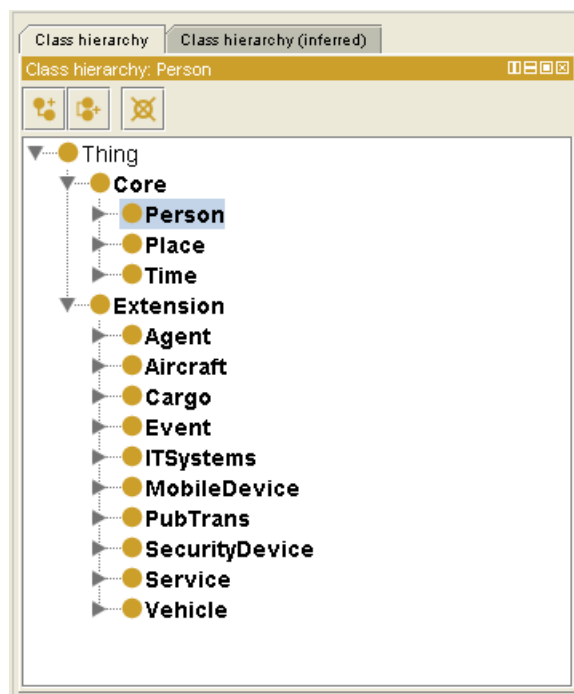
### 3.3.3 XOntology Modeling

This section explains the XOntology modeling specifications designed on Protégé. Once the XOntology for a mobile security system has been developed,

it can be shared and reused by many applications, featuring a context-aware mobile security control option.

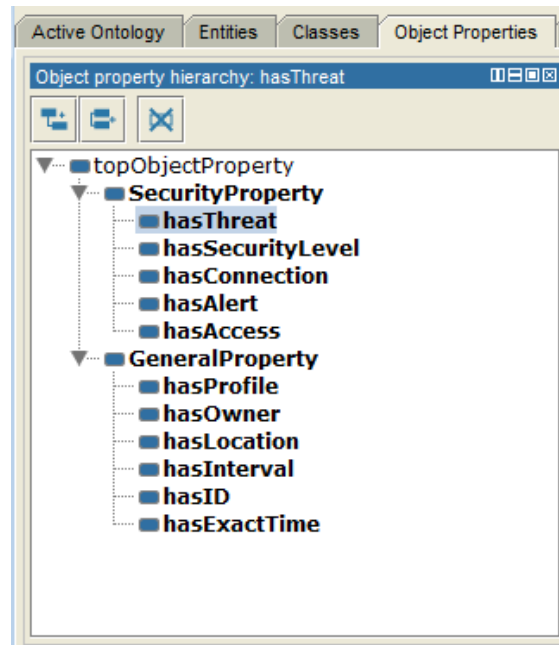
Before we proceed to develop the ontology hierarchy of a domain, we need to list the key terms and their properties [44] as in the previous two tables. After then, the derived terms are defined as classes of the domain as in the following “Figure 14.”

OWL as an ontology language has three types of properties: object properties, datatype properties, and annotation properties [15]. The object properties explain the relationships between two resources (individuals of classes), whereas the datatype properties relate a certain data value to an individual. The annotation properties are used to add meta-data to classes, properties, and individuals.



**Figure 14. XOntology Class Hierarchy on Protégé**

The following “Figure 15” includes the object properties of the XOntology which are categorized into general properties and extension properties for the purpose of mobile security control.



**Figure 15. XOntology Property Hierarchy on Protégé**

A property is a way of describing a relationship between individuals, which are members of classes. An individual is an object in the real-world and is related to other objects and to data values using datatype properties [9].

### 3.3.4 Phase-II XOnt Agent

As a rule engine of an open source project, the Drools [72] has been chosen and integrated with the XOnt agent in the Phase-I XOnt agent [47] for its highly practical application mechanism and strong support by the JBoss Community [72]. The Phase-I was used to validate the feasibility of the

reasoning option within the XCREAM framework. Naturally, the research was extended to support the OWL ontology scheme for better expressive power across a wide range of applications within the framework or even in the Internet. Especially, the fact that the Drools features its own rule format and reasoning function encouraged me to consider the OWL ontology scheme and its associated inference mechanism for the Phase-II XOnt agent.

The Phase-II XOnt agent is integrated with the Jena API [70]. It allows the ontology-based applications to build and share the ontology model and communicates with each other. Once an event has been delivered to the Event Handler, it is sent to the Context Mapper of the XOnt agent, and it is determined whether the event fits the contextual information associated with the predefined rules. The RDF triples, stored in the Ontology Context Repository, are queried and updated by the SPARQL (SPARQL Protocol and RDF Query Language) through the SPARQL API [38], [71] within the Jena API.

The Phase-II XOnt agent is under development, defining XOntology with core and extension ontologies in terms of the smart airport mobile security control system. The Phase-I XOnt agent based on the rule engine is combined with the ontology reasoning scheme in the Phase-II system [2], [16], [39]. The following “Figure 16” shows the component and event flow of the Phase-II XOnt agent with Jena API.

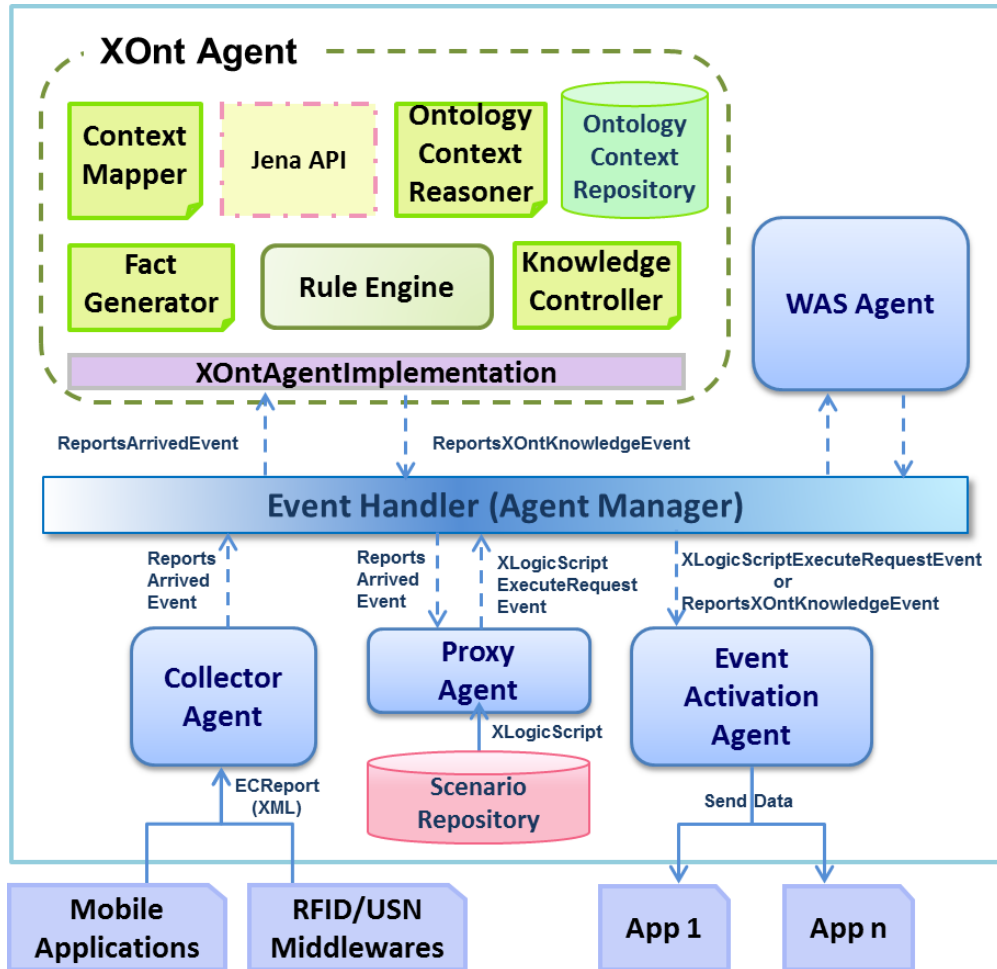


Figure 16. Phase-II XOnt Agent with Jena API

### 3.4 XLogic Script Language

The initial motivation for the development of the XCREAM was the construction of the collaboration framework for the various kinds of information systems used in many independent sites as well as the newly introduced multi-device enabled application services. As the solution to a completely seamless interfacing scheme, the research team designed an XML based infrastructure interface scheme, the XLogic script language. The external systems just need to register XLogic scripts of the service scenarios to

the repository, which correspond to the RFID identification or the USN sensor data through the XEM of the XCREAM framework.

The XLogic script language follows the XML-based scheme and provides the application services with the following statements in the form of the XML tags:

- *XLogicScript* statement
- *invokeWebService* statement
- *iterator* statement
- *set* statement
- *if, then, else* statements
- *while, foreach* statements
- *wait* statement
- *continue, break* statement
- *print* statement

In addition, the following statements are to be added to the existing set in order to support ontology-based context-aware inference.

- *XOntScript* statement
- *object* statements
- *location* statement
- *time* statements

“Figure 17” shows a part of the XLogic script, which registers a service called “PassengerInfoService” with the final destination of the tag data to be sent to the specified URL.

```
<xlogic:invokeWebService service="PassengerInfoService"
  url="http://smartairport.caiair.com/pm/flight"
  name="find_passenger">
  <rawtags xmlns="smartairport">
    <xlogic:iterator name="tag" source="tags">
      <tag>${tag}</tag>
    </xlogic:iterator>
  </rawtags>
</xlogic:invokeWebService>
<xlogic:set name="passenger" select="//id">${find_passenger}
</xlogic:set>
```

**Figure 17. XLogic Script for “PassengerInfoService”**

The XLogic script language and its specifications are described in more detail in Appendix C and D.

### 3.5 XLogic Script Execution Modes

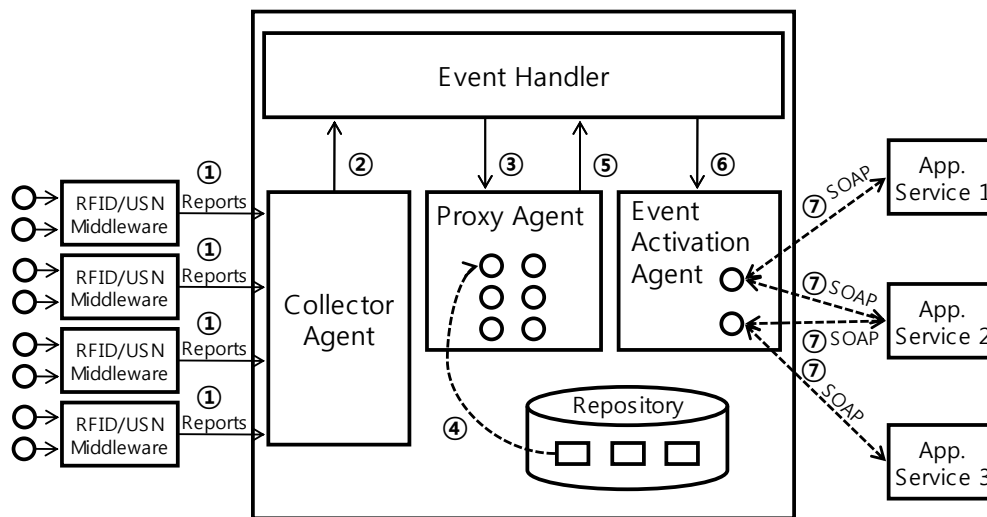
Incoming events from data sources are processed in two modes within the framework: a trigger mode in which a received event will activate a pre-registered XLogic script and an immediate mode in which an XLogic script is



converted into a Java object and executed immediately as soon as the script has been registered to the XCREAM framework.

### 3.5.1 Trigger Mode

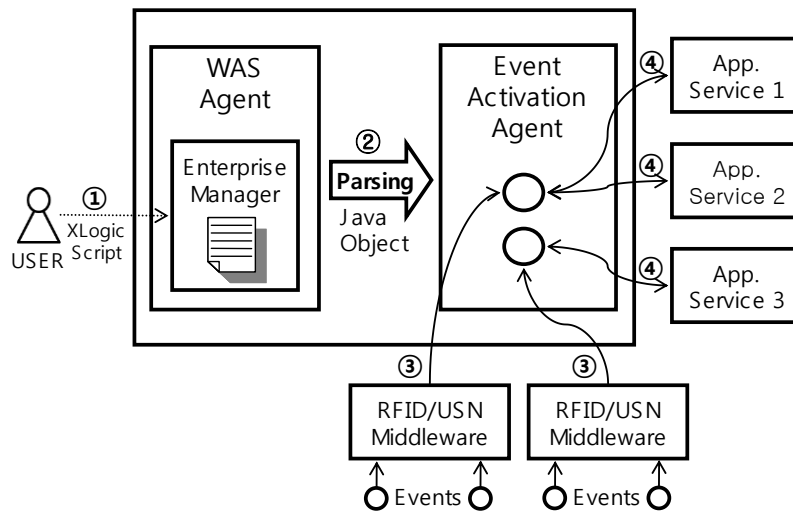
An XLogic script, which is defined as trigger mode, is executed when the external event is collected by the XCREAM according to the flow of event in “Figure 18.”



**Figure 18. Trigger Mode Execution**

### 3.5.2 Immediate Mode

Immediate mode refers to the mode in which an XLogic script written by a user is executed immediately. If a user writes and executes an XLogic script through the XEM, then it is sent to the Event Activation Agent through the WAS Agent and then the XLogic script written in the XML form is transformed into a Java object and then the Java object is executed (see “Figure 19”).



**Figure 19. Immediate Mode Execution**

### 3.6 Context-Aware Inference (CAI) Scheme

The Context-Aware Inference (CAI) scheme [47] suggested in this research uses rule engine, the Drools [72], [73], that matches the pre-registered rules in its rule base with the real-time contextual information, which is referred to as facts. The CAI scheme enables time-varying combination of facts to be matched with the context-aware situation analysis rules according to their constraints (conditional elements of the rules).

#### 3.6.1 Smart Airport Environment

A smart airport is chosen as a prototype environment where the CAI scheme is applied to the smart airport mobile security control option. The major entities of the model are described in section 3.3.2 (see “Figure 13”), including passengers, sensor generating events, air cargoes, and additional smart airport facilities and their corresponding ontology classes are also introduced in

“Table 2.” The ontology classes are categorized into the core ontology for the basic identification of person or any objects with additional information like identified place and time information and the extension ontology for the smart airport mobile security control option. All the concepts and detailed properties of the smart airport environment are shared across the agents and their IT systems through the XOntology of the XCREAM framework.

### **3.6.2 Situation Analysis Rules**

The Context-Aware Inference (CAI) scheme in this research is supported by the assumptions and constraints of the smart airport environment, which are necessary to reason the expected context in the example environment and categorized in the form of the situation analysis rules: the Existence Availability rule (EA Rule); the Access Availability rule (AA Rule); the Expected Transit Time rule (ETT Rule); and the Sensor Range rule (SR Rule). These are used as the fundamental knowledge base of the XCREAM framework. All the facts like real-time contextual information such as USN sensor data and identification information of passengers, employees, and even cargoes are matched according to the fundamental rules [73].

#### **1) Existence Availability Rule (EA Rule)**

The expected pairs of places and their associated availability of simultaneous existence are listed in “Table 4,” which lists the constraints of the Existence Availability rule.

**Table 4. Existence Availability Rules**

<b>Place 'A'</b>	<b>Place 'B'</b>	<b>Sync. Availability</b>
CheckIn	SecurityCheck	Yes
SecurityCheck	PassportControl	Yes
SecurityCheck	ImmigrationCheck	No
PassportControl	Gate001	Yes
Gate	ParkingLot	No
Lounge	Gate002	Yes
Gate001	ImmigrationCheck	No
Gate020	ParkingLot	No
Gate101	ParkingLot	No
Gate120	ImmigrationCheck	Yes
CheckIn	BaggageClaim	No
...	...	...

The Existence Availability rule defined in “Figure 20” is used for validating identification results and figuring out the abnormal identification. Based on the Existence Availability rule, the rule engine is to decide whether a person could appear at different places at the same time.

“Figure 21” shows that a person (“Kyungeun Park”) is identified at different places at the same time and indicates that she may cause a certain security violation.

```

rule "Existence Availability Rule"
when
    $PR1: PersonRecognition ($PRperson1 : person,
                             $PRplace1 : place)
    $PR2: PersonRecognition (person == $PRperson1,
                             place != $PRplace1,
                             $PRperson2 : person,
                             $PRplace2 : place)
    $PR3: eaRules (place1 == $PRplace1,
                  place2 == $PRplace2, sync == false)
then
    System.out.println("EA Violation::" + $PRperson1 +
                      ": @" + $PRplace1 + " and " +
                      "@ " + $PRplace2);
end

```

**Figure 20. Existence Availability Rule**

```

EA Violation::Kyungeun Park: @SecurityCheck and
@ImmigrationCheck

```

**Figure 21. Matching Result of Existence Availability Rule**

## 2) Access Availability Rule (AA Rule)

The Access Availability rule is used for validating whether any person is passing or staying unapproved area, scanning all recognized person objects in working memory for any possibility of illegal pass. Many places in an airport and their associated ID type(s) are listed in the following “Table 5.”

**Table 5. Access Availability Rule**

<b>Place</b>	<b>Accessible ID</b>
CheckIn	“N” (Not required)
SecurityCheck, PassportControl, Lounge, Gate, Gate001, Gate020	“T” (Departure Flight Boarding Pass), “I” (Any ID)
Gate101, Gate120, ImmigrationCheck, BaggageClaim, Custom,	“R” (Arrived Flight Boarding Pass), “I” (Any ID)
AirOperationsArea	A, B
AirCargoHandlingArea	C
QuarantineStation	D
SecurityIDArea	A, B, C, D
...	...

The Access Availability rule validates whether identified person or employees at a certain place of an airport has eligible to be there with a proper ID card or boarding pass. The conditional part after “when” matches ID type of a person identified (see “Figure 22”).

“Figure 23” shows several access violations resulted from illegal access with invalid ID. For example, in order to access “AirOperationsArea”, the “A” or “B”-type IDs are required and a person with departure boarding pass should not be recognized in “ImmigrationCheck.” In addition, departure boarding pass or employee ID are required to access at boarding processing places like “SecurityCheck” or “PassportControl.”

```

rule "Access Availability Rule"
  when
    $PR1: PersonRecognition ($PRperson : person,
                             $PRplace : place)
    $AA1: aaRules (getPlace() == $PR1.getPlace(),
                  (getIDType() not contains "I" &&
                   getIDType() not contains $PR1.getID() ||
                   (getIDType() contains "I" &&
                    $PR1.getID() == "N") ||
                    ($PR1.getID() == "T" &&
                     getIDType() not contains "T" ) ||
                    ($PR1.getID() == "R" &&
                     getIDType() not contains "R")),
                  getIDReq() == true)
  then
    System.out.println("AA Violation::" + $PRperson + " (" +
                       $PR1.getID() + "): " + "appeared @ " + $PRplace +
                       " illegally. ");
  end
end

```

**Figure 22. Access Availability Rule**

```

AA Violation::David Texas (C): appeared @ AirOperationsArea
illegally.

AA Violation::Amanda Maryland (C): appeared @ AirOperationsArea
illegally.

AA Violation::Kyungeun Park (T): appeared @ ImmigrationCheck
illegally.

```

**Figure 23. Matching Result of Access Availability Rule**

### 3) Expected Transit Time Rule (ETT Rule)

The Expected Transit Time rule is used to determine the earliest (MinTime) or the longest (MaxTime) expected time of arrival to move from one location to another (see “Table 6”). Each combination of two different places, where RFID reader or any other identification facilities are installed, is assigned MinTime and MaxTime, presenting time constraints. This information is used to deduce an abnormal behavior of individual passengers or employees in an airport by comparing time difference between the expected transit time and the actual time spent to move. “Figure 24” describes the Drools rule to define the Expected Transit Time rule and capture the abnormal behavior to help the smart airport mobile security control system find and handle immediately.

**Table 6. Expected Transit Time Rule**

LeavingFrom	ArrivingAt	MinTime (min)	MaxTime (min)	Sync.
CheckIn	SecurityCheck	5	150	Yes
SecurityCheck	PassportControl	5	30	Yes
SecurityCheck	ImmigrationCheck	-300	0	No
PassportControl	Gate001	0	100	Yes
Lounge	Gate020	0	60	Yes
Gate001	ImmigrationCheck	-300	0	No
Gate020	ParkingLot	-300	0	Yes
Gate101	ParkingLot	10	100	No
Gate120	ImmigrationCheck	0	60	Yes
CheckIn	BaggageClaim	-300	0	No



“Figure 25” tells “Amanda Maryland” spent 75 minutes to move from “Lounge” to “Gate020,” which takes 15 minutes longer than the maximum transit time, registered with 60 minutes. This matching result will alarm the system administrator and make him (her) keep track of the person and take security measures, if needed.

```

rule "Expected Transit Time Rule"
  when
    $PR1: PersonRecognition ($PRperson1 : person,
                            $PRplace1 : place, $PRtime1 : time)
    $PR2: PersonRecognition (person == $PRperson1,
                            place != $PRplace1, $PRperson2 : person,
                            $PRplace2 : place, $PRtime2 : time)
    $ETT1: ettRules (place1 == $PRplace1,
                   place2 == $PRplace2, getSync() == true,
                   $ettMinTime : minTime, $ettMaxTime : maxTime)
    $PR3: PersonRecognition (!$PR1.withinTimeRange($PR2,
                                                  $ettMinTime, $ettMaxTime))
  then
    System.out.println("ETT Violation::" + $PRperson1 +
                      ": " + ($PRtime2 - $PRtime1) + "(min) spent between @ " +
                      $PRplace1 + " and " + "@ " + $PRplace2 );
  end

```

**Figure 24. Expected Transit Time Rule**

```

ETT Violation::Amanda Maryland: 75(min) spent between @ Lounge
and @ Gate020

```

**Figure 25. Matching Result of Expected Transit Time Rule**

#### 4) Sensor Range Rule (SR Rule)

The allowable ranges of sensors are provided in “Table 7” [82]. An abnormal sensor value is to be identified by the analytic matching procedure of the rule engine and the associated action is performed.

**Table 7. Sensor Range Rule**

Sensor	Min	Max
Temperature (°F)	-10	130
Illumination (lux)	100	1,000
Noise (dB)	0	80
CO (ppm)	0	9
Ozon (ppm)	0	0.12
Lead (ppm)	0	100
FineDust (µg)	0	150

```
rule "Sensor Range Rule"
  when
    $SE1: SensorEvent ($SEsensor : sensor, $SEvalue : value,
                      $SEtime : time, $SEplace : place)
    $SR1: srRules (getSensor() == $SEsensor,
                  $SEvalue < getMinValue() || > getMaxValue() )
  then
    System.out.println("SR Warning::" + $SEsensor + " @ " +
                      $SEplace + " : " + $SEvalue +
                      " (min) " + $SR1.getMinValue() +
                      " (max) " + $SR1.getMaxValue() );
  end
```

**Figure 26. Sensor Range Rule**

```

SR Warning::Ozon @ AirCargoHandlingArea : 0.15 (min) 0.0 (max) 0.12
SR Warning::Lead @ AirCargoHandlingArea : 200.0 (min) 0.0 (max)
100.0
SR Warning::CO @ AirCargoHandlingArea : 12.0 (min) 0.0 (max) 9.0
SR Warning::Noise @ AirCargoHandlingArea : 200.0 (min) 0.0 (max)
80.0
SR Warning::Illumination @ AirCargoHandlingArea : 50.0 (min) 100.0
(max) 1000.0

```

**Figure 27. Matching Result of Sensor Range Rule**

“Figure 26” describes Sensor Range rule and recognize abnormal sensor status based on the available range values. In “Figure 27”, abnormal cases are listed after matching the SR rule.

### **3.6.3 Context-Aware Mobile Security Option**

Based on the essential knowledge of the situation analysis discussed in the previous section, one-step forward reasoning will be available to develop wide range of dynamic solutions which basically request agile reaction to constantly changing environment. Among many choices, the context-aware mobile security option can draw keen attentions from real-world solutions, because the more the smart ubiquitous computing environment is available with various USN sensors and advanced identification technology, the more increasingly the service providers face strong demands to customized and responsive functions as well as crucial security control options. It is true that the inclusion

of the newest technology encourages us to develop far more personalized solutions for each personnel than ever. This circumstance, however, results in growing possibility of taking advantage of the acquired information or abusing them on purpose. By this reason, it is not too much to say that almost every IT solutions might be secured with the context-aware mobile security option against malicious intruders, making ill use of the well-established high-tech environment.

The situation analysis forms fundamental four rules of an environment by analyzing the behavior and status of the objects and defining the assumptions and constraints of the objects. Accordingly, the general knowledge of the context-aware mobile security option can be summarized into the following general models, upon reviewing the situation analysis cases.

```
rule "Determine if an object is supposed to be read here."  
  when  
    $PR1: PersonRecognition($PID :  
                                isNotSuitable(objectid, readerid)  
    $OR1: ObjectRecognition($OID :  
                                isNotSuitable(objected, readerid))  
  then  
    $PR1.notifyViolation($PID);  
    $OID.notifyViolation($OID);  
  end
```

**Figure 28. Read Suitability Model**

### 1) Read Suitability Model

This model validates whether any recognized person or object is supposed to appear at a specific place, during a certain period of time. The model plays a basic role in reasoning complex scenarios which are related with a combination of variety of information from many sensors (see “Figure 28”).

### 2) Event Supervising Model

This is also a general reasoning model with a variety of sensors. Usually, this model causes the alarm signals to be delivered to their corresponding application services, for example, a smart airport monitoring service (see “Figure 29”).

```
rule "Determine if emergency situation happens."  
  when  
    $FIRE : exists Fire( )  
  then  
    Notify($FIRE.room.getName());  
  end
```

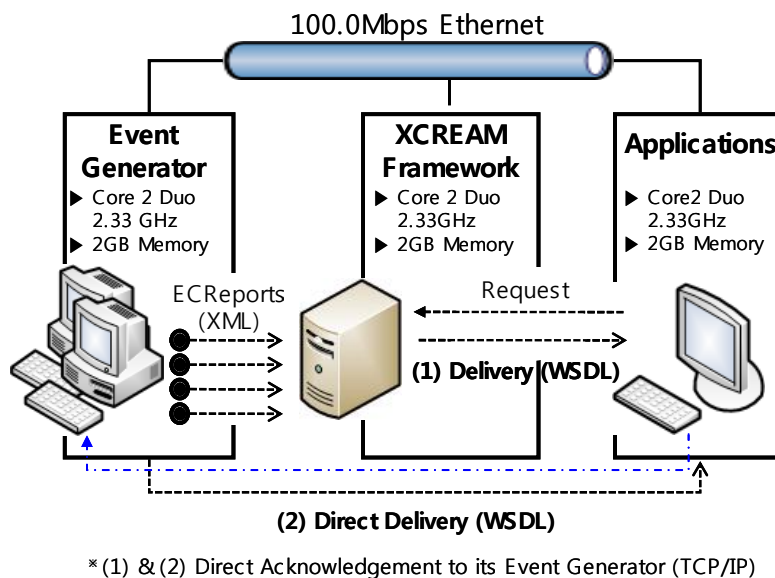
**Figure 29. Event Supervising Model**

## Chapter 4. Simulation

The XCREAM framework was tested in a simulation environment in order to show its performance and prove the validity of the collaboration. This chapter includes the description of the testbed environment as well as performance and collaboration validity test results, looking at the configuration details of individual test cases.

### 4.1 Simulation Environment

In order to evaluate the performance of the XCREAM framework, the following simulation environment has been built (see “Figure 30”). The environment includes a collection of event generators which produce sequences of ECR reports [68] containing RFID tag identification events to send to the XCREAM framework. The XCREAM framework is also interfaced with many applications and matching events trigger these services.



**Figure 30. Simulation Environment**

## 4.2 Performance Tests

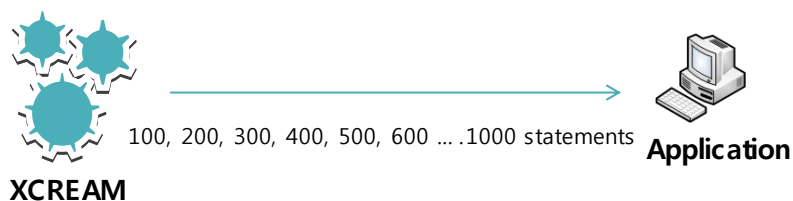
The performance test of the framework has been carried out by checking the efficiency of the framework in terms of two perspectives: parsing efficiency and event processing capacity.

### 4.2.1 Parsing Efficiency Tests

The first performance test measured the time the framework took to convert the XLogic scripts, given in XML form, into a Java object in immediate mode. For this test, multiple set of XLogic scripts, requesting numerous tag data, are adopted as in “Figure 31.”

The experiment has been accomplished by calculating the elapsed time between the start and end point of a parsing process. In order to obtain the average elapsed parsing time, each XLogic script has been parsed 20 times.

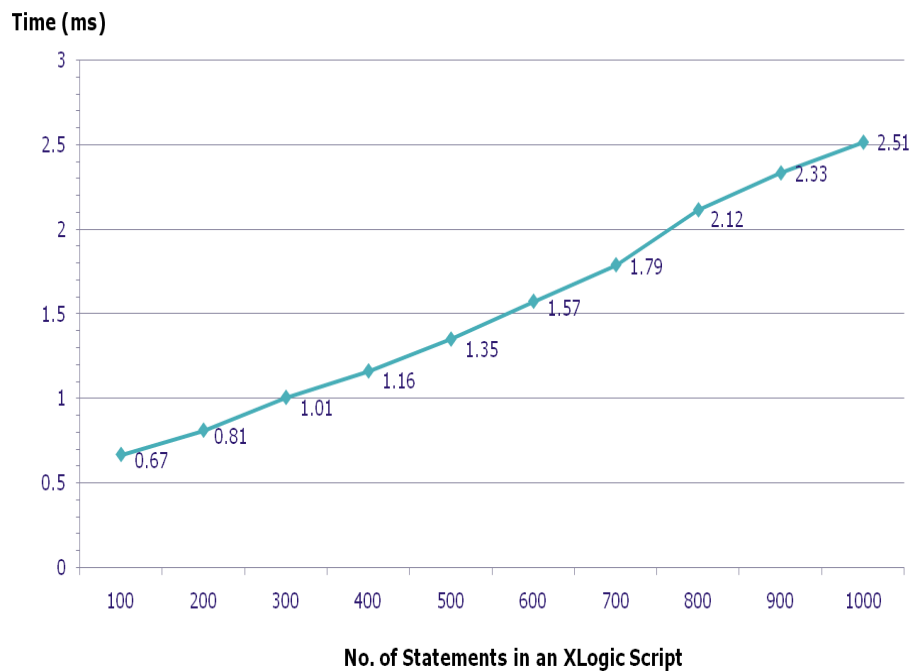
The following “Table 8” shows the results of the performance test and shows the average parsing time depends on the number of statements in the XLogic scripts. The test results show that the XCREAM framework spends linearly increasing time to parse the variable sets of XLogic scripts depending on the number of statements contained in the scripts (see “Figure 32”).



**Figure 31. Parsing Efficiency Test Model**

**Table 8. Average Parsing Time depending on the Number of Statements of the XLogic Scripts**

No. of Statements of XLogic Script (lines)	Average Parsing Time (msec)
100	0.67
200	0.81
300	1.01
400	1.16
500	1.35
600	1.57
700	1.79
800	2.12
900	2.33
1,000	2.51

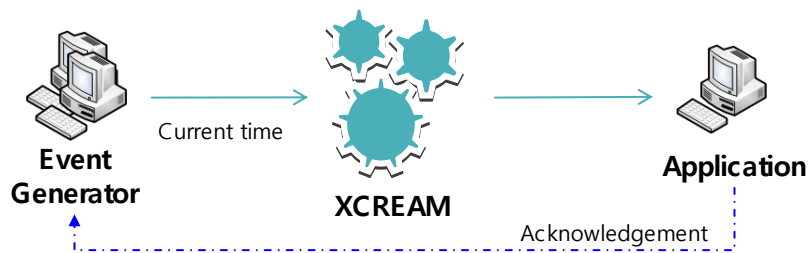


**Figure 32. Average Parsing Time**



### 4.2.2 Event Processing Tests

The event processing test is used to show the robustness of the system by taking time durations to process a number of events as shown in “Table 9”. In the testbed (see “Figure 33”), an XLogic script, registered by a specific application service in trigger mode, recognizes a RFID tag event and delivers it to the application service.

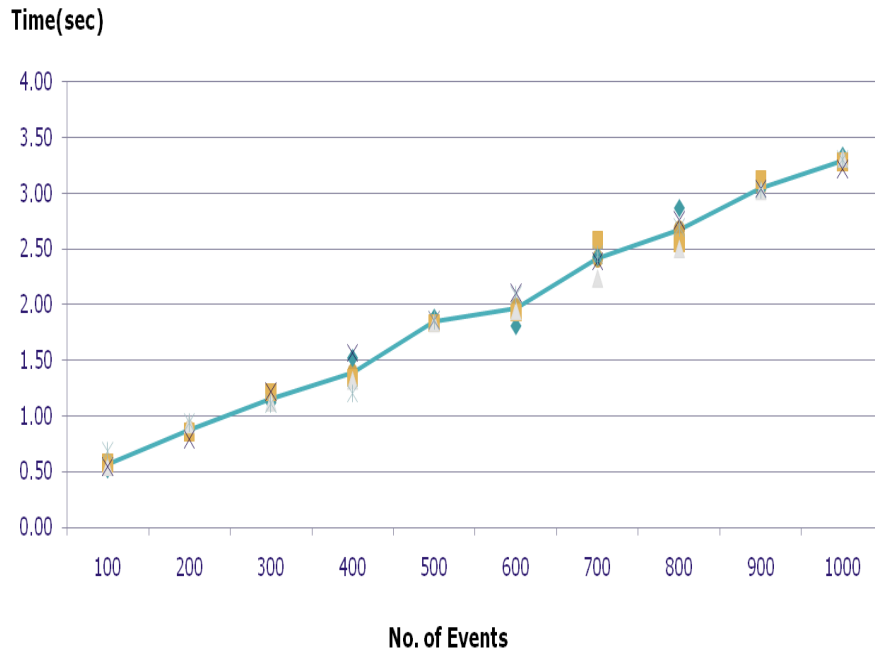


**Figure 33. Event Processing Test Model**

**Table 9. Average Event Processing Time depending on the Number of Events in Trigger Mode**

No. of Events from the Event Generator	Average Processing Time (sec)
100	0.57
200	0.88
300	1.16
400	1.39
500	1.85
600	1.97
700	2.41
800	2.67
900	3.05
1,000	3.29

The following graph (see “Figure 34”) shows the event processing time for a various number of events sent to the XCREAM framework in trigger mode.



**Figure 34. Average Event Processing Time**

Reviewing the test results of the previous two performance tests, the XCREAM framework processes huge amounts of transactions within an acceptable time delay. In the parsing efficiency test, 1,000 XLogic statements are parsed within 2.5 seconds. Thus, we can regard that the framework works reasonably because the service scenario can be defined with the convenient XLogic interface, and the service providers only need to focus on the service logics instead of the individual connections to various kinds of sensor devices.

In addition, the event processing test is completed within 3.3 seconds to process 1,000 events, especially RFID tags. It indicates that the performance of the framework is about 300 transactions per second (tps). This is more than acceptable because most commercial services require middleware servers to process about 100 transactions per second.

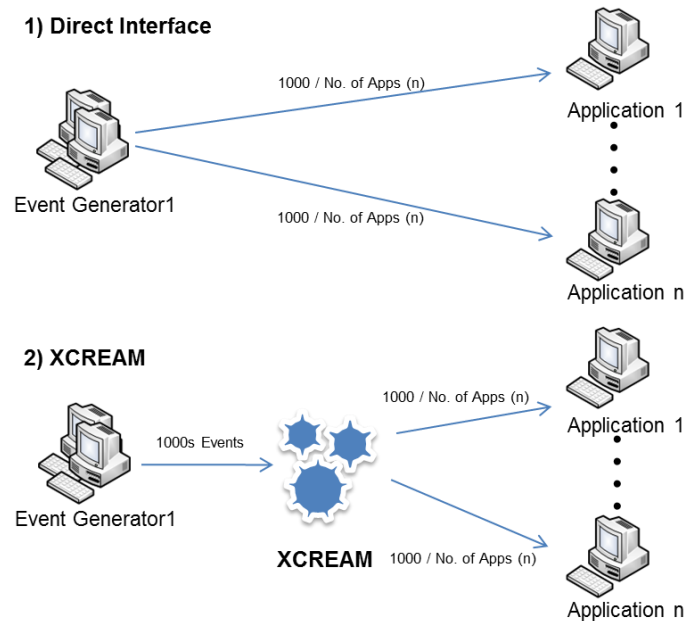
### 4.3 Collaboration Validity Tests

The main idea of developing the XCREAM is to establish a collaborating infrastructure for numerous application services, and the tests verify that collaboration validity was a success. These tests include three processing scenarios with or without the XCREAM by varying the number of event generators and application services of the simulation environment:  $1$  event generator versus  $N$  applications;  $N$  event generators versus  $1$  application; and  $M$  event generators versus  $N$  applications.

The simulation experiments, occurring at the event generator side, measure the elapsed time from just before sending tag events, ECREports of RFID tag identification event(s) generated by the event generator(s), to just after all the acknowledgements from their receivers, application services, have been received via the XCREAM framework or not.

#### 4.3.1 1 Event Generator vs. $N$ Applications

The first test configures one event generator which generates a sequence of 1,000 ECREports and a varying number of application services (see “Figure 35”). The test results (see “Table 10” and “Figure 36”) indicate that the direct interface process is faster than processing through the XCREAM because direct interface requires less transaction for the event transmission than the XCREAM model. Nevertheless, we are encouraged with the test result because it shows the XCREAM provides gradually increasing performance as well as the reliable communication interface to the individual applications.

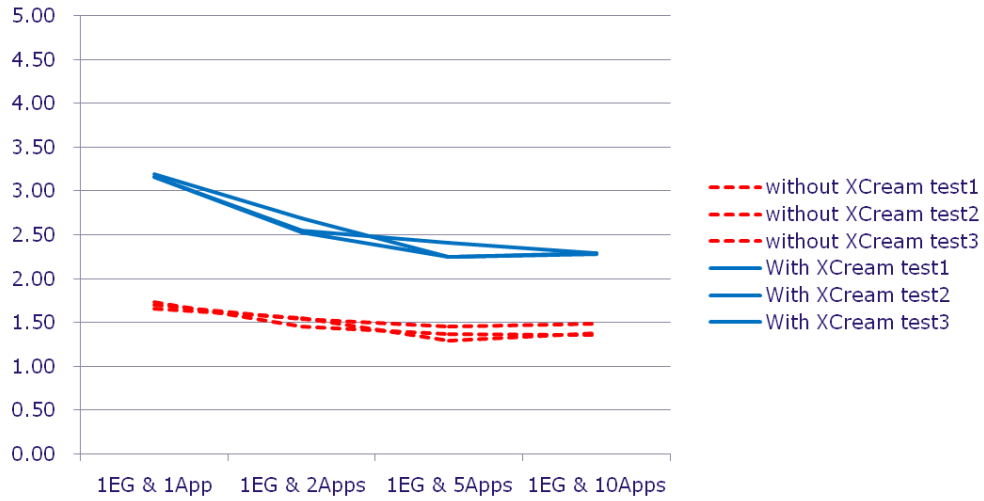


**Figure 35. 1 EG vs. N Apps Test Model**

Most importantly, this test confirms that the XCREAM framework could support highly interrelated application services to be included within the framework. This feature is the fundamental requirement in order to realize the collaboration between various applications in our target environment.

**Table 10. 1 EG vs. N Apps**

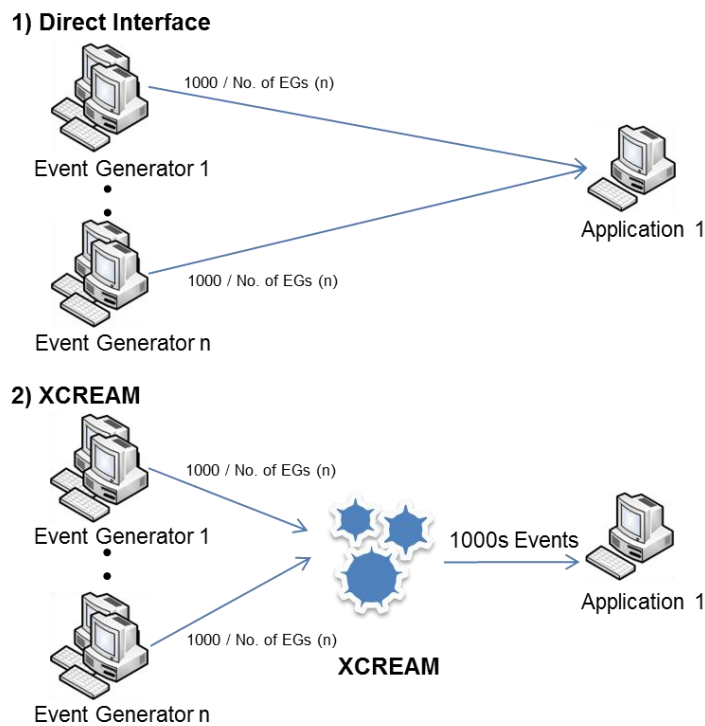
Test Type	1 EG/ 1 App	1 EG/ 2 Apps	1 EG/ 5 Apps	1 EG/ 10 Apps
<b>1) Direct Interface</b>				
No. of total transactions (connections) on EG side	1,000	1,000	1,000	1,000
Average time (sec) / tps	1.7/588	1.52/658	1.38/725	1.41/709
<b>2) XCREAM</b>				
No. of total transactions (connections) on EG side	2,000	2,000	2,000	2,000
Average time (sec) / tps	3.17/631	2.59/772	2.3/870	2.29/873



**Figure 36. Average Time for 1 EG vs. N Apps**

### 4.3.2 N Event Generators vs. 1 Application

The second experiment, described in “Figure 37,” increases the number of event generators and lets the individual event generators produce events by 1,000 divided by the number of event generators.

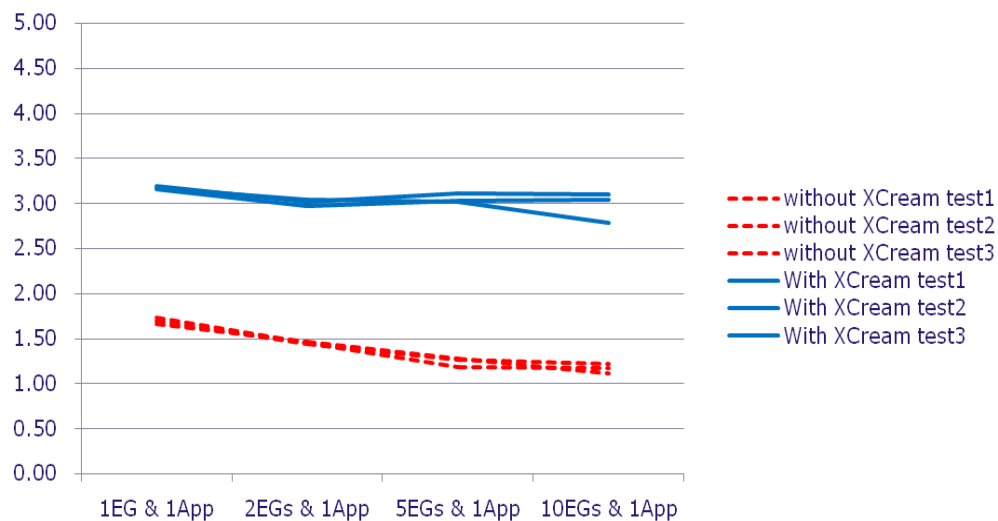


**Figure 37. N EGs vs. 1 App Test Model**

**Table 11. *N* EGs vs. 1 App**

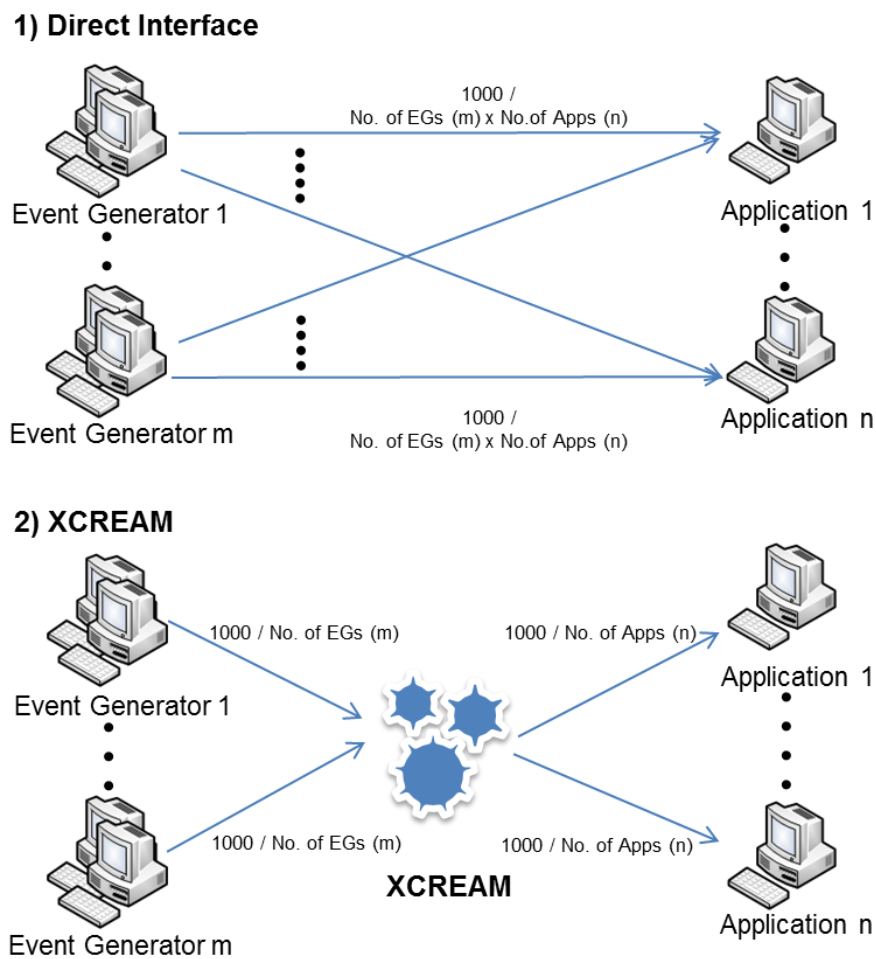
Test Type	1 EG/ 1 App	2 EGs/ 1 App	5 EGs/ 1 App	10 EGs/ 1 App
1) Direct Interface				
No. of total transactions (connections) on all EG sides	1,000	1,000	1,000	1,000
Average time (sec) / tps	1.7/588	1.45/690	1.24/806	1.17/855
2) XCREAM				
No. of total transactions (connections) on all EG sides	2,000	2,000	2,000	2,000
Average time (sec) / tps	3.17/631	3.01/664	3.05/656	2.98/671

The test results (see “Table 11” and “Figure 38”) show that the XCREAM works reliably even in an environment with numerous RFID readers and multiple-devices such as a highly networked ubiquitous computing environment.

**Figure 38. Average Time for *N* EGs vs. 1 App**

The XCREAM model in this case shows that there is almost two times a difference between direct interface and the XCREAM model. However, this case is not suitable for our target environment because the framework is developed to support the collaboration among multi-device enabled applications. According to this reason, the experiment proceeds to test  $M$  event generators versus  $N$  applications case as far more relevant environment than  $N$  event generators versus  $1$  application case.

#### 4.3.3 $M$ Event Generators vs. $N$ Applications



**Figure 39.  $M$  EGs vs.  $N$  Apps Test Model**

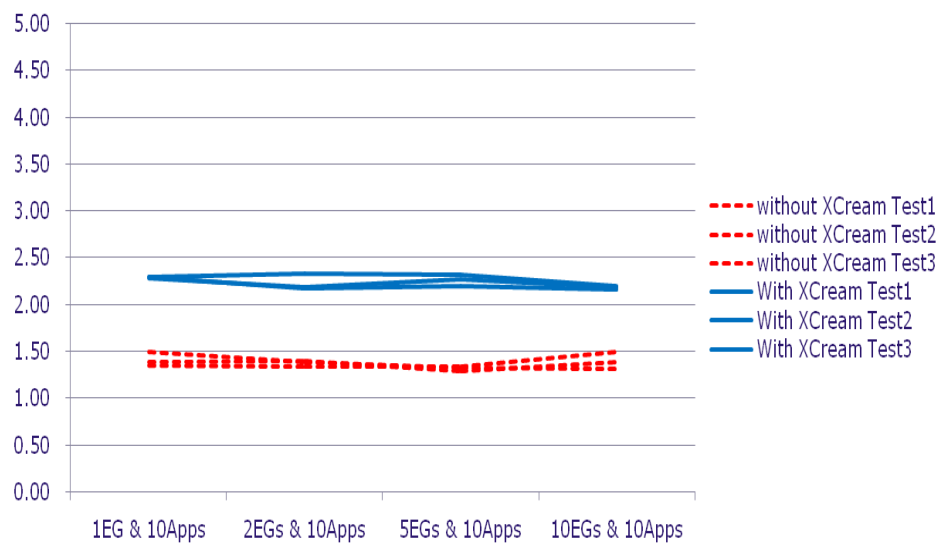
The final test is designed to check the validity of the collaboration under the XCREAM framework, looking at the real-world with varying number of the event generators and various business applications. As shown in the previous experiments, the XCREAM works better with multiple applications. Whereas, the number of event generators does not influence the overall performance significantly. According to the test results, the number of applications is fixed to ten at this time, and only the number of event generators is increased. An event generator generates events by 1,000 divided by the number of the event generators (see “Figure 39”). In addition, the XCREAM will deliver the events to the individual applications by 1,000 divided by the number of applications which is fixed with 100 in this case.

**Table 12. *M* EGs vs. *N* Apps**

Test Type	1 EG/ 10 Apps	2 EGs/ 10 Apps	5 EGs/ 10 Apps	10 EGs/ 10 Apps
1) Direct Interface				
No. of total transactions (connections) on all EG sides	1,000	1,000	1,000	1,000
Average time (sec) / tps	1.41/709	1.38/725	1.32/758	1.41/709
2) XCREAM				
No. of total transactions (connections) on all EG sides	2,000	2,000	2,000	2,000
Average time (sec) / tps	2.29/873	2.24/893	2.27/881	2.19/913



The test results (see “Table 12” and “Figure 40”) show that the XCREAM works well in an environment where multiple applications collaborate with each other. This means the XCREAM can be applied to the complex smart environment, where the orchestration among various business applications is strongly required. It is also expected to enable various application services to be flexibly added to or dropped from the framework.



**Figure 40. Average Time for  $M$  EGs vs.  $N$  Apps**

The above tests show that indirect collaboration interface has doubled the number of connections; therefore, it takes a longer time than direct interface to process the events from event generators to the individual application service agents. However, the test result in  $M$ -to- $N$  test case implies that the XCREAM works well with its competitive transaction performances.

As a result, this framework is expected to be used to build complex smart environment in which various application services are flexibly added to or dropped from.

It is true that this collaboration validity tests motivate us to develop the next phase of the framework in order to promote the development of collaborating services. The Phase-II XOnt agent is used to combine its rule matching mechanism with ontology-based representation scheme to specify a variety of sensors and their characteristics and enhance information sharing across the framework.

## Chapter 5. Conclusion

This research aims at presenting a collaboration framework, the XCREAM framework, for a context-aware environment with multi-device enabled applications as an intelligently coordinating framework over lots of application services. The framework mediates between smart devices (physical objects) and collaborative application services (cyber services) by collecting massive events from a variety of event origins and distributing them to the appropriate service parties depending on the predefined application business scenarios.

This research first built a RFID/USN middleware framework which collects RFID tag data, finds the service scenario(s) related to the tag events, and directs the associated application services to work and collaborate with each other. The application services are connected to the framework through the SOA-based Web interface, called the XEM, and integrated to the framework by registering the associated service scenario(s) that are written in the XLogic script language. This interface scheme could give many additional application services flexibility and interoperability in the future. The XLogic script language has been developed as a key element of the XCREAM framework and evolved to support a new context-aware environment by extending its tag elements in this research.

As the next step, the performance and collaboration validity tests were accomplished. In the performance tests, the XLogic script interpreting and event processing rates are reasonably acceptable, with about 400 tps and 300 tps respectively, because the framework presents the convenient service

scenario registration mechanism to the application providers. On the other hand, the collaboration validity tests take a longer time than direct interface to process the events from event generators to the individual application service agents. The *M-to-N* test case, however, shows that the XCREAM works well in a multi-device enabled environment with its competitive transaction processing capacity.

Also, application cases (see Appendix A and B) which simulate the collaboration of multiple agencies were presented to validate the collaboration capability of the framework. The proposed applications put variable heterogeneous service systems together within an infrastructure environment and made them organically flow through the XCREAM framework without disturbing any autonomy belonging to the individual application services.

In addition, the context-aware inference (CAI) scheme suggests a widely applicable collaboration model for a fast-growing pervasive computing environment. To support the CAI scheme, the XCREAM framework has been extended by adding a new agent called the XOnt agent, which basically infers the current situation out of contextual information depending on the rule matching mechanism of the XOnt agent. The framework allows the XOnt agent to deploy the XOntology specification, classifying the real-world entities in order to enhance its interoperability and reusability across 3<sup>rd</sup>-party services.

The CAI scheme motivates the design of the context-aware mobile security option which is based on a general situation analysis described by four fundamental rules: Existence Availability Rule; Access Availability Rule;

Expected Transit Time Rule; and Sensor Range Rule. The assertive adoption of the context-aware mobile security option results in transforming normal application services into context-aware security-enabled solutions.

This new attempt leads the research to the development of the Phase-II XOnt agent which shares the knowledge across the XOntology scheme and the rule engine. In addition, the application of the ontology scheme is expected to enhance the availability of the contextual information and support the future development of the finely customized services by combining the ontology scheme with the rule-based scheme of the framework. This subject is still under research and development and will be left to the next phase of the XCREAM framework.

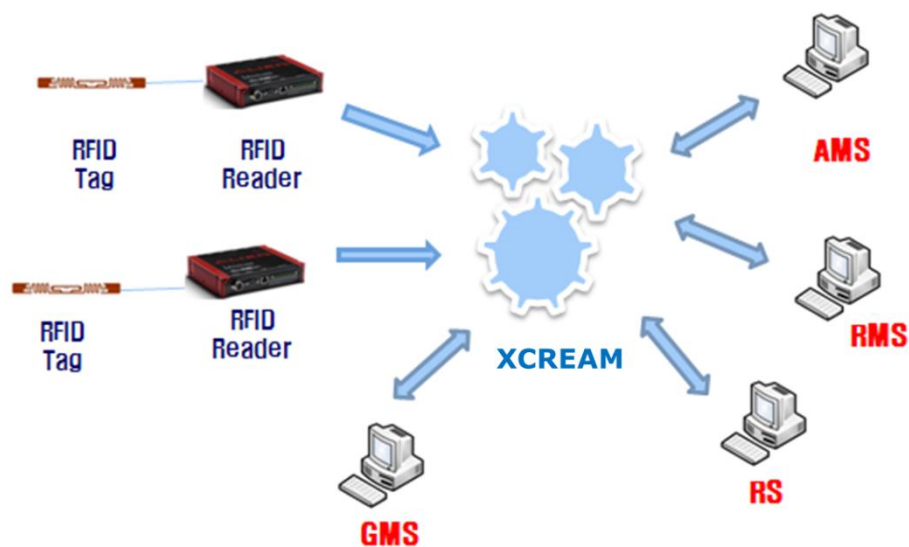
The study concludes by providing an intelligent collaboration framework, the XCREAM framework, for a context-aware environment with multi-device enabled applications. Since I started this research in 2006, the information technology environment has been rapidly innovated from enterprise-centric solutions to personalized adaptive services. In order to proactively cope with the significant changes, the framework has also been evolved from the initial coordinating framework to a context-aware collaboration framework. Furthermore, I hope the target framework of this research can be extended to present context-aware personalized/customized security services or a new type of social networking infrastructure based on a wide variety of contextual information.

APPENDICES

APPENDIX A: THE XCREAM APPLICATION, SPORTS COMPLEX  
MANAGEMENT SYSTEM (SCMS)

In order to assess the feasibility of the XCREAM framework's collaboration capability, the SCMS (Sports Complex Management System) is developed. The SCMS enables multiple 3<sup>rd</sup> party service systems to be interfaced with the framework. It also provides athletes with seamlessly customized services in combination with several application services which offer various services to athletes through their specific solutions.

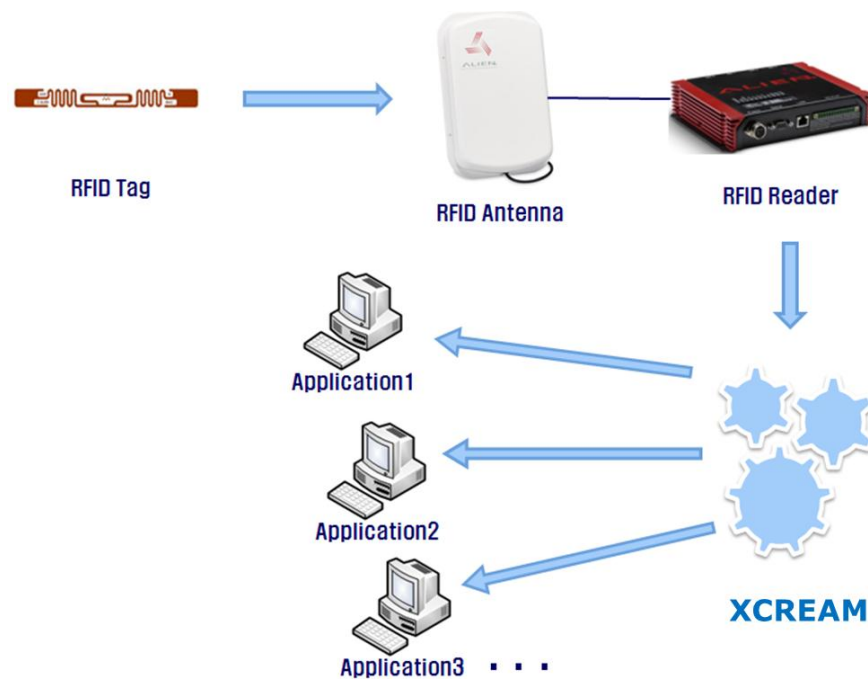
The SCMS is designed to provide a variety of RFID-enabled services within a collaborative service framework. An antenna attached to a RFID reader identifies a RFID tag, and then a RFID reader retrieves tag information from the RFID antenna. The RFID reader sends the tag information to the XCREAM framework. The XCREAM delivers the event to the specific application according to the predefined scenarios.



**Figure 41. The SCMS Configuration**

### (A) The SCMS Components

The SCMS consists of four major business applications: The Registration System (RS); The Restaurant Management System (RMS); Gym Management System (GMS); and Athlete Management System (AMS). The following “Figure 42” shows the event flow of the SCMS.

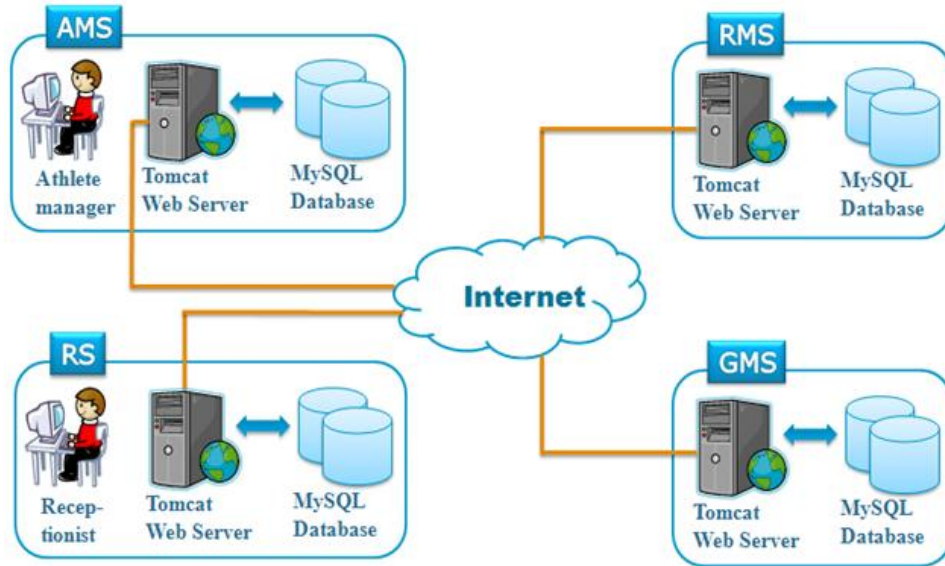


**Figure 42. Event Flow of the SCMS**

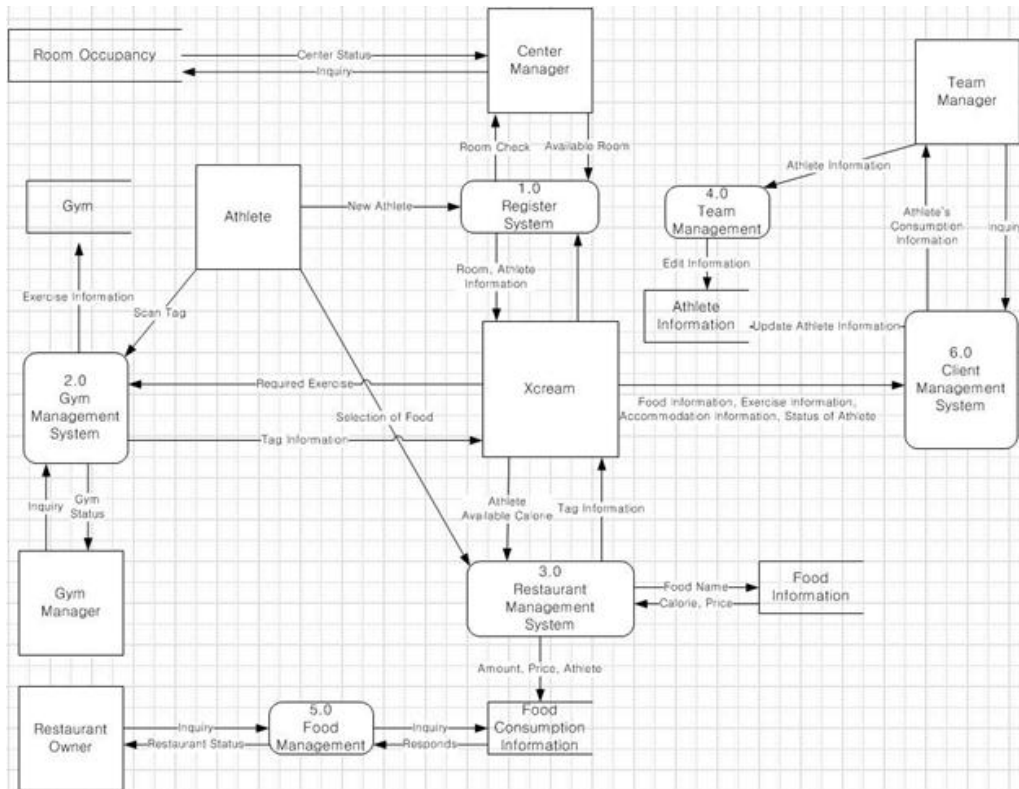
Users are authenticated with a SSN throughout the whole processes. Once authenticated, the user is given the permission to get an RFID tag from the RS and can use various facilities such as the gym, restaurant, pool, and so on. Each facility is controlled by the XCREAM framework and cooperates with other systems by sharing athletes’ identification information. While users are operating any facilities, the related data is kept in the database and immediately transferred to other systems, if needed. The following figures



introduce the network configuration and the data flow diagram (DFD) of the SCMS (see “Figure 43” and “Figure 44”).



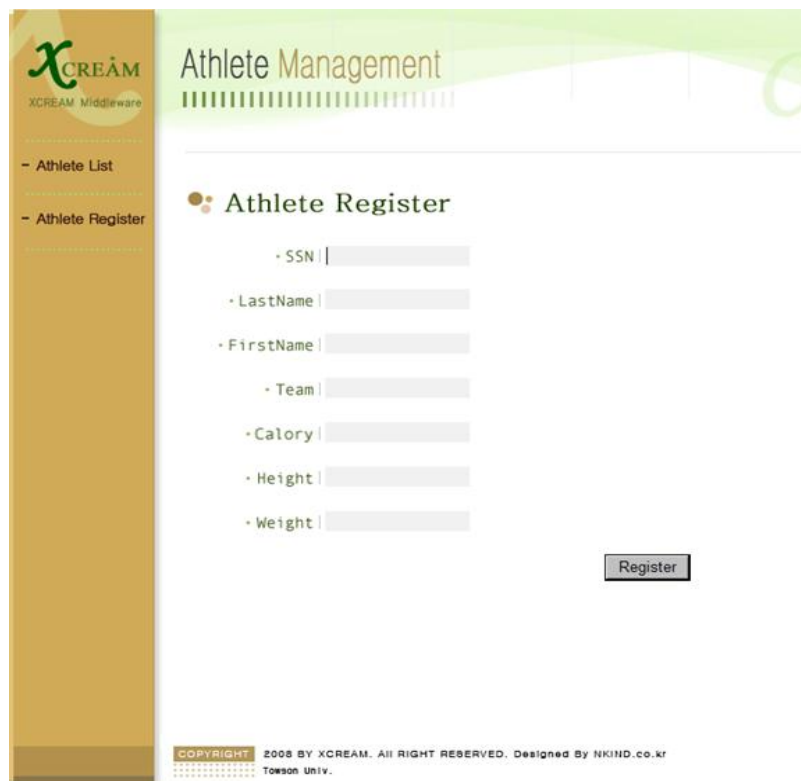
**Figure 43. Network Configuration of the SCMS**



**Figure 44. DFD of the SCMS**

The followings are captured images of the user interface of the SCMS including: (1) Athlete Registration (see “Figure 45”); (2) Athlete Management (see “Figure 46”); (3) Room Management (see “Figure 47”); (4) Gym Management (see “Figure 48”); and (5) Food Management (see “Figure 49”).

### (1) Athlete Registration



The screenshot displays the 'Athlete Management' web interface. On the left is a vertical navigation menu with the XCREAM logo and two options: 'Athlete List' and 'Athlete Register'. The main content area is titled 'Athlete Register' and contains a form with the following fields: SSN, LastName, FirstName, Team, Calory, Height, and Weight. A 'Register' button is located at the bottom right of the form. At the bottom of the page, there is a copyright notice: 'COPYRIGHT 2008 BY XCREAM. All RIGHT RESERVED. Designed By NKIND.co.kr Towson Univ.'

**Figure 45. Athlete Registration**

(2) Athlete Management



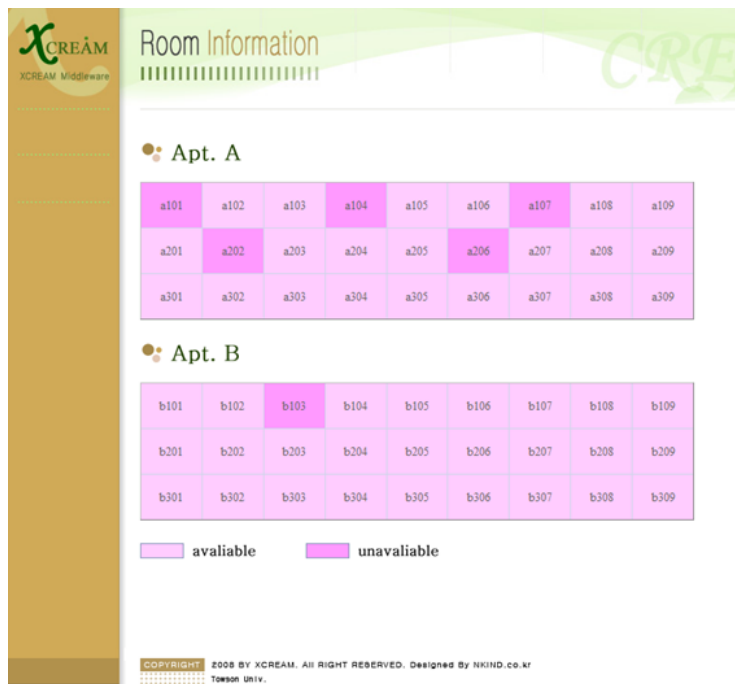
**Athlete List**

Tag	SSN	LastName	FirstName	Team	Calory	height	weight	room	Update	Delete
a001	132-21-1334	Sol	Bee	Singer	1000	170	60	a202	Update	Delete
a002	211-11-3212	Park	TaeHwan	Swim	1400	177	65	a104	Update	Delete
a003	211-11-3213	Park	ChanHo	Baseball	1400	166	65	a107	Update	Delete
a004	211-11-3214	Lee	DaeHo	Baseball	1400	169	70	a101	Update	Delete
a005	211-11-3215	Lee	SeungYup	Baseball	1400	173	81	a206	Update	Delete
a006	211-11-3216	Lee	JongBum	Baseball	1400	175	84		Update	Delete
a007	112-12-3331	Yun	Jesuk	PmgPong	1200	176	74	b103	Update	Delete
a008	211-12-3244	Kim	MinJong	Singer	900	173	50		Update	Delete

COPYRIGHT 2008 BY XCREAM. All RIGHT RESERVED. Designed By NKIND.co.kr Toeson Univ.

Figure 46. Athlete Management

(3) Room Management



**Room Information**

**Apt. A**

a101	a102	a103	a104	a105	a106	a107	a108	a109
a201	a202	a203	a204	a205	a206	a207	a208	a209
a301	a302	a303	a304	a305	a306	a307	a308	a309

**Apt. B**

b101	b102	b103	b104	b105	b106	b107	b108	b109
b201	b202	b203	b204	b205	b206	b207	b208	b209
b301	b302	b303	b304	b305	b306	b307	b308	b309

available      unavailable

COPYRIGHT 2008 BY XCREAM. All RIGHT RESERVED. Designed By NKIND.co.kr Toeson Univ.

Figure 47. Room Management

#### (4) Gym Management

**Gym Management**

**Gym** | THE LIST OF PLAYED EXERCISES LAST TIME (UP TO 5)

NAME	DATE
pool_08	2009-11-04
pool_08	2009-10-31
pool_05	2009-10-31
pool_02	2009-10-31
fitness_05	2009-10-31

**Gym Status**

fitness_01	fitness_02	fitness_03	fitness_04	fitness_05	fitness_06	fitness_07	fitness_08	fitness_09	fitness_10
pool_01	pool_02	pool_03	pool_04	pool_05	pool_06	pool_07	pool_08	pool_09	pool_10

available     unavaliable

Close

**Figure 48. Gym Management**

#### (5) Food Management

**Food Management**

**Requested Calories**

1400 Cal

**Selected Food Information**

Name: Hamberger  
Price: \$8  
Calory: 1100 Cal

Eat    Close

**Figure 49. Food Management**

## **(B) Collaboration of the Application Services**

The following test scenario has been devised to verify the functionalities of the middleware platform, especially for collaboration processes among heterogeneous application services.

1. Player “A” who is the member of the baseball team of “B” has arrived at the complex training center, “XX Corp.” and has been undergoing intense training.
2. “A” goes to the cafeteria in the center for breakfast.
3. When he just enters the restaurant, he is recognized as a member of the “B” team by sensing his RFID-enabled ID card in the Cafeteria Management System.
4. After “A” places his favorite foods on the tray, he just passes the counter in which the RFID antenna has been installed.
  - a. The food expense is calculated and transmitted to the Cafeteria Management System and is charged to the “B” team with the food expense at the end of the month.
  - b. The total calories of the food are calculated and sent to the Player Management System of the “B” team, which keeps track of the individual player’s health record.

The following XLogic script (see “Figure 50”) is presented to describe the above collaboration scenario.

```

<?xml version="1.0" encoding="UTF-8"?>
<xlogic:XLogicScript
  xmlns:xlogic="urn:xcream:xlogic:script:xsd:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:xcream:xlogic:script:xsd:1
  http://triton.towson.edu/xlogic.xsd">

  <!-- Bookmark 1 -->
  <xlogic:set name="tags" select="//tag"> ${_REPORTS}
</xlogic:set>

  <!-- Bookmark 2 -->
  <xlogic:invokeWebService service="PlayerInfoService"
  url="http://baseballteam.bcity.com/pm/baseball"
  name="find_player">
    <rawtags xmlns="bcity">
      <xlogic:iterator name="tag" source="tags">
        <tag>${tag}</tag>
      </xlogic:iterator>
    </rawtags>
  </xlogic:invokeWebService>
  <xlogic:set name="player" select="//id"> ${find_player}
</xlogic:set>

```

**Figure 50. Collaboration Scenario of the SCMS**

```
<!-- Bookmark 3 -->
<xlogic:invokeWebService service="RestaurantPayService"
  url="http://restaurant.xxcorp.com/PayService"
  name="restaurant_pay_result">
  <foods xmlns="xxcorp">
    <xlogic:iterator name="tag" source="tags">
      <food>${tag}</food>
    </xlogic:iterator>
  </foods>
</xlogic:invokeWebService>

<!-- Bookmark 4 -->
<xlogic:invokeWebService service="CalorieService"
  url="http://restaurant.xxcorp.com/CalorieService"
  name="calorie_result">
  <foods xmlns="xxcorp">
    <xlogic:iterator name="tag" source="tags">
      <food>${tag}</food>
    </xlogic:iterator>
  </foods>
</xlogic:invokeWebService>
<xlogic:set name="calorie" select="//calorie">
  ${calorie_result}
</xlogic:set>
```

**Figure 50. Collaboration Scenario of the SCMS (cont.)**

```
<!-- Bookmark 5 -->
<xlogic:invokeWebService service="PlayerManagementService"
  url="http://baseballteam.bcity.com/pm/baseball"
  name="manage_result">
  <updateFoodLog xmlns="bcity">
    <player>${player}</player>
    <calorie>${calorie}</calorie>
  </updateFoodLog>
</xlogic:invokeWebService>
</xlogic:XLogicScript>
```

**Figure 50. Collaboration Scenario of the SCMS (cont.)**

The execution process of the above XLogic script is as follows:

1. Tag data arrives at the Subscriber Agent of the middleware platform.
2. XLogic is executed according to the processing sequences of the trigger mode.
3. The captured tag data are set to variable, called tags (Bookmark 1).
4. From the tags, the player's tag information is set to variable, player (Bookmark 2).
5. The tag ID of the food is transmitted to "RestaurantPayService" of the Cafeteria Management System to accumulate the charges to the whole expenses (Bookmark 3).

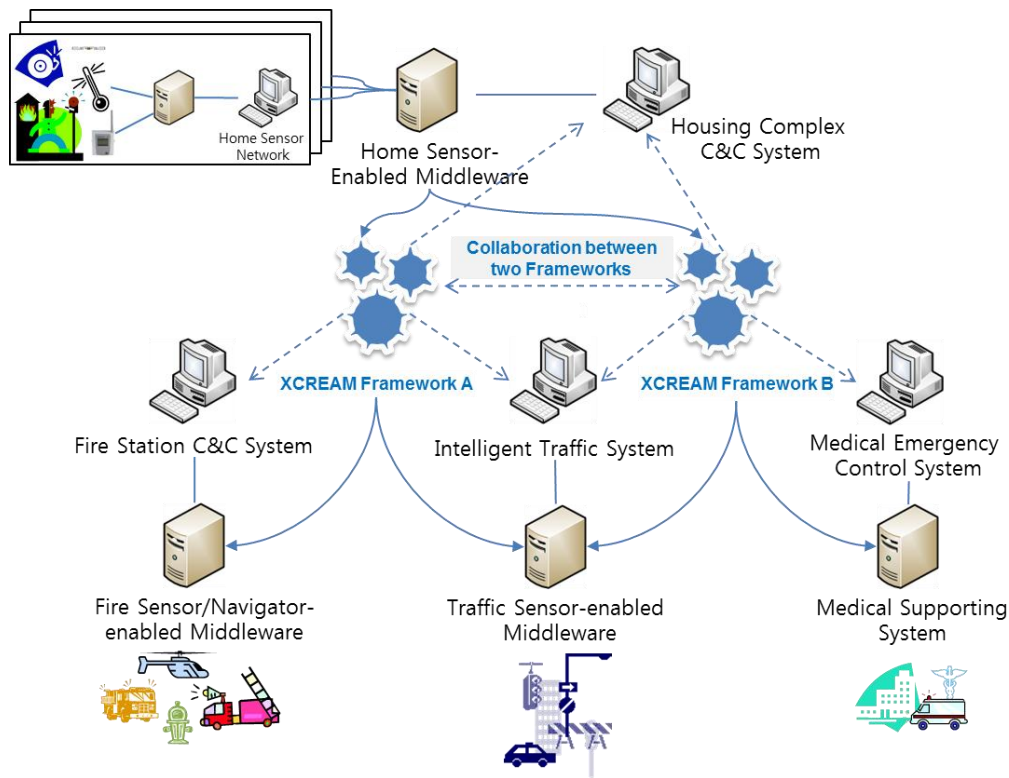


6. The tag ID of the food is transmitted to “CalorieService” of the Cafeteria Management System and all calories are calculated (Bookmark 4).
7. The calculated caloric value is sent to “PlayerManagementService” of the Player Management System to help the baseball team keep track of the nutrition history of the player (Bookmark 5).

In this scenario, the XLogic script, designed to show the collaboration among heterogeneous application services, has been experimented and shared with the external services such as “RestaurantPayService,” “CalorieService,” and “PlayerManagementService” according to the expected execution sequences.

## APPENDIX B: THE EXTENDED COLLABORATION MODEL WITH THE XCREAM FRAMEWORK

The XCREAM framework or similar infrastructure framework are increasingly applied to our environment. As a result, frequently seamless integration between many service frameworks, which previously have been operated independently, is expected.



**Figure 51. The Extended Collaboration Framework**

This section suggests an extended collaboration model with the XCREAM framework by introducing a housing complex where fire sensors and medical sensors are installed and a centralized command and control

system (C&C system) is maintained by a housing company (see “Figure 51”). Usually, a fire station maintains its own C&C system with many fire sensors across its governing region and location identifiable fire engines equipped with GPS (Global Positioning System) and wireless communication facilities. A local traffic agency may operate an Intelligent Traffic System to control the traffic lights on the road. Hospitals of that region have their own medical systems, especially, an emergency rescue system that controls the conditions of emergency rooms and keeps track of an emergency situation.

The growing needs to develop highly collaborative service systems enhance the applicability of the XCREAM framework. Individual systems autonomously work their own business processes and expose some of their functions as service interfaces.

“Figure 51” assumes an extended application environment, in which two frameworks, “A” and “B,” collaborate to serve complex services. The framework is then connected to service specific middlewares such as a fire sensor and navigator-enabled middleware, a traffic sensor-enabled middleware, and a medical sensor-enabled middleware.

The following scenario is supposed for the framework “A”:

1. If a fire alarm starts to beep, the sensor data is delivered to the fire sensor and navigator-enabled middleware.
2. This is propagated to the framework “A” as well as the Fire Station C&C System of the local fire station.

3. The system orders a fire engine to be dispatched to the housing complex. And the route information to the destination is sent to help the Intelligent Traffic System control the traffic lights along the route of the fire engine.

The second scenario describes how the framework “B” works depending on the medical emergency event:

1. When a medical sensor starts to beep, the data is transmitted to the framework “B” through the home sensor-enabled middleware.
2. The framework “B” executes a pre-registered scenario of the Medical Emergency Control System.
3. The location information of the housing complex, delivered from the home sensor enabled middleware, will be recognized with the event.
4. The service scenario for this situation is activated by the framework “B” and the Medical Emergency Control System orders an ambulance to be dispatched.
5. Throughout the emergency rescuing process, the Intelligent Traffic System controls the traffic lights along the route of the ambulance and the Medical Emergency Control System follows up the patients with the continuing sensor data.

## APPENDIX C: XLOGIC SCRIPT LANGUAGE AND ITS EXTENSION FOR XONTOLOGY

The XLogic script is a kind of scenario written by the user. A user can write an XLogic script using the XLogic script language at the XEM. The scenario below is one example of the XLogic script. The following XLogic script is part of a sample service using the XCREAM, finding food information after identifying a RFID tag.

The XLogic languages are similar to conventional programming languages such as Java and C++ and easy for users to write a scenario. The scenario responds when specific tag information comes into the XCREAM.

As the XCREAM is extended to have ontology-based context awareness features, the existing XLogic script system should be extended to support wider concepts and operations.

For example, users may want to handle timestamp of the identified tag data by using time operators or compare a certain predicate with other predicate or any value (literals) by using logical operators

As a result, the basic XLogic script system should be extended by enabling time operators and logical operators: SINCE, UNTIL, BETWEEN a AND b, =, <, >, <=, >=, &&, ^^, and != (see “Table 13”). By adopting these operators, application services become powerful enough to express the ontology-based context aware functions with the XOnt Agent.

“Table 13” summarizes the XLogic script tags including *object*, *location*, *time*, and *XOntScript* as well as the existing tags.

**Table 13. XLogic Script Tags**

Tags	Description	Required Attributes	Optional Attributes
XLogicScript	Root statement of XLogicScript,	xmlns:xlogic, xmlns:xsi, xsi:schemaLocation	name, creationDate
invokeWebService	Invokes web service.	name, url, service	port, namespace, async
iterator	Sets parameter when invoking web-service	name, source	-
set	Sets variable using JEXL.	name	select
if	Controls flow of statement(s).	condition	-
then	Coexists with an <if> tag.	-	-
else	Comes with an <if> Tag.	-	-
while	Iteration statement.	condition	-
foreach	Iteration statement.	name, source	-
wait	Holds process during given milliseconds.	-	-
continue	Continues loop statement.	-	-
break	Escapes loop statement.	-	-
print	Prints JEXL statement.	-	-
XOntScript	Root statement of XOntScript Used to define ontology-based context-aware script	xmlns:xlogic, xmlns:xsi, xsi:schemaLocation	name, creation date
object	Shows information of the identified object	PID, name, idtype	cellno, description, tagno
location	Describes location information	name, address*, latitude and longitude**	address**, latitude and longitude*
time	Returns time information	-	exacttime between ... and ..., before, after, every

The location of an identified person, object, and sensors is specified with an object tag. The location information is considered in two ways: (1) logical location like street address with a floor number, parking lot number, or room number and (2) physical location with latitude and longitude of GPS-based positioning system. Regarding positioning and its user friendly representation, GEO coding or reverse GEO coding should be considered to cover the gap between the logical and physical representations.

## APPENDIX D: XLOGIC SCRIPT LANGUAGE SPECIFICATIONS

The Expression Language for the XLogic Script is JEXL (Java Expression Language). Users follow “xlogic.xsd,” which is XSD (XML Schema Definition) file, in order to write the XLogic script.

### (A) <XLogicScript> Tag

#### (1) Description

When writing XLogic script, the root tag always starts with XLogicScript tag.

The XLogicScript tag can include all other tags.

#### (2) Script example

```
<xlogic:XLogicScript
  xmlns:xlogic="urn:xcream:xlogic:script:xsd:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:xcream:xlogic:script:xsd:1 http://triton.
  towson.edu/xcream/xlogic.xsd" name="" creationDate="">
</xlogic:XLogicScript>
```

#### (3) Attributes

Attributes	Description	Required/Optional
xmlns:xlogic	XLogic namespace	required
xmlns:xsi	xsi namespace	required
xsi:schemaLocation	Value of location of the namespace and the schema	required
name	Name of the script	optional
creationDate	Date of creation of the script	optional



**(B) < invokeWebService > Tag****(1) Description**

Invokes the associated web-service.

**(2) Script specification**

```
<xlogic:invokeWebService name="" url="" service="" port="" namespace=""
  async="">
</xlogic:invokeWebService>
```

**(3) Attributes**

Attributes	Description	Required/Optional
name	Stores result which is transformed to XML	required
url	Location of the web-service	required
service	Name of the web-service	required
port	Port for the web-service	optional
namespace	Namespace of the web-service	optional
async	Indicates whether the synchronization mode is set	optional

**(C) < iterator > Tag****(1) Description**

Sets parameter when invoking web-service. It always follows <invokeWebService> tag.

**(2) Script specification**

```
<xlogic:iterator name="" source=""></xlogic:iterator>
```

**(3) Attributes**

Attributes	Description	Required/Optional
name	Variable name	required
source	XLogic variable name of the script	required

**(D) < set > Tag****(1) Description**

Sets a variable using JEXL. This tag extracts specific value of expression using XPath.

**(2) Script specification**

```
<xlogic:set name="" select="">${}</xlogic:set>
```

**(3) Attributes**

Attributes	Description	Required/Optional
name	Sets name of variable	required
select	XPath of extracted value which is XML type from JEXL	optional

**(E) <if> Tag****(1) Description**

Controls flow of statement(s). It should be followed by a <then> tag. If its condition attribute has been met, then statement(s) following the <then> tag is (are) executed.

**(2) Script specification**

```
<xlogic:if condition="">${}</xlogic:if>
```

**(3) Attribute**

Attributes	Description	Required/Optional
condition	Logical condition of the if statement Returns a boolean value	required

**(F) <then> Tag****(1) Description**

Coexists with an <if> tag. Statement(s) following a <then> tag is (are) only executed when the condition of the <if> tag has been met.

**(2) Script specification**

```
<xlogic:then></xlogic:then>
```

**(3) Attributes**

NA

**(G) <else> Tag****(1) Description**

Comes with an <if> Tag. Statement(s) following an <else> tag is (are) only executed when the condition <if> tag has not been met.

**(2) Script specification**

```
<xlogic:else></xlogic:else>
```

**(3) Attributes**

NA

**(H) <while> Tag****(1) Description**

Iteration statement. It continuously iterates until the condition of JEXL has been met.

### (2) Script specification

```
<xlogic:while condition="$ {}"></xlogic:while>
```

### (3) Attribute

Attributes	Description	Required/Optional
condition	Condition of <while> statement Returns a boolean value	required

## (I) <foreach> Tag

### (1) Description

Iteration statement. It continuously iterates as long as the value of the variable matches a certain value.

### (2) Script specification

```
<xlogic:foreach name="" source=""></xlogic:foreach>
```

### (3) Attribute

Attributes	Description	Required/Optional
name	Name of a variable capturing a value of source	required
source	Value sources	required

## (J) <wait> Tag

### (1) Description

Holds process during given milliseconds. It is requested by using JEXL.

### (2) Script specification

<xlogic:wait>\${ }</xlogic:wait>

**(3) Attributes**

NA

**(K) <continue> Tag**

**(1) Description**

It make the control of a loop continue one more time unconditionally within a <while> statement or a <foreach> statement.

**(2) Script specification**

<xlogic:continue></xlogic:continue>

**(3) Attributes**

NA

**(L) <break> Tag**

**(1) Description**

Escapes loop statement. It completely escapes iteration from <while> statement or <foreach> statement.

**(2) Script specification**

<xlogic:break></xlogic:break>

**(3) Attributes**

NA

**(M) <print> Tag****(1) Description**

Prints JEXL statement.

**(2) Script specification**

```
<xlogic:print>${ }</xlogic:print>
```

**(3) Attributes**

NA

**(N) <XOntScript> Tag****(1) Description**

Root tag of extended XOntScript statement. Used to define ontology-based context-aware script

**(2) Script specification**

```
<xlogic:XOntScript
```

```
  xmlns:xlogic="urn:xcream:xlogic:script:xsd:1"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="urn:xcream:xlogic:script:xsd:1 http://triton.towson.
```

```
  edu/xcream/xlogic.xsd"
```

```
  name="" creationDate="">
```

```
</xlogic:XOntScript>
```

**(3) Attributes**

Attributes	Description	Required/Optional
xmlns:xlogic	XLogic namespace	required
xmlns:xsi	xsi namespace	required
xsi:schemaLocation	Value of location of the namespace and the schema	required
name	Name of the script	optional
creationDate	Date of creation of the script	optional

**(O) <Object> Tag****(1) Description**

Shows information of the identified object.

**(2) Script specification**

```
< xlogic:object PID="" name="" tagtype="" >${ }</ xlogic:location>
```

**(3) Attributes**

Attributes	Description	Required/Optional
PID	Person Identification Number	required
name	Name of object	required
idtype	ID type	required
cellno	Cell phone number	optional
description	Description of an object	optional
tagno	RFID tag number	optional

**(P) <Location> Tag****(1) Description**

Describes location information.

**(2) Script specification**

```
< xlogic:location name="" address="">${}\</ xlogic:location>
```

**(3) Attributes**

Attributes	Description	Required/Optional
name	Name of a location of interest	required
address	Address of a location of interest	required/optional
latitude, longitude	Latitude, Longitude Pair of a location of interest	required/optional

**(Q) <Time> Tag****(1) Description**

Returns time information.

**(2) Script specification**

```
< xlogic:time exacttime="">${}\</ xlogic:time>
```

**(3) Attributes**

Attributes	Description	Required/Optional
exacttime	Notifies a specific time	optional
between ... and ...	Sets time duration	optional
before	Sets a time to apply before operation	optional
after	Sets a time to apply before operation	optional
every	Sets a periodic operation	optional



## REFERENCES

- [1] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist*, Morgan Kaufmann Publishers, Waltham, MA, USA, 2011.
- [2] G. Antoniou, C. V. Damásio, B. Grosz, I. Horrocks, M. Kifer, J. Maluszynski, and P. F. Patel-Schneider, *Combining Rules and Ontologies: A Survey*, Technical Report IST506779/Linköping/I3-D3/D/PU/al, Linköping University, February 2005. IST-2004-506779 REWERSE Deliverable I3-D3, Available from: <http://rewerse.net/deliverables/m12/i3-d3.pdf>.
- [3] G. Antoniou and F. V. Harmelen, “Web Ontology Language: OWL,” *Handbook on Ontologies*, 2004, Available from: <http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf>.
- [4] A. Arasu, S. Badu, and J. Widom, “CQL: A Language for continuous queries over streams and relations,” *DBPL*, 2003.
- [5] F. Baader, I. Horrocks, and U. Sattler, “Description Logics,” In F. V. Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 3, Elsevier, 2008, pp.135-180, Available from: <http://www.cs.ox.ac.uk/ian.horrocks/Publications/download/2007/BaHS07a.pdf>.

- [6] F. Baader and W. Nutt, "Basic Description Logics," In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 2, Cambridge University Press, 2003, pp.47-100, Available from:  
<http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-02.pdf>.
- [7] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," *ACM*, 2002.
- [8] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int'l Journal of Ad Hoc and Ubiquitous Computing*, vol.2, no. 4, 2007.
- [9] S. Bechhofer, I. Horrocks, and P. F. Patel-Schneider, *Tutorial on OWL*, Available from:  
<http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/>.
- [10] P. Borst, "Engineering ontologies," *Int'l Journal of Human-Computer Studies*, vol.46, no.2-3, pp.365-406, February 1997, Available from:  
<http://doc.utwente.nl/18019/1/Borst97engineering.pdf>.
- [11] C. Byun, K. Park, J. Yun, and Y. Kim, "Design and Implementation of the Context-Aware Collaboration Framework with the XCREAM," *Proc. of Int'l Conference on Smart IT Applications (SITA 2011)*, Seoul, Korea, August 2011.

- [12] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, and J.M. Hellerstein, W. Hong, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraohCQ: Continuous dataflow processing for an uncertain world," *CIDR 2003, First Biennial Conf. on Innovative Data Systems Research*, Asilomar, CA, USA, 2003.
- [13] J. Chen, D.J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *Proc. ACM SIGMOD Int'l Conference on Management of data*, 2000, pp.379-390.
- [14] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," *Proc. of the Workshop on the What, Who, Where, When and How of Context-Awareness*, ACM Press, New York, 2006.
- [15] N. Drummond, M. Horridge, and H. Knublauch, "Protégé-OWL Tutorial," *The 8<sup>th</sup> International Protégé Conference*, Madrid, July 2005, Available from:  
[http://protege.stanford.edu/conference/2005/slides/T2\\_OWLTutorialI\\_Drummond\\_final.pdf](http://protege.stanford.edu/conference/2005/slides/T2_OWLTutorialI_Drummond_final.pdf).
- [16] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits, "Reasoning with Rules and Ontologies," *Reasoning Web 2006*, 2006, pp.93-127, Available from: <http://rewerse.net/publications/download/REWERSE-RP-2006-070.pdf>.

- [17] N. D. Evans, "Middleware is the key to RFID," *RFID Journal*, April 2004, Available from:  
<http://www.rfidjournal.com/article/view/858/1/0/>.
- [18] C. Floerkemeie and M. Lampe, "RFID middleware design – addressing application requirements and RFID constraints," *Joint sOc-EUSAI conference*, 2005, France, pp.219-224.
- [19] C. L. Forgy, "Rete: a fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, 19(1), September 1982, pp.17-37.
- [20] M. R. Genesereth, and R. E. Fikes, *Knowledge Interchange Format, Version 3.0 Reference Manual*, Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
- [21] T. M. Ghanem, M. A. Hammad, M. F. Mokbel, W. G. Aref, and A. K. Elmagarmid, "Incremental Evaluation of Sliding-Window Queries over Data Stream," *IEEE Transactions on Knowledge and Data Engineering*, vol.31, no.1, January 2007, pp.57-72.
- [22] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int'l Journal of Human-Computer Studies*, vol.43, November 1995, August 23 1995, pp. 907-928, Available from: <http://tomgruber.org/writing/onto-design.pdf>.
- [23] T. Gruber, *A translation approach to portable ontology specifications*, Knowledge Systems Laboratory Technical Report

- KSL 92-71, Stanford University, CA, USA, 1993, Available from:  
<http://www.dbis.informatik.hu-berlin.de/dbisold/lehre/WS0203/SemWeb/lit/KSL-92-17.pdf>.
- [24] J. Hanson, "Event-driven services in SOA," *JavaWorld*, January 2005, Available from: <http://www.javaworld.com/javaworld/jw-01-2005/jw-0131-soa.html>.
- [25] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo, "Middleware to Support Sensor Network Applications," *IEEE Network Magazine Special Issue*, January 2004.
- [26] I. Horrocks and U. Sattler, *Description Logics*, Tutorial given by at *ECAI-2002*, Lyon, France, July 2002, Available from:  
<http://www.cs.man.ac.uk/~horrocks/Slides/ecai-handout.pdf>.
- [27] D. K. Kim, *Extension of the XCREAM Framework with Ontology-based Context-Aware Scheme*, Master Thesis, Dept. of Computer and Information Sciences, Towson University, MD, USA, 2010.
- [28] G. J. Kim, S. J. Kim, N. S. Kim, and C. S. Pyo, "USN Service and Market Trend," *Journal of KISS*, vol.25, no.12, December 2007.
- [29] M. S. Kim, Y. J. Lee, and J. H. Park, "USN Middleware Technology Trend," *Journal of ETRI*, vol.22, no.3, June 2007.
- [30] Y. B. Kim, C. S. Kim, and J. W. Lee, "A Middleware Platform Based on Multi-Agents for u-Healthcare Services with Sensor

Networks,” *Symposium on Applied Computing Proceedings of the 2008 ACM symposium on Applied computing*, 2008.

- [31] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Scientific American*, May 2001.
- [32] K. S. Leong, M. L. Ng, and D.W. Engels (Auto-ID Labs), *EPC Network Architecture*, AUTOIDLABS-WP-SWNET-012, 2005, Available from: <http://autoidlab.eleceng.adelaide.edu.au/static/EPC%20Network.pdf>.
- [33] S. Li, S. H. Son, and J. A. Stankovic, “Event Detection Services Using Data Service Middleware in Distributed Sensor Networks,” *Information Proc. In Sensor Networks*, April 2003, LNCS 2634, pp.502-517.
- [34] L. Liu, C. Pu, and W. Tang, “Continual Queries for Internet Scale Event-Driven Information Delivery,” *IEEE Tran. on Knowledge and Data Engineering*, vol.11, no.4, 1999.
- [35] T. Liu and M. Martonosi, “Impala: A Middleware System for Managing Autonomic,” *Parallel Sensor Systems, Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, 2003, pp. 107-118.
- [36] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001.

- [37] S. R. Madden, M.J. Franklin, and J.M. Hellerstein, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *ACM TODS*, vol.30, no.1, 2005, pp.122-173.
- [38] P. McCarthy, "Search RDF data with SPARQL," *developWorks of IBM*, May 2005, Available from: <http://www.ibm.com/developerworks/xml/library/j-sparql/>.
- [39] G. Meditskos and N. Bassiliades, "Combining a DL Reasoner and a Rule Engine for Improving Entailment-based OWL Reasoning," *ISWC 2008*, 2008, Available from: <http://lpis.csd.auth.gr/publications/med-iswc08.pdf>.
- [40] J. Moskal and C. Matheus, "Detection of Suspicious Activity Using Different Rule Engines - Comparison of BaseVISor, Jena and Jess Rule Engines," In N. Bassiliades, G. Governatori, A. Paschke, editors, *Rule Representation, Interchange and Reasoning on the Web, Int'l Symposium, RULEML 2008*, FL, USA, October 2008, Proc. vol.5321 of Lecture Notes in Computer Science, pp.73-80, Springer, 2008.
- [41] I. Motakis and C. Zaniolo, "Temporal aggregation in active database rRules," *Proc. of Int'l Conference on Management of Data (SIGMOD)*, ACM Press, 1997.
- [42] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout, "Enabling technology for knowledge sharing,"

*AI Magazine*, 12, 1991, pp.36-56, Available from:

<http://tomgruber.org/writing/AIMag12-03-004.pdf>.

- [43] N. F. Noy, “Semantic Integration: A Survey Of Ontology-Based Approaches,” *ACM SIGMOD Record* vol.33, no.4, December 2004, pp.65-70, Available from: <http://disi.unitn.it/~p2p/RelatedWork/Matching/13.natasha-10.pdf>
- [44] N. F. Noy and S. W. Tu, “Developing Medical Informatics Ontologies with Protégé,” *AMIA (American Medical Informatics Association) Annual Symposium 2003*, 2003, Available from: <http://protege.stanford.edu/amia2003/index.html>.
- [45] M. Obitko, *Introduction to Ontologies and Semantic Web* extracted from Marek Obitko (advisor Vladimir Marik):  
Translations between Ontologies in Multi-Agent Systems, Ph.D. dissertation, Faculty of Electrical Engineering, Czech Technical University in Prague, 2007. Available from:  
<http://www.obitko.com/tutorials/ontologies-semantic-web/introduction.html>.
- [46] J. H. Paik, H. S. Kim, Y. H. Kim, and S. I. Han, “New Technology Trends on Indexing for Moving Objects,” *Journal of KISS*, vol.25, no.1, January 2007.
- [47] K. Park, C. Byun, J. Yun, Y. Kim, and J. Chang, “Context-Aware Inference (CAI) Model on Smart Computing Environment,”



- Proc. of Int'l Conference on Information Science and Applications (ICISA 2012)*, Suwon, Korea, May 2012.
- [48] K. Park, J. Yun, C. Byun, Y. Kim, and J. Chang, "The XCREAM Framework and Collaboration Validity Tests," *Proc. of the 1<sup>st</sup> ACIS/JNU Int'l Conference on Computers, Networks, Systems, and Industrial Engineering (CNSI 2011)*, Jeju, Korea, May 2011.
- [49] K. Park, J. Yun, C. Byun, Y. Kim, and J. Chang, "The XCREAM: Collaborative Middleware Framework for RFID/USN-Enabled Applications," *Proc. of Int'l Conference on Information Integration and Web-based Applications and Services (iiWAS 2010)*, Paris, France, November 2010.
- [50] K. Park, J. Yun, Y. Kim, and J. Chang, "Design and Implementation of Scenario-based Collaborative Framework: XCREAM," *Proc. of Int'l Conference on Information Science and Applications (ICISA 2010)*, Seoul, Korea, April 2010.
- [51] K. Park, Y. Kim, and J. Chang, "Implementation of Collaborative Service Framework with the XCREAM Platform," *Proc. of UKC on Science Technology and Entrepreneurship (UKC 2009)*, Raleigh, NC, USA, July 2009.
- [52] S. Rizvi, S. R. Jeffery, S. Krishnamurthy, M. J. Franklin, N. Burkhart, A. Edakkunni, and L. Liang, "Events on the edge," *Proc. of the 2005 ACM SIGMID Int'l Conference on Management of data*, 2005, pp.885-887.

- [53] N. Ryan, J. Pascoe, and D. Morse, "Enhanced reality fieldwork: the context-aware archaeological assistant," *Proc. of the 25<sup>th</sup> Anniversary Computer Applications in Archaeology*, 1997, Available from: <http://www.caaconference.org/>.
- [54] S. Sarma, D.L. Brock, and K.Ashton, *The networked physical world – proposals for engineering the next generation of computing, commerce & automatic identification*, Technical Report MIT-AUTOID-WH-001, MIT Auto-ID Center, 2000, Available from: <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-001.pdf>.
- [55] B. Schilit and M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol.8, no.5, 1994, pp.22-32.
- [56] R. W. Schulte, "The Growing Role of Events in Enterprise Applications," *Gartner AV-20-3900*, July 2003, Available from: [http://www.gartner.com/DisplayDocument?doc\\_cd=116129](http://www.gartner.com/DisplayDocument?doc_cd=116129).
- [57] C. Sliwa, "Event-driven architecture poised for wide adoption," *Computerworld*, May 2003, Available from: [http://www.computerworld.com/s/article/81133/Event\\_driven\\_architecture\\_poised\\_for\\_wide\\_adoption?taxonomyId=063](http://www.computerworld.com/s/article/81133/Event_driven_architecture_poised_for_wide_adoption?taxonomyId=063).
- [58] J. F. Sowa, "Conceptual Graphs," *Handbook of Knowledge Representation*, 2008, pp.213-237, Available from: [http://www.jfsowa.com/cg/cg\\_hbook.pdf](http://www.jfsowa.com/cg/cg_hbook.pdf).

- [59] V. Spiliopoulos, G. A. Vouros, and V. Karkaletsis, "On the discovery of subsumption relations for the alignment of ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web*, 2010.
- [60] D. Terry, D. Goldberg, D. Nichols, and B. Oki, "Continuous Queries over Append-Only Databases," *Proc. of the ACM SIGMOD Int'l Conference on Management of data*, 1992, pp.321-330.
- [61] M. Wolska and M. Regneri, "Description Logic in a nutshell," Seminar, *Resources for Computational Linguists*, SS 2007, Available from: <http://www.cse.iitd.ernet.in/~kkb/DL-1.pdf>.
- [62] Y. Yao and J. E. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *SIGMD RECORD*, vol.31, no.3, September 2002.
- [63] J. Yun, K. Park, C. Byun, Y. Kim, and J. Chang, "Mobile Real-time Tracking System based on the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware)," *Proc. of the 9<sup>th</sup> ACIS Conference on Software Engineering Research, Management & Applications (SERA 2011)*, Towson, MD, USA, August 2011.
- [64] D. Zimmer and R. Unland, "On the semantics of complex events in active database management systems," *Proc. of Int'l*

*Conference on Data Engineering (ICDE)*, IEEE Computer Society Press, 1999, pp.392-399.

- [65] Conceptual Graphs, *A World of Conceptual Graphs*, Available from: <http://conceptualgraphs.org/>.
- [66] EPCglobal, *GS1 EPC Tag Data Standard*, ver. 1.6, 2011, Available from: [http://www.gs1.org/gsmp/kc/epcglobal/tds/tds\\_1\\_6-RatifiedStd-20110922.pdf](http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf).
- [67] EPCglobal, *The EPCglobal Architecture Framework*, ver.1.4, 2010, Available from: [http://www.gs1.org/gsmp/kc/epcglobal/architecture/architecture\\_1\\_4-framework-20101215.pdf](http://www.gs1.org/gsmp/kc/epcglobal/architecture/architecture_1_4-framework-20101215.pdf).
- [68] EPCglobal, *The Application Level Events (ALE) Specification*, ver.1.1.1, 2009, Available from: [http://www.gs1.org/gsmp/kc/epcglobal/ale/ale\\_1\\_1\\_1-standard-core-20090313.pdf](http://www.gs1.org/gsmp/kc/epcglobal/ale/ale_1_1_1-standard-core-20090313.pdf).
- [69] GE Company, *Semantic Application Design Language (SADL)*, Available from: <http://sadl.sourceforge.net/>.
- [70] Apache Jena, *Apache Jena*, Available from: <http://jena.apache.org/index.html>.
- [71] Apache Jena, *SPARQL Tutorial*, Available from: <http://jena.apache.org/tutorials/sparql.html>.
- [72] The JBoss Drools team, *Drools Introduction and General User Guide*, ver.5.5.0.Beta1 Available from: <http://docs.jboss.org/drools/release/5.5.0.Beta1/droolsjbpm-introduction-docs/pdf/droolsjbpm-introduction-docs.pdf>.

- [73] The JBoss Drools team, *Drools Expert User Guide*, ver.5.5.0.Beta1, Available from: <http://docs.jboss.org/drools/release/5.5.0.Beta1/drools-expert-docs/pdf/drools-expert-docs.pdf>.
- [74] Jess, *the Rule Engine for the Java™ Platform*, Available from: <http://www.jessrules.com/>.
- [75] National Institute of Standards and Technology (NIST), *Special Publication 800-12: An Introduction to Computer Security: The NIST Handbook*, Available from: <http://csrc.nist.gov/publications/nistpubs/800-12/800-12-html/index.html>.
- [76] Oracle, *Oracle Fusion Middleware*, 2011, Available from: [http://docs.oracle.com/cd/E23943\\_01/core.1111/e10103/intro.htm#ASCON110](http://docs.oracle.com/cd/E23943_01/core.1111/e10103/intro.htm#ASCON110).
- [77] The Port Authority of New York and New Jersey, *Airport Rules and Regulations*, August 2009, Available from: [http://www.panynj.gov/airports/pdf/Rules\\_Regs\\_Revision\\_8\\_04\\_09.pdf](http://www.panynj.gov/airports/pdf/Rules_Regs_Revision_8_04_09.pdf).
- [78] Protégé Team (Stanford Center for Biomedical Informatics Research), *Protégé*, Available from: <http://protege.stanford.edu/>.
- [79] Protégé Team, *User Documentation*, Available from: <http://protege.stanford.edu/doc/users.html>.
- [80] RDF Primer, *W3C Recommendation*, February 10 2004,

Available from: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.

- [81] Siemens, *Siemens Airport – Integrated Solutions*, Available from: [http://w3.siemens.com/market-specific/global/en/airports/integrated\\_it\\_solutions/Documents/Integrated\\_Airport\\_Solutions\\_brochure.pdf](http://w3.siemens.com/market-specific/global/en/airports/integrated_it_solutions/Documents/Integrated_Airport_Solutions_brochure.pdf).
- [82] State of Vermont (Agency of Natural Resources), *AIR Pollution Control Regulations*, September 2011, Available from: <http://www.anr.state.vt.us/air/docs/APCR%202011.pdf>.
- [83] The STREAM Group, “STREAM: The Stanford stream data manager,” *IEEE Data Engineering Bulletin*. vol.26, no.1, 2003.
- [84] W3C Recommendation, *OWL 2 Web Ontology Language Primer*, 2009, Available from: <http://www.w3.org/TR/owl2-primer/>.
- [85] W3C Recommendation, *RDF Primer*, February 2004, Available from: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- [86] W3C Recommendation, *RDF Semantics*, February 2004, Available from: <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>.
- [87] W3C Recommendation, *RDF Vocabulary Description Language 1.0: RDF Schema*, February 2004, Available from: <http://www.w3.org/TR/rdf-schema/>.
- [88] W3C School, *Introduction to OWL*, Available from: [http://www.w3schools.com/rdf/rdf\\_owl.asp](http://www.w3schools.com/rdf/rdf_owl.asp).

## CURRICULUM VITAE

## CURRICULUM VITAE

NAME: KYUNG EUN PARK

PERMANENT ADDRESS: 716 LEISTER DR. TIMONIUM, MD 21093

PROGRAM OF STUDY: INFORMATION TECHNOLOGY

DEGREE AND DATE TO BE CONFERRED: DOCTOR OF SCIENCE, 2012

Secondary Education: Master of Science in Computer Science

Seoul National University, Seoul, Republic of Korea, 1992

Collegiate institutions attended	Dates	Degree	Date of Degree
Towson University Maryland, U.S.A.	Aug 2006- Dec 2012	Doctor of Science	December 2012
Seoul National University Republic of Korea	Mar 1990- Feb 1992	Master of Science	February 1992
Seoul National University Republic of Korea	Mar 1986- Feb 1990	Bachelor of Science	February 1992

Major: Information Technology

Research Interest: Multi-Device (RFID/USN) Enabled Middleware Framework,

Complex Event Processing, Location-Based Services,

Geographic Information System, Geospatial DBMS,

Context-Aware System with Rule-Based System and Ontology

## Professional publications:

- [1] K. Park, C. Byun, J. Yun, Y. Kim, and J. Chang, "Context-Aware Inference (CAI) Model on Smart Computing Environment," *Proc. of Int'l Conference on Information Science and Applications (ICISA 2012)*, Suwon, Korea, May 2012.
- [2] C. Byun, K. Park, J. Yun, and Y. Kim, "Design and Implementation of the Context-Aware Collaboration Framework with the XCREAM," *Proc. of Int'l Conference on Smart IT Applications (SITA 2011)*, Seoul, Korea, August 2011.
- [3] J. Yun, K. Park, C. Byun, Y. Kim, and J. Chang, "Mobile Real-time Tracking System based on the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware)," *Proc. of the 9<sup>th</sup> ACIS Conference on Software Engineering Research, Management & Applications (SERA 2011)*, Towson, MD, USA, August 2011.
- [4] K. Park, J. Yun, C. Byun, Y. Kim, and J. Chang, "The XCREAM Framework and Collaboration Validity Tests," *Proc. of the 1<sup>st</sup> ACIS/JNU Int'l Conference on Computers, Networks, Systems, and Industrial Engineering (CNSI 2011)*, Jeju, Korea, May 2011.
- [5] K. Park, J. Yun, C. Byun, Y. Kim, and J. Chang, "The XCREAM: Collaborative Middleware Framework for RFID/USN-Enabled Applications," *Proc. of Int'l Conference on Information Integration and Web-based Applications and Services (iiWAS 2010)*, Paris, France, November 2010.
- [6] K. Park, J. Yun, Y. Kim, and J. Chang, "Design and Implementation of Scenario-based Collaborative Framework: XCREAM," *Proc. of Int'l Conference on Information Science and Applications (ICISA 2010)*, Seoul, Korea, April 2010.
- [7] K. Park, Y. Kim, and J. Chang, "Implementation of Collaborative Service Framework with the XCREAM Platform," *Proc. of UKC on Science Technology and Entrepreneurship (UKC 2009)*, Raleigh, NC, USA, July 2009.
- [8] K. Park, Y. Kim, J. Chang, D. Rhee, and J. Lee, "The Prototype of the Massive Events Streams Service Architecture and its Application," *Proc. of Int'l Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2008)*, Thailand, August 2008.
- [9] K. Park, Y. Kim, J. Lee, and J. Chang, "Integrated Design of Event Stream Service System Architecture (ESSSA)," *Proc. of Int'l Conference on E-business(ICE-B 2007)*, Barcelona, Spain, July 2007.



## Professional positions held:

1999-2006                      Project Manager, Director  
 Korea Telecom Data, Inc.  
 Seoul, South Korea

---

*Managed and Developed*  
 LBS Applications of Spatial Database (ZEUS)

1992-1999                      Researcher, Software Engineer  
 Korea Telecom Research Center  
 Seoul, South Korea

---

*Managed and Developed*  
 Spatial Database (ZEUS) and GIS Applications

## Teaching experience:

2007-2012                      Lecturer  
 Department of Computer and Information Science  
 Towson University  
 Towson, MD

---

*Course developed & taught*  
 Computer and Creativity (COSC 109)  
*Assisted*  
 A Department Faculty

2006                              Teaching Assistant  
 Department of Computer and Information Science  
 Towson University  
 Towson, MD

---

*Assisted*  
 A Department Faculty

## Research experience:

2006-2012                      Doctoral Research: Collaboration Framework for a  
 Context-Aware Environment with Multi-Device  
 Enabled Applications: the XCREAM (XLogic  
 Collaborative RFID/USN-Enabled Adaptive  
 Middleware)

Department of Computer and Information Science  
 Towson University, Towson, MD  
 Advisor: Yanggon Kim Ph.D.

---

This dissertation proposes collaboration framework for a context-aware environment with multi-device enabled applications based on the XCREAM, including CAI model and its applied context-aware mobile security option for smart airport environment.

1990-1992

Master's Research: Kernel Code Analysis tool for Multiprocessor Operating System (MOS)

Department of Computer Science  
 Seoul National University, Seoul, Republic of Korea  
 Advisor: Kern Koh Ph.D.

---

This research proposes Kernel Code Analysis tool for Multiprocessor Operating System (MOS) by scanning the Kernel Code and sorting concurrently processing part out of sequentially processing part. This research had been made by using various Unix kernel utilities.

Skills:

- Proficient in the following languages: C, C++, Pascal, and Java
- Highly experienced in Information Management System, Geospatial Database Management System, and Location Based Services: Structured Query Language, Extended Geospatial Query Language, Rule Language, Ontology Languages including RDF and OWL, etc.
- Extensive experience with Windows platform, also familiar with UNIX environments
- Teaching Multimedia Authoring Software: Flash, Photoshop, Website development tool, Audacity, etc.

Awards:

- Graduate Student Scholarship Award (2006-2012) – Full-time/Half-time Graduate Assistant, Towson University, Maryland
- Jang Youngsil Award (1998) for ZEUS Software Development : KOITA(Korea Industrial Technology Association), Korea

