WS-ECA 프레임워크 설계 및 구현

이원석* · 신동민** · 이규철***

Design and Implementation of WS-ECA Framework

Lee Won-Suk* · Shin Dong-Min** · Lee Kyu-Chul***

요 약

유비쿼터스 환경으로 발전하면서 이종의 디바이스들을 자연스럽게 연동시킬 수 있는 기술에 대한 요구가 크게 증가하고 있으며, 이를 해결하기 위한 하나의 대안으로 웹서비스가 주목 받고 있다. 최근 디바이스 연동을 선언적인 ECA(Event-Condition-Action) 룰로 정의하고 이를 이용하여 디바이스 연동을 수행할 수 있는 웹서비스 기반의 WS-ECA(Web Services-ECA) 프레임워크가 제안 되었다. 본 논문에서는 WS-ECA 프레임워크에 대한 구체적인 모듈 단위의 설계 부분을 제안하고, 이를 기반으로 WS-ECA 프레임워크를 구현하였다.

ABSTRACT

Requirement of coordination among heterogeneous devices is significantly increasing to meet the challenges presented by the evolvement of the ubiquitous computing environment. The web services technology is receiving much attention as a promising solution to support the inter-operability between such heterogeneous devices by means of well-defined protocol. Recenlty, ECA(Event-Condition-Action) rule-based framework for coordinating devices is proposed, which is called WS-ECA(Web Services-ECA) framework. In this paper we propose a specific design and an implementation methodology for WS-ECA framework.

키워드

WS-ECA 프레임워크, 디바이스 연동, 프레임워크 구현, 웹서비스

I. 서 론

최근 컴퓨팅 환경이 수많은 디바이스가 긴밀하게 네트워크로 연결되는 유비쿼터스 환경으로 빠르게 발전하고 있으며, 이러한 환경은 다양한 응용, 프로토콜, 포맷 등이 상존하는 이종의 시스템 환경을 포함한다[1]. 이와 같은 유비쿼터스 컴퓨팅 환경에서 웹서비스 기술은 이종의 하드웨어와 소프트웨어 플랫폼을 가지고 있는 다양한 디바이스들이 상호운용성을 보장하면서 자연스럽게 연동할 수 있는 효율적인 수단을 제공할 수 있다.

이와 같이 웹서비스를 기반으로 디바이스 간의 연동을 수행하기 위한 주요 연구들을 보면, MS의 유비쿼터스 컴퓨팅 프로젝트의 일환으로 초소형 디바이스에 웹서비스 기술을 내장하는 것을 목표로 수행한 Invisible computing 프로젝트[2]와 Trento 대학의 웹서비스를 기반으로 가정의 모든 디바이스(센서, 가전 등) 간의 효과적인 연동 방법을 연구한 스마트 홈 프로젝트[3]가 있다. 또한, 홈네트워킹의 미들웨어 표준을 개발하고 있는

* 한국전자통신연구원 표준연구센터
** 한양대학교 정보경영공학과
*** 충남대학교 컴퓨터공학과

접수일자 : 2007. 6. 26

UPnP 포럼에서는 UPnP 2.0에 웹서비스 기반의 디바이스 연동 기술에 대한 내용을 준비하고 있다.

웹서비스를 기반으로 보다 쉽고, 정확하게 디바이스 연동을 수행하기 위해서는 비즈니스 분야의 WS-BPEL (Web Services Business Process Execution Language)[4]과 WS-CDL(Web Services Choreography Description Language)[5]과 같은 서비스 연동을 기술하기 위한 선언적인 언어에 대한 명세가 필요하다. 이러한 명세는 웹서비스를 기반으로 외부의 다양한 사업 파트너들과 내부의 시스템간의 효율적인 통합을 가능하게 한다. 그러나, 기존에 정의된 WS-BPEL과 WS-CDL은 기본적으로 유비쿼터스 환경의 동적인 특성을 고려하지 않았고, 스펙이 방대한 양의 기능을 정의하고 있어 이러한 기능을 유비쿼터스 환경의 소형 디바이스에 적용하는 것은 적합하지 않다.

WS-ECA 프레임워크[6, 7]는 유비쿼터스 환경의 동적인 특성을 지원하고, 소형 디바이스에 적용 가능한 XML 기반의 WS-ECA 룰 기술 언어 및 이를 실행하기 위한 프레임워크를 제안하였다. 제안된 WS-ECA 룰 기술 언어는 데이터베이스 분야에서 기존의 데이터베이스에 이벤트를 기반의 동적인 처리 메커니즘을 제공하기 위해서 고안된 ECA(Event-Condition-Action) 룰을 기반으로 한다. ECA 개념은 현재 다양한 분야에서 분산된 환경에서 가장 효율적인 연동 방법 중의 하나로 인식되고 있으며[8], 유비쿼터스 환경에서 특정 이벤트에 대한 반응을 명확히 기술할 수 있는 장점이 있다.

본 논문에서는 제안된 WS-ECA 프레임워크의 특성을 만족시킬 수 있는 구체적인 모듈의 설계 및 이를 기반으로 한 WS-ECA 프레임워크의 구현 내용을 소개한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안된 WS-ECA 프레임워크를 소개하고, 3장에서는 WS-ECA 프레임워크에 대한 구체적인 모듈 및 이들 간의 관계에 대한 설계 내용을 설명한다. 4장에서는 WS-ECA 프레임워크에 대한 구현 내용을 설명하고, 마지막으로 5장에서 결론을 맺는다.

II. WS-ECA 프레임워크

기본적으로 WS-ECA 프레임워크는 WS-Eventing의 publish/subscribe 메커니즘과 웹서비스 기반의 request/response 메커니즘을 기반으로 하다. 그리고, 디바이스 연동을 정의하는 선언적인 룰을 통해 디바이스 간의 명확한 연동을 가능하게 한다.

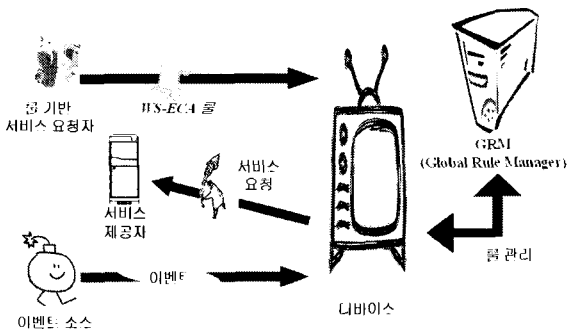


그림 1. WS-ECA 프레임워크
Fig. 1 WS-ECA framework

그림 1은 WS-ECA 프레임워크를 보여주며, 이는 기본적으로 룰 기반 서비스 요청자, 디바이스, GRM (Global rule manager), 이벤트 소스, 서비스 제공자로 구성이 된다. 처음에 룰 기반 서비스 요청자는 자신이 원하는 서비스에 대한 디바이스 간의 연동을 WS-ECA 룰을 이용하여 정의하고 이를 디바이스에 저장한다. 디바이스는 룰을 저장하라는 요청이 오면, 기존에 저장되어 있는 룰과의 충돌 문제가 없는지 GRM에게 확인 후 문제가 없으면 저장한다. 디바이스에 저장된 WS-ECA 룰은 이벤트를 기반으로 하기 때문에 특정 이벤트가 발생하기를 기다린다. 디바이스가 이벤트 소스에 의해 이벤트를 받게 되면 수행할 대상이 되는 룰을 찾아 조건을 확인한 후 정의된 액션을 수행하게 된다. 이때, 액션은 룰 기반 서비스 요청자가 정의한 서비스 시나리오에 따라 다음 단계의 수행을 위한 내부나 외부의 새로운 이벤트를 생성시킬 수도 있으며, 웹서비스 제공자가 제공하는 서비스를 호출할 수도 있다. 본 프레임워크에서 모든 디바이스는 웹서비스 제공자가 될 수 있으며, 인터넷에 존재하는 웹서비스도 연동의 대상이 된다. 또한, 이벤트 소스도 모든 디바이스가 될 수 있으며, WS-Eventing 표준을 따르는 인터넷의 서비스도 연동의 대상이 된다.

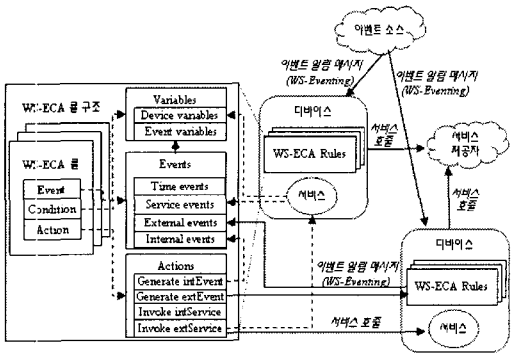


그림 2. WS-ECA 룰 구조 및 구성요소 연동 구조
Fig. 2 WS-ECA rule structure and coordination architecture among components

그림 2는 WS-ECA 룰의 구조와 이를 기반으로 연동되는 디바이스, 이벤트 소스, 서비스 제공자의 전체적인 관계를 보여준다. WS-ECA 룰 구조는 이벤트(event), 조건(condition), 액션(action)으로 구성되며, 여기서 이벤트는 내부 이벤트(internal events), 외부 이벤트(external events), 서비스 이벤트(service events), 시간 이벤트(time events) 중 하나의 알림 메시지(notification message)가 된다.

내부 이벤트는 하나의 디바이스 내에서 다른 룰을 호출할 때 사용되며, 외부 이벤트는 디바이스 간의 연동을 수행할 때 다른 디바이스에 이벤트를 전달하기 위한 것이다. 서비스 이벤트는 호출된 내부 서비스가 완료된 경우 이를 알리는 이벤트의 발생을 정의한다. 그리고, 시간 이벤트는 시간 관련 이벤트를 정의한다. 즉, 주기적으로 발생하거나 특정 시간에 발생하는 이벤트를 정의할 수 있고, OR, AND, SEQ, NOT 연산자를 이용하여 복합 이벤트를 정의할 수 있다. 조건은 논리 표현식(boolean expression)으로 표현할 수 있으며, 이벤트 메시지에 있는 이벤트 변수나 디바이스에 의해서 유지되는 디바이스 변수를 활용하여 조건을 정의할 수 있다. 액션은 디바이스에 의해서 실행되는 명령을 의미하며, 내부 이벤트를 생성하는 액션인 createIntEvent, 외부 이벤트를 생성하는 액션인 createExEvent, 외부의 웹서비스를 호출하거나 내부 서비스를 호출하는 액션인 invoke가 있다. 또한, 이들 각 액션을 AND 연산자를 이용하여 복합 액션으로 정의할 수 있다.

본 절의 내용은 참고문헌 [6, 7]의 내용을 기반으로 한 것이며, 자세한 사항은 [6, 7]에 설명되어 있다.

Ⅲ. WS-ECA 프레임워크 설계

3.1 디바이스

디바이스는 자신에게 등록된 룰을 직접 실행하는 역할을 수행하며, 메시지 핸들러, 로컬 룰 매니저, DB 매니저, 시간 이벤트 생성기, WS-ECA 룰 데이터베이스, 룰 엔진의 6개 컴포넌트로 구성되어 있다. 이 중 룰 엔진의 경우는 이벤트 감지기, 조건 처리기, 액션 실행기, 룰 처리기의 4개의 컴포넌트를 포함한다.

메시지 핸들러는 외부에서 오는 모든 메시지를 받아서 적절한 내부 컴포넌트에게 전달한다.

로컬 룰 매니저는 DB 매니저를 활용하여 디바이스에 저장되는 WS-ECA 룰을 관리하는 역할을 한다. 처음 WS-ECA 룰이 디바이스에 등록을 요청되면, 로컬 룰 매니저는 룰의 기본적인 문법 검사를 한 후, GRM의 설계 시점 룰 검증기에게 룰을 전달해서 정의되어 있는 룰들이 자신이 가지고 있는, 혹은 다른 디바이스들이 가지고 있는 룰과 모순되는 부분이 있는지 검사한다. 문제가 없으면 룰은 WS-ECA 룰 데이터베이스에 저장되며, 로컬 룰 매니저는 이후에 룰의 유효성과 룰 관리에 필요한 다양한 요청들을 처리한다.

DB 매니저는 룰의 등록, 검색, 수정, 삭제 등 WS-ECA 룰 데이터베이스와 관련된 모든 상위 레벨의 인터페이스를 제공한다.

시간 이벤트 생성기는 룰에 정의된 시간 이벤트를 발생시켜 이벤트 감지기에게 전달한다.

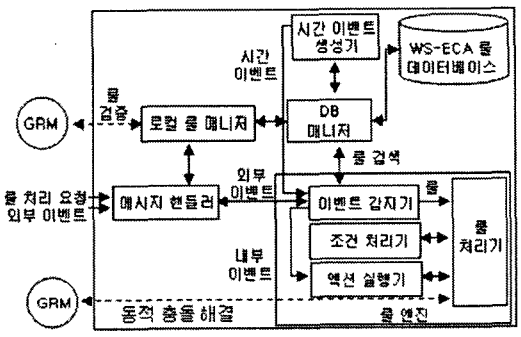


그림 3. WS-ECA 프레임워크의 디바이스 부분 설계
Fig. 3 Device part design of WS-ECA framework

이벤트 감지기는 내/외부에서 발생하는 이벤트들을 받아들이고 해당 이벤트와 관련된 룰을 검색하여, 바로

실행이 필요한 룰은 룰 처리기에게 전달하여 실행한다. 이벤트 감지기에게 전달되는 이벤트는 WS-Eventing 표준을 이용해 외부로 부터 전달되는 이벤트 알림 메시지, 시간 이벤트 생성기가 발생시키는 시간 이벤트, 액션 실행기가 발생시키는 내부 이벤트가 있다.

조건 처리기는 룰에 기술된 조건식을 검사한다. 룰 처리기는 이벤트 감지기로부터 실행이 필요한 룰을 전달받고, 이 룰의 조건식을 조건 처리기에게 전달한다. 조건 처리기는 전달받은 조건식을 검사한 후, 참/거짓 여부를 룰 처리기에게 전달한다.

액션 실행기는 룰에 기술된 액션을 수행한다. 룰 처리기는 조건 처리기의 결과를 받아 조건을 만족하는 룰의 액션들을 액션 실행기에게 전달하여 수행되도록 한다.

룰 처리기는 이벤트 감지기로부터 바로 실행이 필요한 룰을 받아서 조건 처리기, 액션 실행기와 함께 룰을 실행한다. 또한 전달 받은 룰이 동적 충돌을 발생시킬 가능성이 있다면, 즉, 룰 등록 당시 동적 충돌 가능성에 대한 표시가 되어 있다면, GRM의 실행시점 룰 검증기에게 확인 요청을 한다. 동적 충돌이 발생하는 경우는 충돌 해결 룰을 이용하여 룰을 실행한다.

3.2 GRM(Global Rule Manager)

GRM의 역할은 크게 세 가지로 나눌 수 있다. 첫 번째는 설계시점에 룰의 정적 충돌을 검증하고 동적 충돌의 발생 가능 조건을 확인하는 역할이며, 두 번째는 실행시점에 룰의 동적 충돌 발생 여부를 확인하고 이를 해결하는 역할이다. 세 번째는 WS-ECA 프레임워크 환경에 존재하는 서비스와 이벤트 정보를 관리하고, 이들 서비스의 상황을 모니터링 하는 것이다.

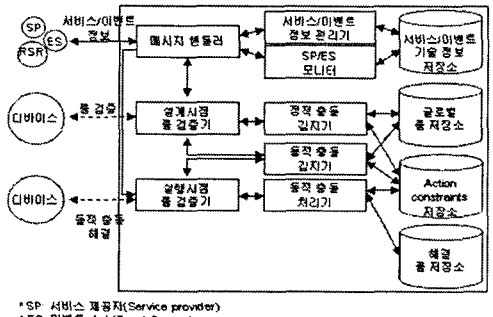


그림 4. WS-ECA 프레임워크의 GRM 설계
Fig. 4 GRM design of WS-ECA framework

GRM은 이러한 세 가지 역할을 수행하기 위하여 메시지 핸들러, 서비스/이벤트 정보 관리기, SP/ES 모니터, 서비스/이벤트 기술 정보 저장소, 설계시점 룰 검증기, 정적 충돌 감지기, 동적 충돌 감지기, 실행시점 룰 검증기, 동적 충돌 처리기, 글로벌 룰 저장소, Action constraints 저장소, 해결 룰 저장소의 12개의 컴포넌트로 구성된다.

메시지 핸들러는 외부에서 오는 모든 메시지를 받아서 적절한 내부 컴포넌트에게 전달한다.

서비스/이벤트 정보 관리기는 WS-ECA 프레임워크에 존재하는 서비스와 이벤트 정보를 서비스/이벤트 기술 정보 저장소를 이용하여 관리한다. 서비스/이벤트 기술 정보는 서비스/이벤트 정보 관리기가 각 디바이스에 요청하여 관리하며, 프레임워크의 다른 구성요소가 요청할 때 제공한다.

SP/ES 모니터는 WS-ECA 프레임워크에서 제공되는 서비스와 이벤트들이 발생하고 있는 상태를 모니터링 하여, 정상적으로 제공되지 않는 서비스와 이와 관련된 이벤트에 대한 정보를 룰 기반 서비스 요청자에게 알려준다.

룰 검증기는 설계시점과 실행시점에 수행되는 두 가지 종류가 있다. 설계시점 룰 검증기는 새로운 WS-ECA 룰 등록 시에 호출된다. 새로운 룰이 다른 룰과의 정적 충돌이 없는지, 그리고 동적 충돌이 발생할 가능성은 없는지를 검증한다. 정적 충돌이 발생한다면 사용자는 룰을 수정해야 하고, 동적 충돌이 발생 가능하다면 사용자는 동적 충돌을 해결하는 방법을 룰로 표현한 동적 충돌 해결 룰을 정의해야 한다. 반면에, 실행시점 룰 검증기는 동적 충돌이 발생할 가능성을 지닌 경우, 즉 룰 등록 시에 동적 룰 충돌 가능성에 대한 표시가 된 경우, WS-ECA 룰이 실행되기 직전에 호출된다. 동적 충돌이 발생하는지 여부를 검사한 후, 충돌이 발생한다면 미리 정의된 동적 충돌 해결 룰을 디바이스에게 전달한다.

정적 충돌 감지기와 동적 충돌 감지기는 글로벌 룰 저장소와 Action constraints 저장소를 기반으로 각각 정적 충돌와 동적 충돌을 검사하는 알고리즘을 수행한다. 또한, 동적 충돌 처리기는 동적 충돌 해결 룰을 기반으로 충돌을 해결하는 알고리즘을 수행한다.

GRM은 이를 처리하기 위하여 세 가지 저장소를 보유하고 있다. 첫 번째 저장소인 글로벌 룰 저장소는 GRM이 관리하고 있는 모든 디바이스에 등록된 WS-ECA 룰을 보유하고 있다. 두 번째 저장소는 동시에 발생할 수

없는 액션들을 지정한 Action Constraints를 저장하고 있다. Action Constraints는 동일한 이벤트에 의해 발생되는 활동들의 정적 충돌이나 동적 충돌을 판정하는 기준으로 사용된다. 세 번째 저장소는 실행시점에 발생 가능한 동적 충돌을 해결하는 기준을 제공하는 동적 충돌 해결 룰을 저장하고 있다.

Ⅳ. WS-ECA 프레임워크 구현

4.1 구현환경

WS-ECA 프레임워크의 구현은 기본적으로 리눅스 환경에서 진행하였고, 디바이스와 GRM의 기본 환경은 아파치 프로젝트로 진행되고 있는 Axis를 활용하였다. Axis는 SOAP 프로토콜을 지원하며 WS-Addressing과 WS-Eventing 표준을 지원하기 때문에 WS-ECA 프레임워크를 구현하는데 적합하다. 또한, WS-ECA 룰은 XML로 정의되므로, 이를 저장하고 관리하는 데이터베이스로 XML 문서를 효과적으로 지원할 수 있는 데이터베이스인 Xindice를 사용하였다. WS-ECA 룰을 받은 처리하는 로직(logic) 부분의 구현은 자바 언어를 사용하였다.

4.2 디바이스 프레임워크 구현

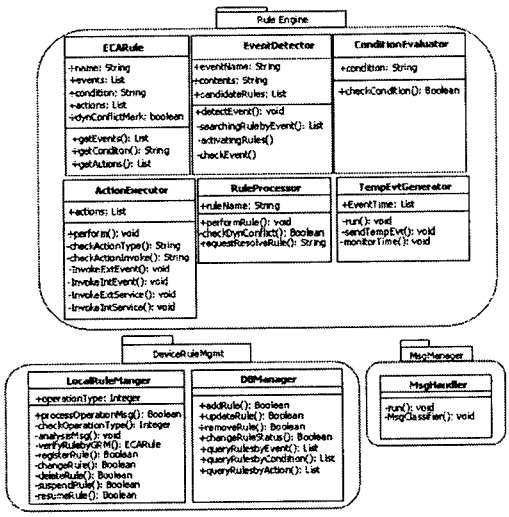


그림 5. WS-ECA 프레임워크의 디바이스 부분에 대한 클래스 정의

Fig. 5 Class definition of device part in WS-framework

디바이스 부분은 그림 5와 같이 크게 3개의 패키지로 나누어 구현하였다. 룰 엔진 패키지는 여섯 개의 클래스로 정의된다. EventDetector 클래스는 DBManager 클래스를 활용하여 감지된 이벤트와 관련된 룰을 찾고, 실행될 룰이 있으면 이를 RuleProcessor 클래스의 performRule 메소드에게 넘긴다. performRule 메소드는 Condition Evaluation 클래스와 ActionExecutor 클래스와 함께 룰을 실행한다. ActionExecutor는 쓰레드 클래스를 상속받아 생성된 클래스로 이는 디바이스가 동시다발적인 룰의 처리를 가능하게 하며, TempEvtGenerator 역시 쓰레드 형태로 동작하여 저장된 룰에 정의된 시간 이벤트에 따라서 이벤트를 발생시킨다. DeviceRuleMgmt 패키지는 두 개의 클래스로 구성되며, LocalRuleManager는 룰의 저장, 변경 등 다양한 룰 관리 기능에 대한 처리를 담당하며, 또한 새로운 룰 등록시 verifyRulebyGRM을 통해서 룰의 충돌에 대한 부분을 검사한다. DBmanager 클래스는 WS-ECA 룰 데이터베이스와 관련된 메소드를 제공하는 클래스이다. MsgManager 패키지는 MsgHandler 클래스를 포함하고 있으며, 이는 쓰레드 형태로 동작하며 들어오는 메시지를 분류하여 이를 처리하기 위한 클래스로 전달한다.

4.3 GRM 프레임워크 구현

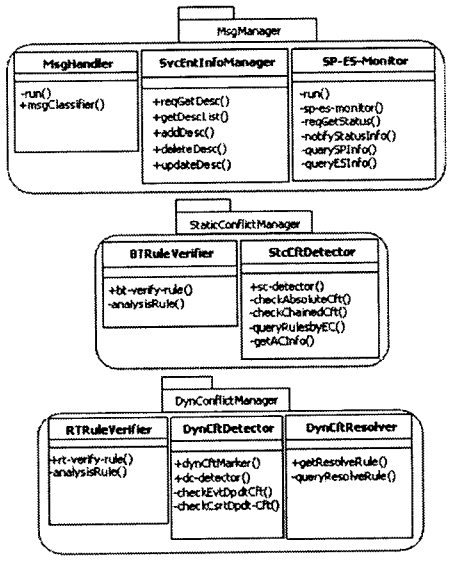


그림 6. WS-ECA 프레임워크의 GRM 부분에 대한 클래스 정의

Fig. 6 Class definition of GRM part in WS-ECA framework

GRM 부분은 그림 6과 같이 3개의 패키지로 구성되어 있다. MsgManager 패키지의 MsgHandler 클래스는 쓰레드로 동작하며 들어오는 메시지의 분류를 담당하며, SvcEvtInfoManager 클래스는 서비스와 이벤트의 정보 관리를 담당하는 클래스이다. 그리고, SP-ES-Monitor 클래스는 정의된 서비스와 이벤트가 정상적으로 동작하고 있는지에 대한 모니터링을 담당하는 클래스이다. StaticConflictManager 패키지의 BTRuleVerifier 클래스는 룰을 분석하여 정적 및 동적 충돌에 대한 확인을 담당하며, StcCftDetector 클래스는 정적 충돌에 대한 검사를 담당하는 클래스로 Absolute 충돌과 Chained 충돌에 대한 확인을 통해서 충돌 여부를 결정한다. DynConflictManager 패키지의 RTRuleVerifier 클래스는 실행시점에 룰의 동적 충돌을 검사를 담당하며, DynCftDetector 클래스는 룰 설계시점의 동적 충돌 마킹 부분 및 실행시점의 동적 충돌 감지를 담당한다. DynCftResolve 클래스는 감지된 동적충돌에 대한 해결 룰을 찾아주는 역할을 담당한다.

V. 결 론

최근 컴퓨팅 환경이 수많은 디바이스가 긴밀하게 네트워크로 연결되는 유비쿼터스 환경으로 빠르게 발전하면서, 이종의 하드웨어와 소프트웨어 플랫폼을 가지고 있는 다양한 디바이스들이 상호운용성을 보장하면서 자연스럽게 연동할 수 있는 대안으로 웹서비스가 대두되고 있다. WS-ECA 프레임워크는 웹서비스를 기반으로 유비쿼터스 환경의 동적인 특성을 지원하고, 소형 디바이스에 적용 가능한 WS-ECA 룰 기술 언어 및 이를 실행할 수 있는 프레임워크를 제안하였다.

본 논문에서는 WS-ECA 프레임워크의 핵심적인 컴포넌트인 디바이스와 GRM에 대한 상세한 모듈과 이들 간의 관계를 설계하여 제안하였고, 이를 기반으로 WS-ECA 프레임워크를 구현하였다. 향후 WS-ECA 프레임워크가 다양하고 수많은 디바이스들의 연동을 지원하기 위해서는 룰의 실행에 대한 성능 개선 방법에 대한 연구가 필요하다.

참고문헌

[1] Shankar R. Ponnekanti, Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd ICrafter: "A Service Framework for Ubiquitous Computing Environments" Proceedings of Ubicomp 2001, September 30-October 2, 2001

[2] Microsoft, The Microsoft invisible computing project web site. <http://research.microsoft.com/invisible/>.

[3] Aiello, M., Marchese, M., Busetta, P., and Calabrese, G. (2005a). "Opening the Home: a Web Service Approach to Domotics". In Applied Computing 2005, volume I:271-279.

[4] A. Alves, et al., Web Services Business Process Execution Language Version 2.0, OASIS, 2007.

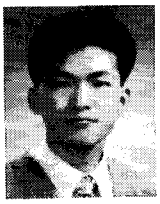
[5] N. Kavantzas, et al., Web services choreography description language Version 1.0, W3C Candidate Recommendation, 2005.

[6] Jaeyoon Jung, Jonghun Park, Seung-Kyun Han, and Kangchan Lee, "An ECA-Based Framework for Decentralized Coordination of Ubiquitous Web Services", To Appear in Information and Software Technology

[7] Kangchan Lee, Wonsuk Lee, Jonghong Jeon, Seungyun Lee, Jonghun Park, "Event-driven Coordination Rule of Web Services enabled Devices in Uniquitous environments", W3C Ubiquitous Web Workshop, March 2006

[8] G. vonBulltzingsloewen, A. Koschel, P.C. Lockemann, H.-D. Walter, ECA functionality in a distributed environment, in: N.W. Paton(Ed.), Active Rules in Database Systems, Springer, 1999, pp. 147 - .175.

저자소개



이 원 석(Won-Suk Lee)

1996. 배재대학교 전자계산학과 학사
1998. 충남대학교 컴퓨터공학과 석사
1998 ~ 2000. 한국교육학술정보원 연구원
2000 ~ 2002. 해동정보통신㈜ 기술연구소 연구원
현재 한국전자통신연구원 표준연구센터 연구원
※관심분야 : 데이터베이스, XML, 웹서비스, 모바일 웹, 유비쿼터스 웹서비스

신 동 민(Dongmin Shin)

1994, 한양대학교 산업공학과 학사
1996, 포항공과대학교 산업공학과 석사
2005, 펜실베니아 주립대 산업공학과 박사
1996-2001, 현대자동차 남양 연구소 연구원
2005-2006, 펜실베니아 주립대 박사 후 연구원
2006- , 한양대학교 정보경영공학과 전임강사
※관심분야 : 이산 사건 기반 시스템 설계, 유비쿼터스 웹서비스, 모바일 웹, 인간-컴퓨터 상호작용

이 규 철(Kyu-Chul Lee)

1984. 서울대학교 컴퓨터공학과 학사
1986. 서울대학교 컴퓨터공학과 석사
1990. 서울대학교 컴퓨터공학과 박사
1994. 미국 IBM Almaden Research Center 초빙연구원
1996. 미국 Syracuse University, CASE Center 초빙 교수
1997. ~ 1998. 첨단학술정보센터 파견 교수
2000. ~ 현재. 한국 ebXML 전문위원회 위원장
2001. ~ 현재. 한국정보과학회 논문편집위원
현재 충남대학교 공과대학 컴퓨터공학과 교수
※관심분야 : 데이터베이스, XML, 정보 통합, 멀티미디어 시스템, e-비즈니스 시스템, 유비쿼터스 웹 서비스