효과적인 RFID 애플리케이션 개발을 위한 비즈니스 이벤트 프레임워크의 설계 및 구현

유 선 미[†] · 이 찬 영[††] · 문 미 경[†††] · 염 근 혁[††††]

요 약

RFID(Radio Frequency IDentification)는 라디오 신호에 따라 반응하는 태그(Tag)를 이용하여 해당 사물을 인식하는 방법이다. RFID 환경에서는 태그로부터 EPC를 읽어오는데 이 EPC만을 가지고는 해당 물품이 무엇인지만을 알 수 있고 부가적인 상세 정보를 얻기에는 제한적이다. 이 때문에 RFID를 효과적으로 응용하여 사용할 수 있도록 여러 업체나 기관에서 RFID 시스템을 연구하였다. 그러나 여전히 기존의 RFID 시스템은 해당 어플리케이션의 비즈니스 이벤트를 처리하기 위해 RFID에 대한 다양한 지식과 통신 방법 등을 모두 이해하여야 하는 복잡함과 어플리케이션 개발을 지원하는 측면에서 부족함이 있다.

본 논문에서는 RFID 어플리케이션에서 편리하게 이용할 수 있는 비즈니스 이벤트를 정의하였다. 그리고 이러한 비즈니스 이벤트를 생성하여 어플리케이션에 전달할 수 있는 비즈니스 이벤트 프레임워크(Business Event Framework, BEF)를 제시하였다.

키워드 : RFID, 미들웨어, 비즈니스 이벤트 프레임워크, 비즈니스 이벤트

Design and Implementation of Business Event Framework for Developing RFID Applications

Sunmee Yu[†] · Chanyoung Lee[††] · Mikyeong Moon[†††] · Keunhyuk Yeom[††††]

ABSTRACT

Radio Frequency Identification(RFID) is technology that identify products using radio frequency transmission. We read EPC from Tag in RFID environment, but we can see only what the identify objects are and it is too limited to get the additional detailed information by the EPC. Therefore many companies and organizations have studied RFID system in order to effectively use it. But the existing RFID system still has the complication that we should fully understand various knowledges, communicating ways, etc. and the insufficiency to support the application development in order to treat the business event of the related application.

In this paper, I propose business event that we can conveniently use in FRID application and shows Business Event Framework, BEF which is able to deliver the business event produced to the application.

Key Words : RFID, Middleware, Business Event Framework, Business event

1. 서 론

RFID(Radio Frequency IDentification)는 라디오 신호에 따라 반응하는 태그(Tag)를 이용하여 해당 사물을 인식하는 방법이다[1, 2]. 또한 RFID 이벤트의 특징은 짧은 시간 내에 빠르게 생성되며, 생성된 각각의 RFID 이벤트 형태는 단순하지만 이벤트들의 총 데이터 크기는 아주 거대해 진다[3, 4]. 그러므로 이런 특징을 가지고 있는 RFID 이벤트를 처리해 주는 RFID 미들웨어 시스템들이 최근 Sun, IBM, Oracle 등의 여러 Grobal 소프트웨어 업체들에 의해서 개발되고 있다[5, 6, 7].

RFID를 시스템에 적용하기 위해서는 RFID의 특성을 고려하여 시스템을 개발해야 한다. RFID 환경에서는 대량의 태그 데이터가 수집되어 끊임없이 연속적으로 들어오게 되는데 이러한 데이터에는 중복된 데이터나 필요 없는 데이터도 포함된다. 이 때문에 RFID를 이용하는 어플리케이션에서는 RFID 데이터에서 필요한 정보를 분류하고 이것을 실시간으로 효율적으로 처리할 필요가 있다[8].

또한 해당 시스템과의 통신을 위하여 다양한 RFID 프로토콜을 이용해야 한다. 하지만 이런 다양한 시스템을 사용할 수

※ 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/차세대물류IT기술연구사업단).
† 준 회 원 : 부산대학교 대학원 컴퓨터공학과 박사과정
†† 정 회 원 : 대우조선해양 정보기술 R&D팀
††† 정 회 원 : 부산대학교 정보컴퓨터공학부 연구교수
†††† 정 회 원 : 부산대학교 정보컴퓨터공학부 부교수(교신저자)
논문접수 : 2006년 12월 11일, 심사완료 : 2007년 2월 28일

있는 좀 더 편리한 방법이 있다면 RFID 어플리케이션 개발에 도움이 될 것이다.

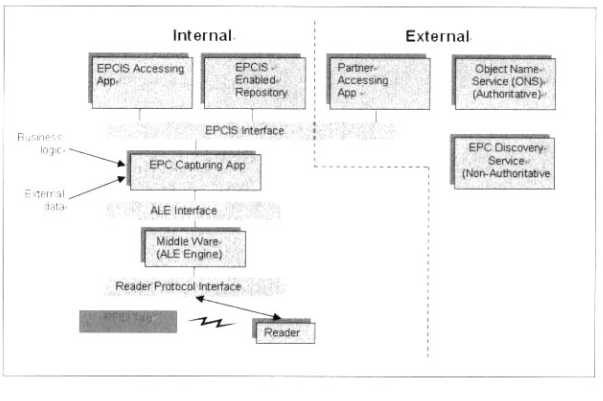
따라서 본 논문에서는 RFID 어플리케이션 개발을 편리하게 할 수 있도록 지원하는 방법을 제시한다. 먼저 기존의 RFID 이벤트와 다르게 비즈니스 로직에 바로 사용될 수 있는 비즈니스 이벤트를 정의한다. 비즈니스 이벤트는 RFID 이벤트와 달리 어플리케이션에서 바로 알 수 있는 의미를 담고 있다. 그리고 비즈니스 이벤트를 생성하여 넘겨줄 수 있는 비즈니스 이벤트 프레임워크(Business Event Framework, BEF)를 제시한다. 이 때 개발자가 원하는 비즈니스 이벤트를 정의하기 위한 방법으로 비즈니스 이벤트 스펙을 제공한다. 개발자는 RFID 어플리케이션을 개발할 때 BEF를 이용하여 비즈니스 로직 수행에 필요한 비즈니스 이벤트를 전달받는다.

본 논문의 전체 흐름은 다음과 같다. 2장에서는 RFID 관련 연구를 살펴보고 3장에서는 비즈니스 이벤트와 BEF에 대하여 알아본다. 4장에서는 비즈니스 이벤트를 정의하는 BESpec(Business Event Specification)을 소개하고 5장에서는 BEF 설계 및 구현에 대하여 소개한다. 마지막으로 6장에서는 결론 및 향후 연구과제에 대하여 제시한다.

2. 관련 연구

RFID의 특성상 범위내의 태그를 모두 읽기 때문에 같은 EPC(Electronic Product Code)정보가 여러 번 읽히거나 원하지 않는 EPC정보가 다른 데이터와 함께 리더를 통해 읽혀질 수 있다. 이러한 RFID의 특성으로 인하여 어플리케이션 개발자는 스무싱(smoothing), 필터링(filtering)과 같은 데이터 가공 방법과 많은 양의 데이터를 실시간으로 효율적으로 처리할 수 있는 방법을 고려해야 한다. 이 외에도 읽혀진 EPC에 대한 제품의 상세정보가 필요하거나 해당 물품의 이동경로에 대한 정보 등이 필요할 수 있는데 이러한 것들을 지원하기 위해서 RFID 시스템 아키텍처에 대한 연구개발이 진행 중이다. 이 중에서 EPCglobal[9]에서 제안한 아키텍처(그림 1)가 업계에서 사실상의 표준으로 간주되고 있다.

그림 1에서 처럼 EPCglobal의 구조에서는 미들웨어로써 ALE(Application Level Events)[10] 엔진이 존재하고 이 미들웨어의 상위에 어플리케이션이 연결된다.



(그림 1) EPC 네트워크 아키텍처

이외에도 제품에 대한 상세한 정보를 제공하는 EPCIS(EPC Information Service)[11], 특정 EPC와 관련된 상세 정보가 저장된 EPCIS의 위치를 알려주는 ONS(Object Name Service)[12] 등의 외부 시스템이 존재하고 어플리케이션은 이러한 시스템과도 통신하여 정보를 주고받는다. 이처럼 EPCglobal의 아키텍처에서는 어플리케이션이 다양한 시스템과 정보를 주고받아야 한다. 이렇게 여러 시스템을 직접 제어하여 어플리케이션을 개발할 수 있지만 이 시스템들을 통합하여 좀 더 편리하게 사용할 수 있도록 지원하는 플랫폼이 있다면 더 편리할 것이다. 이러한 이유로 어플리케이션에서 RFID 이벤트를 편리하고 효과적으로 이용할 수 있도록 지원하는 방법에 대한 연구가 필요하다.

3. 비즈니스 이벤트 프레임워크

3.1 이벤트 비교

• *RFID 이벤트* EPCglobal에서 제시한 구조에서 미들웨어가 어플리케이션에 전달하는 이벤트로써 그림 2와 같이 숫자로 이루어진 EPC와 읽혀진 리더, 시간 등을 포함한다.

이러한 RFID 이벤트는 어플리케이션이 비즈니스 프로세스에 바로 이용하기에는 어려움이 있다. 이 이벤트 중에서 EPC는 단순히 헤더, 업체코드, 상품코드, 일련번호로만 이루어져 있기 때문에 코드와 관련된 상세정보는 알 수 없다. 뿐만 아니라, 해당 EPC 값은 숫자로 되어 있어서 어떠한 정보인지 구체적으로 알기 위해서는 외부 시스템을 이용해야 한다.

• *RFID 비즈니스 이벤트* 본 논문에서는 이전에서 설명한 기존의 RFID 이벤트의 제약적인 한계점을 개선하기 위한 RFID 비즈니스 이벤트(이하 비즈니스 이벤트)를 제안한다. 비즈니스 이벤트는 RFID 이벤트와 달리 비즈니스 로직에 바로 쓰일 수 있는 이벤트를 말한다. 예를 들어 출입구에서 사람의 출입을 인증하기 위하여 RFID를 사용할 때 ALE 엔진에서는 해당 사람이 가진 태그로부터 EPC를 읽어서 이것을 어플리케이션에 전달한다. 하지만 어플리케이션은 이 EPC만을 가지고서는 해당 사람이 인증된 사람인지 아닌지 알 수가 없으므로 인증관련 정보가 저장된 EPCIS에 질문을 해야 한다. 이러한 과정을 거친 후에야 인증된 사람 또는 인증되지 못한 사람에 대한 비즈니스 로직을 수행할 수 있다. 이때 태그로부터 읽은 EPC는 RFID 이벤트이고 이것을 가공한 '인증된 사람이 입구에 나타났다' 또는 '인증되지 않은 사람이 나타났다'라는 이벤트는 비즈니스 이벤트이다. 이러한 비즈니스 이벤트는 어플리케이션이 바로 어떤

```
urn:epc:tag:sgtin-64:4.011562.0557083.19212150
urn:epc:tag:sgtin-64:4.011562.1523083.29356166
urn:epc:tag:sgtin-64:4.011563.4534525.19206803
```

(그림 2) RFID 이벤트 예제

의미인지를 알 수 있고 추가적인 처리 없이도 관련 비즈니스 로직을 수행시킬 수 있다.

비즈니스 이벤트는 이벤트를 전달할 때에 관련된 데이터도 같이 전달하여 어플리케이션이 이 데이터를 이용할 수 있도록 한다. 위의 예제에서는 '인증된 사람이 입구에 나타났다'라는 비즈니스 이벤트와 함께 해당 사람의 이름과 인증등급 등을 같이 어플리케이션에 전달한다.

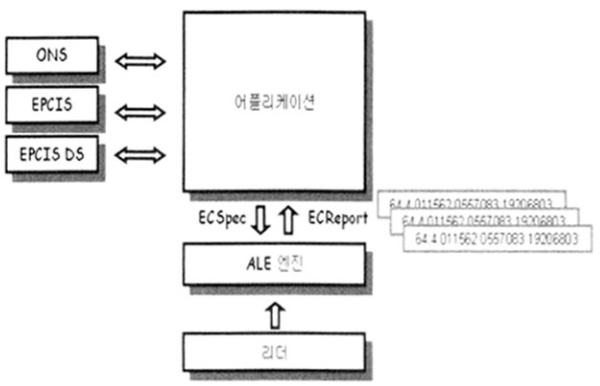
3.2 아키텍처 비교

• *Non-BEF 기반 아키텍처* 그림 3은 기존의 RFID 시스템 아키텍처를 보여준다. 리더는 태그 정보를 읽어서 미들웨어(ALE 엔진)에 전달하고 미들웨어는 이 정보를 필터링과 같은 처리를 한 뒤 어플리케이션이 이용할 수 있도록 전달한다.

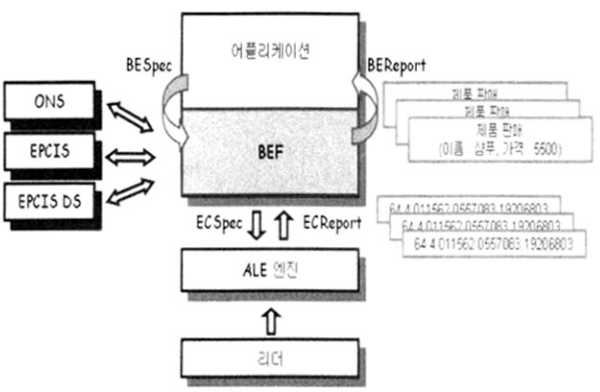
이 때 전달되는 이벤트 형태는 EPC나 RawHex등의 저수준의 RFID 이벤트이다. EPC는 어플리케이션이 비즈니스 로직을 수행할 때 직접적으로 필요한 정보가 아니라 필요한 정보를 얻기 위한 중간 매개체이다.

• *BEF 기반 아키텍처* 본 논문에서 제시하는 하는 것은 그림 4와 같이 어플리케이션과 다른 시스템과의 사이에 BEF가 존재하여 다른 시스템과의 상호작용을 전담하게 된다.

어플리케이션은 이전의 구조(그림 3과 같은 Non-Bef 기반 아키텍처 구조)와 달리 BEF을 이용하여 RFID관련 처리



(그림 3) Non-BEF 기반 아키텍처



(그림 4) BEF 기반 아키텍처

를 할 수 있다. BEF는 어플리케이션이 수행할 작업 중, RFID 이벤트와 관련된 작업을 대신하여 수행한다. 예를 들어, 리더로부터 데이터를 수집하기 위하여 ECSpec을 생성하여 ALE 엔진에 전달하거나, 수신된 ECReport를 해석하여 데이터에 따른 처리를 하는 등의 작업을 대신 수행한다. 또한 외부 지원 시스템인 EPCIS, ONS, EPCIS DS 등과도 해당 시스템이 제공하는 인터페이스와 프로토콜에 따라 통신하여 필요한 정보를 받아오기도 한다. BEF는 이러한 여러 처리 과정들을 거쳐 어플리케이션이 원하는 비즈니스 이벤트를 생성하여 전달한다. 따라서 어플리케이션 개발자가 BEF를 사용하기 위해서는 자신이 원하는 비즈니스 이벤트를 정의할 수 있어야 한다. BEF에서는 비즈니스 이벤트 등록을 위하여 BESpec(Business Event Specification)을 제공한다. BESpec은 XML기반의 비즈니스 이벤트 정의 언어로써 개발자는 이 언어를 이용해 어떤 리더로부터 RFID 이벤트를 받을지, 언제 RFID 이벤트를 받을지, 또는 이 수신된 RFID 이벤트를 이용하여 어떤 처리과정을 거칠지 등을 정의할 수 있다. 개발자는 자신이 필요한 비즈니스 이벤트 종류마다 BESpec을 정의하여 프로그램 실행 시에 BEF에 전달한다. BEF는 정의된 BESpec에 따라서 외부 시스템과의 상호작용을 하거나 내부 처리과정을 거쳐서 비즈니스 이벤트를 생성하고 생성된 비즈니스 이벤트를 어플리케이션에 전달한다.

4. 비즈니스 이벤트 명세

본 논문에서는 다음의 비즈니스 이벤트 명세를 제안함으로 비즈니스 이벤트를 기반한 아키텍처를 구현하고 처리하기 위한 스펙을 제안하였다.

• *BESpec* 크게 두 부분으로 구성되는데 변수 선언 부분과 여러 처리 과정을 정의하는 액티비티 부분이다. 변수 선언 부분에서는 비즈니스 이벤트 관련 데이터 전달이나 중간 처리과정에서 사용되는 변수를 선언한다. 변수에는 일반적인 타입 이외에 RFID 고유의 타입이 지원된다. 액티비티 부분에서는 미리 정의된 여러 액티비티를 비즈니스 이벤트 생성에 필요한 순서에 따라 정의한다.

• *변수* 태그정보를 읽었을 때는 BESpec의 정의에 따라 여러 처리과정을 거치게 된다. 각각의 처리과정은 독립적으로 수행되는데 각 처리과정 사이에 필요한 정보를 넘겨주기 위한 매개체로서 변수를 사용한다. 각 처리과정에서는 처리결과를 변수에 저장하거나 입력으로 변수를 사용한다. 변수는 int 등의 일반적인 타입뿐만 아니라 RFID 시스템에 특화된 타입도 지원하여 BESpec의 사용 편이성을 높인다. 예를 들면,ALE 엔진에서 한 RIFD 이벤트 사이클 동안 넘겨주는 태그 목록들을 한 번에 저장할 수 있는 태그리스트 같은 데이터 타입을 제공하여 BESpec 작성 시 편의성과 가독성을 높일 수 있다. 변수를 선언하기 위해 사용되는 DTD 문법은 그림 5와 같다.

DTD 문법과 같이 변수 선언 시작은 variables로 시작되고 이 태그의 자식부분에 여러 변수를 선언하게 된다. 각 변수는 variable태그를 이용하여 선언하고 variable의 #PCDATA 부

분이 변수의 이름을 나타낸다. 변수의 타입이나 리스트 타입여부, 초기 값은 속성정보를 이용하여 표현한다. 이 중 타입은 항상 정해줘야 하고 리스트 속성과 초기 값은 명시적으로 표현하거나 생략이 가능한데 생략했을 때에는 리스트 타입이 아닌 일반 타입이 된다. 그림 6은 EPC 목록을 저장할 EPC형 리스트 변수와 int형 변수를 선언한 실제 예제이다.

• *액티비티* 비즈니스 이벤트를 생성하기 위해서는 ALE에서 리더로부터 읽은 정보를 가져오고 EPCIS나 ONS와도 정보를 주고받아야 한다. 이러한 행동들은 서로 간섭 없이 독립적으로 수행할 수 있다. 이러한 독립적인 처리에 대한 정의를 할 수 있도록 액티비티(Activity)를 정의하였다. 액티비티는 기본적으로 ALE, EPCIS나 ONS 등과의 상호작용에 대한 정의를 하는 것이다. 외부 시스템과의 상호작용에 대한 정의 외에도 내부적인 처리 및 비즈니스 이벤트 전달에 대한 정의를 하기 위한 추가적인 액티비티가 있다. 이러한 액티비티를 일련의 처리 순서대로 배치하여 비즈니스 이벤트를 생성한다. 액티비티의 종류는 표 1과 같다.

액티비티는 종류에 따라서 다른 액티비티를 내부에 가질 수 있는 액티비티(Contain Activity)와 가질 수 없는 액티비티(Primitive Activity)로 구분된다.

그림 7은 각 액티비티간의 관계를 보여준다.

```
<!ELEMENT variables (variable)*>
  <!ELEMENT variable (#PCDATA)>
  <!ATTLIST variable type (int|string|EPC|Tag|RawHex|RawDecimal) #REQUIRED>
  <!ATTLIST variable list (true|false) #IMPLIED>
  <!ATTLIST variable initValue CDATA #IMPLIED>
```

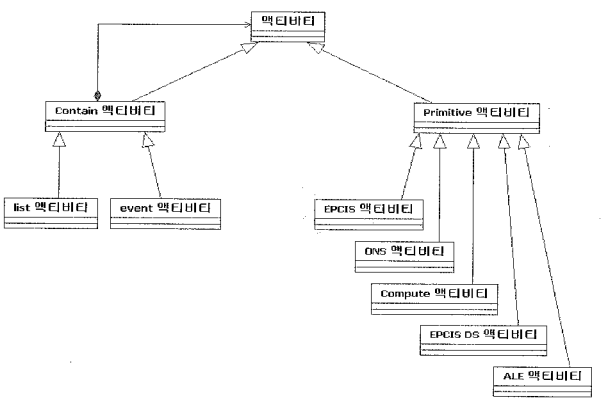
(그림 5) 변수 선언 DTD

```
<variables>
  <variable type="EPC" list="true">vEPCList</variable>
  <variable type="int" initValue="0">vInt</variable>
</variables>
```

(그림 6) 변수 선언 예제

〈표 1〉 BEF 액티비티 종류

액티비티 이름	설명
ALE	ALE로부터 태그에 대한 정보를 획득
EPCIS	EPC에 대한 상세정보를 획득
ONS	EPC에 대한 상세정보를 저장한 EPCIS의 주소 획득
event	조건에 따라 비즈니스 이벤트 생성
list	list 타입에 대한 반복 처리
compute	산술 연산 처리
EPCIS DS	EPC의 이동에 대한 상세정보를 획득



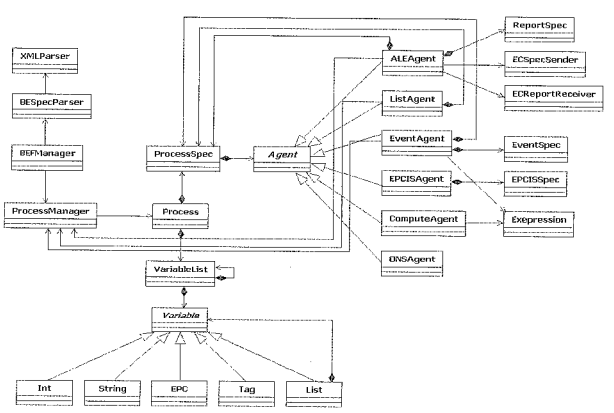
(그림 7) 액티비티 간의 관계

5. BEF 설계 및 구현

• *요구사항* BEF는 어플리케이션 개발 시에 이용되는 프레임워크인데 어플리케이션은 자신이 필요한 비즈니스 이벤트에 대하여 XML 기반의 BESpec을 작성하여 BEF에 알려준다. 이러한 특성으로 인하여 외부 인터페이스는 BESpec로드, 시작, 중지, 이벤트 전달의 몇 가지만 필요하므로 BESpec을 효율적으로 처리할 수 있는 구조로 설계한다.

• *BEF 아키텍처 설계* 요구사항을 바탕으로 변수와 액티비티, 프로세스를 효율적으로 처리할 수 있도록 설계한다. 그리고 어플리케이션이 사용할 외부 인터페이스를 제공하는 BEFManager 클래스를 정의하고 BESpec 분석 모듈을 추가한다. 그리고 각 에이전트별로 필요한 정보를 담고 있는 클래스를 추가한다. 그림 8은 BEF의 전체 구조이다. 프로그램은 메인은 BEFManager이고 BESpec로드 명령에 따라 XML로 작성된 BESpec은 BESpecParser모듈을 이용해 분석되어 읽혀지게 된다. 이후 각 BESpec에 따라서 독립적인 프로세스를 생성하여 처리하게 되고 이 프로세스들은 ProcessManager가 관리하게 된다. 각 프로세스에는 변수 목록과 프로세스 스펙에 대한 정보를 담고 있는데 이것은 읽혀진 BESpec에 따라서 동적으로 구성되어 실행 시간에 설정된 내용에 따라 작업을 수행하게 된다.

• *BEF 구현* BEF의 구현은 Visual C++ 6.0을 이용하여 구현한다. BEF는 어플리케이션 개발을 지원하는 프레임워크이므로 DLL(Dynamic Link Library)형태로 제공하여 어플리케이션 개발 시에 쉽게 이용가능하게 한다. DLL에는 BEFManager라는 외부 클래스가 있어서 BEF를 이용하는



(그림 8) BEF 구조

어플리케이션은 이 클래스를 이용하여 자신이 필요한 비즈니스 이벤트를 등록할 수 있다. BEF로부터 비즈니스 이벤트를 수신할 때는 복잡한 통신 프로토콜을 사용하지 않아도 비즈니스 이벤트를 받을 수 있다.

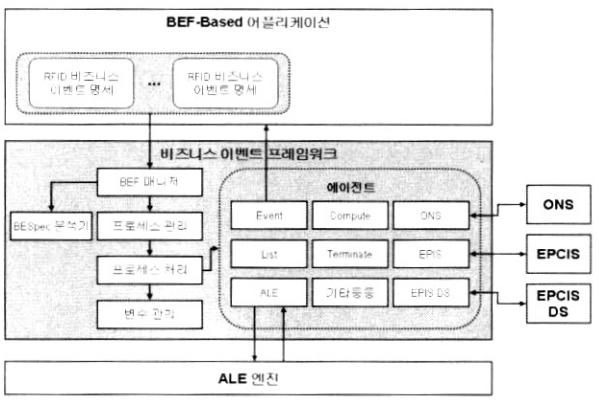
내부 클래스들은 설계에 따라서 구현을 하고 외부 시스템과의 통신을 위해서는 별도의 추가 프로그램을 구현한다. ALE와의 통신 프로토콜은 자바 웹 서비스를 이용하고 EPCIS와의 통신에는 자바 RMI를 이용한다. 이러한 프로토콜은 C++과 직접적으로 통신이 힘들기 때문에 중간에 연결을 위한 자바 프로그램을 작성한다. 중간 자바 프로그램은 외부 시스템이 제공하는 프로토콜에 따라서 해당 시스템과 통신을 하고 그 결과를 BEF에 전달할 때는 소켓통신을 이용하여 전달한다. 향후 외부 시스템이 지원하는 프로토콜을 확장하면 해당 프로토콜을 지원하도록 구조가 수정될 수 있다. 그림 9는 최종적으로 구현된 BEF의 개념도이다.

또한 BEF는 Running시에 서버형태로 동작한다. BEF의 동작은 RFID 리더에 RFID 태그가 인식되면 ALE는 그 태그의 EPC 코드 값을 XML 리포트 형태로 전달한다. BEF는 ALE로부터 수신대기 상태에서 이벤트가 수신될 경우 EPC 이벤트를 수신한다. 수신 받은 EPC 이벤트는 ONS, EPCIS를 통해 상품 정보에 대한 참조 데이터를 얻어서 상위 애플리케이션에 비즈니스 이벤트로 가공되어 전달된다. 이때 BEF는 ALE의 Event 주기마다 각 비즈니스의 상황에 따른 조건을 기술하는 BESpec을 작성하여 등록한다. BESpec은 그림 10과 같이 XML 에 기반을 두고 있다.

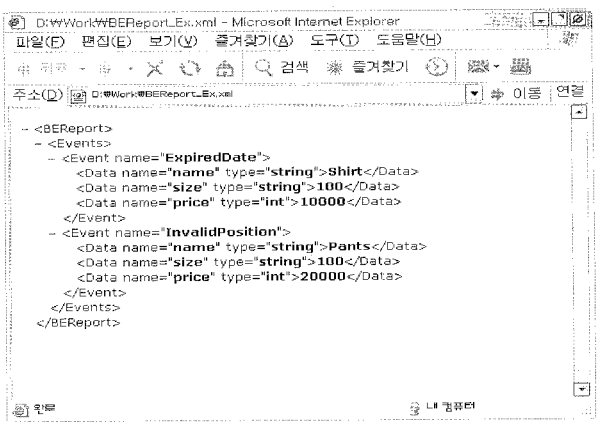
BESpec의 <event>요소는 <condition>에 기술되어 있는 조건에 따라 BEReport(Business Event Report)형태로 발생할 수 있도록 기술하는 요소이다. EPC 이벤트가 올라오게 되면 그림 11과 같이 비즈니스 이벤트를 XML 형태의 BEReport 형태로 가공하여 상위의 애플리케이션으로 전달한다.

6. 결론 및 향후 연구

본 연구에서는 RFID 어플리케이션을 편리하게 개발할 수 있도록 지원하는 BEF에 대하여 제시하였다. EPCglobal에서 제시한 기존의 RFID 시스템 아키텍처에서는 ALE 엔진, EPCIS, ONS 등의 외부 시스템들이 존재하고 어플리케이션을 개발할 때 이러한 외부 시스템의 기능을 이용한다. 이것은 어플리케이션 개발자가 습득해야 하는 지식을 증가시키고 개발을 어렵게 한다. 이러한 어려움을 줄이고자 본 논문



(그림 9) BEF 개념도



(그림 11) BEF로부터 리포팅되는 XML형태의 BEReport

(그림 10) BESpec 예제

에서는 비즈니스 로직을 바로 수행 시킬 수 있는 비즈니스 이벤트를 정의하였다. 그리고 이러한 비즈니스 이벤트를 생성하여 어플리케이션에 전달할 수 있는 BEF를 제시하였다.

향후 연구 과제로는 BESpec의 단순화와 기능의 확장이다. 현재는 외부 시스템의 인터페이스 변화에 따른 영향이 큰데 여러 외부 시스템의 기능을 추상화하여 여러 시스템에서 같은 형태로 쓰일 수 있도록 할 필요가 있다.

참 고 문 헌

[1] *A Basic Introduction to RFID technology and Its use in the supply chain*, http://www.printronix.com/uploadedFiles/Laran_WhitePaper_RFID.pdf, January 2004.

[2] H.K. *Launches RFID Supply Chain Project*, RFID journal, http://www.rfidjournal.com/article/articleview/1630/1/1/, June 2005.

[3] DoD, *Final Regulatory Flexibility Analysis of Passive Radio Frequency Identification*, http://www.acq.osd.mil/log/rfid/EA_08_02_05_UnHighlighted_Changes.pdf, August 2005.

[4] Teresko J., "Winning with Wireless", Indestry Week, 252(6), www.industryweek.com/CurrentArticles/Asp/articles.asp?ArticleId=1434, June 2003.

[5] Oracle, *Oracle Sensor Edge Server Developers Guide*, http://www.oracle.com/technology/products/sensor_edge_server/SES%2010.1.2%20Documentation.zip, December 2004.

[6] Sun Microsystems, *The Sun Java System RFID Software Architecture*, http://www.sun.com/software/solutions/rfid/EPCNetArch_wp.pdf, March 2005.

[7] IBM, *RFID Premises Server*, http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/, December 2004.

[8] Palmer M., "Seven Principles of Effective RFID Data Management", http://www.objectstore.com/docs/articles/7principles_rifd_mgmnt.pdf, August 2004.

[9] EPCglobal, *The Savant Version 0.1(Alpha) Technical manual*, February 2002.

[10] EPCglobal, *The Application Level Events (ALE) Specification Version 1.0*, September 2005.

[11] Mark Harrison, "EPC Information Service", January 2004.

[12] EPCglobal, EPCglobal Object Name Service (ONS) 1.0 Working Draft Version, November 2004.

유 선 미

e-mail : smyou@pusan.ac.kr
1995년2월 연세대학교 전자계산학과 (학사)
2006년2월 부산대학교 대학원 컴퓨터공학과(공학석사)
2006년3월 ~ 현재 부산대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 상황인식 미들웨어, RFID기반 비즈니스 프레임워크, 서비스기반 비즈니스 프로세스 등

이 찬 영

e-mail : young78@dsme.co.kr
2004년 2월 부산대학교 컴퓨터공학과 (학사)
2006년 2월 부산대학교 대학원 컴퓨터공학과(공학석사)
2006년 ~ 현재 대우조선해양 정보기술 R&D팀
관심분야 : 소프트웨어공학, 캐드 시스템 등

문 미 경

e-mail : mkmoon@pusan.ac.kr
1990년 2월 이화여자대학교 전자계산학과 (학사)
1992년 2월 이화여자대학교 전자계산학과(이학석사)
2005년 2월 부산대학교 컴퓨터공학과(공학박사)
2005년 3월 ~ 2005년8월 부산대학교 차세대물류IT기술사업단 박사후 연구원
2005년 9월 ~ 2006년8월 부산대학교 컴퓨터 및 정보통신 연구소 기금교수
2006년 9월 ~ 현재 부산대학교 정보컴퓨터공학부 연구교수
관심분야 : 소프트웨어 프로덕트 라인 공학, 적응형 소프트웨어 개발, RFID 미들웨어 개발 및 RFID 기반 애플리케이션 개발 등

염 근 혁

e-mail : yeom@pusan.ac.kr
1985년 2월 서울대학교 계산통계학과(학사)
1992년 8월 Univ. of Florida 컴퓨터공학과(공학석사)
1995년 8월 Univ. of Florida 컴퓨터공학과(공학박사)
1985년 1월 ~ 1988년 2월 금성반도체 컴퓨터연구실 연구원
1988년 3월 ~ 1990년 6월 금성사 정보기기연구소 주임연구원
1995년 9월 ~ 1996년 8월 삼성SDS 정보기술연구소 책임연구원
1996년 8월 ~ 현재 부산대학교 정보컴퓨터공학부 부교수
부산대학교 컴퓨터 및 정보통신연구소 연구원
관심분야 : 소프트웨어 재사용, 프로덕트라인 공학, 서비스기반 비즈니스 프로세스, 상황인식 미들웨어, 적응형 소프트웨어 개발 방법 등