웹 서비스 기반의 유비쿼터스 워크플로우 언어

한 주 현[†] · 조 용 윤[†] · 최 재 영[††]

요 약

현재 비즈니스 및 분산 컴퓨팅 환경에서 사용되는 워크플로우는 서비스 통합 및 자동화, 동적 흐름 관리, 동시 수행, 실시간 서비스 등의 기능을 제공한다. 유비쿼터스 컴퓨팅 환경에서 서비스들은 사용자와 현재 사용자가 속해 있는 환경으로부터 다양한 정보를 제공받아 상황에 맞는 서비스를 동적으로 제공해야 한다. 이와 같은 서비스를 효과적으로 제공하기 위해서는 서비스의 자동화에 사용되고 있는 워크플로우를 유비쿼터스 컴퓨팅에 적용하고, 상황인지 서비스를 제공하기 위해 상황 정보를 워크플로우의 전이조건으로 명시해야 한다.

본 논문에서는 유비쿼터스 환경에서 발생하는 상황 정보를 워크플로우의 서비스 전이조건으로 사용하기 위한 웹 서비스 기반의 워크플로우 언어인 uWDL (Ubiquitous Workflow Description Language)을 제안한다. uWDL은 상황 정보인 컨텍스트, 프로파일 및 이벤트 정보에 따라 사용자 상황에 맞는 서비스를 선택할 수 있다. 또한 uWDL 전용 파서와 컨텍스트 처리기를 통해 uWDL로 기술된 시나리오 문서가 제대로 실행되는지를 검증하였다. 유비쿼터스 환경을 위한 시나리오 설계를 위해 개발자는 uWDL 시나리오 편집기를 이용하여 유비쿼터스 환경의 상황 정보를 고려해 그에 맞는 서비스들의 관계를 워크플로우 형태로 기술함으로써 작업을 통합하고 자동화할 수 있다.

키워드 : 유비쿼터스/오토노믹 컴퓨팅, 워크플로우, 웹 서비스

A Ubiquitous Workflow Language based on Web Services

Joohyun Han[†] · Yongyoon Cho[†] · Jaeyoung Choi[††]

ABSTRACT

Currently workflows in business processes and distributed computing environments have provided service automation by connecting many tasks with rules and/or orderings. The services in ubiquitous computing environments have to automatically provide users with adaptive services according to dynamically changing context information, which is obtained from both the users and their environment. To adapt these workflows to ubiquitous computing, we must specify the situation information on their transition conditions.

In this paper, we propose uWDL, Ubiquitous Workflow Description Language, based on Web Services to use the situation information on the transition constraints of workflow's services. uWDL can select adaptive services according to contexts, profiles, and events information, which are situation information. Furthermore, we verified the execution of a scenario document described with uWDL using the parser and the context handler for uWDL. The scenario developers can use the uWDL scenario editor for a design of scenarios, and they can easily specify the transition conditon of the services according the situation information of ubiquitous environments using the uWDL.

Key Words : Ubiquitous/Autonomic Computing, Workflow, Web Services

1. 서 론

유비쿼터스는 "언제 어디에든 존재한다[1]"라는 의미의 라틴어로 1988년 처음으로 제록스 팔러 알토 연구소의 마크 와이저에 의해 소개되었다. 그는 유비쿼터스 컴퓨팅을 "현실세계의 어디서나 네트워크와 연결된 컴퓨터를 눈에 띄지 않는 형태로 언제 어디서나 사용 가능하며 사용자 상황에 따라 서비스가 변하는 환경[2]"이라고 정의하고, 이는 컴퓨팅 환경이 현실세계의 사물과 환경속으로 스며들어 일상생활에 통합되는 것[3]을 의미한다.

기존의 전통적인 컴퓨팅 환경의 비즈니스, 분산 서비스들은 서비스 통합 및 자동화, 동적 흐름 관리, 동시 수행, 실시간 서비스 등의 기능을 제공하기 위해 서로 연계된 작업의 흐름을 가지는 워크플로우 모델을 가진다. 워크플로우는 어떤 특정 목적을 달성하는 과정에서 해당 서비스가 제시간에 해당 사용자 및 응용 프로그램에 전달될 수 있도록 한다. 그러나 기존의 비즈니스 프로세스 및 분산 컴퓨팅 환경에 적용되었던 워크플로우를 유비쿼터스 컴퓨팅 환경에 그대로 적용하기에는 다음과 같은 한계가 있다. 첫째, 유비쿼터스 컴퓨팅 환경에서는 작업이 수행되는 자원의 수와 종류가 사용자의 이동에 따라 수시로 변화하기 때문에 사용자의 작업 흐름뿐만 아니라 공간 및 시간 변화도 고려해야 한다. 둘째, 사용자가 제공하는 정보뿐만 아니라 사용자가 속해 있는 환

※ 본 연구는 2005년도 숭실대학교 교내연구비 지원으로 이루어졌습니다.
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 컴퓨터학부 부교수
논문접수 : 2005년 9월 12일, 심사완료 : 2005년 10월 26일

경으로부터 발생하는 상황 정보에 맞게 적절한 서비스를 선택해야 한다. 셋째, 사용자의 이동에 따라 사용자가 속해 있는 환경이 동적으로 변화하기 때문에 환경 자체가 다양하면서 이질적인 특성이 있다. 이러한 상황에서 사용자들은 동적으로 변화하는 환경에서 사용자가 원하는 서비스들을 사용자가 원하는 형태로, 원하는 시간에, 사용자의 직접적인 개입없이 자동화된 서비스를 제공받기 원한다. 그러므로 유비쿼터스 컴퓨팅 환경에서 사용자에게 상황에 따라 적절한 서비스를 제공하기 위해서는 전통적인 컴퓨팅 환경의 서비스 자동화에 사용되던 워크플로우 모델을 유비쿼터스 컴퓨팅 환경에 적용하기 위한 연구가 필요하다.

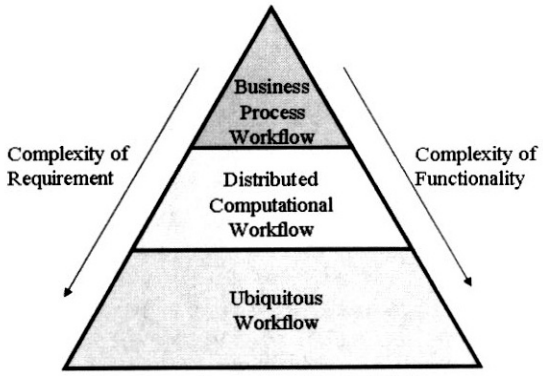
본 논문에서는 유비쿼터스 환경에서 발생하는 상황 정보인 컨텍스트, 프로파일 및 이벤트와 이를 이용하는 서비스간의 연관 관계를 규칙 기반의 서술 방식을 통하여 효과적으로 표현할 수 있는 uWDL(Ubiquitous Workflow Description Language)을 제안한다. uWDL은 유비쿼터스 환경에서 사용자의 특성을 파악하고 그에 맞는 서비스들의 관계를 기술할 수 있는 언어로, 이를 이용하여 서비스들 간의 연관 관계를 워크플로우로 구성함으로써 작업의 통합 및 자동화를 이룰 수 있다. 또한 이질적인 컴퓨팅 환경을 위해 웹 서비스 표준인 WSDL(Web Service Definition Language)을 이용하여 uWDL의 서비스를 기술하고, 워크플로우에 의해 구성된 서비스들의 연관 관계를 컨텍스트의 의미정보에 따라 동적으로 재구성하여 상황에 맞는 서비스를 제공할 수 있다.

논문의 구성은 다음과 같다. 2장에서는 유비쿼터스 컴퓨팅 환경에서 워크플로우의 필요성에 대해 설명한다. 3장에서는 유비쿼터스 컴퓨팅 환경을 위한 웹 서비스 기반의 워크플로우 언어인 uWDL을 제안한다. 4장에서는 3장에서 제안한 uWDL 언어로 기술된 시나리오 문서를 처리하는 방식에 대해 설명하고, 5장에서는 이 시나리오를 통해 uWDL 언어를 검증한다. 6장에서는 관련연구와 본 논문의 유비쿼터스 워크플로우 언어를 비교하고, 7장에서는 본 논문의 결론을 내린다.

2. 유비쿼터스 컴퓨팅 환경에서 자동화 서비스를 위한 요구사항

2.1 상황인지 워크플로우의 필요성

WfMC(Workflow Management Coalition)에서는 워크플로우를 "전체적인 또는 부분적인 비즈니스 프로세스의 자동화를 의미하며, 이때 문서, 정보, 태스크가 한 사용자에서 다른 사용자로 일련의 업무절차 규칙에 의한 처리를 위해 전달된다[4]"로 정의하고 있다. 워크플로우는 하나의 큰 작업이 수행 완료될 때까지 일어나는 하위 작업들의 수행 흐름을 XML 기반의 언어를 이용하여 표준화된 방법으로 표현한다. 이때 워크플로우의 하위 작업들 간에는 의존성이나 수행 순서, 동시 수행 가능 여부 등의 다양한 관계가 나타나게 된다. 워크플로우 관리 시스템은 워크플로우의 작업 흐름을 결정하기 위하여 워크플로우 언어에 명시된 상태 전이 제약조건을 사용하고, 이러한 제약 조건은 워크플로우 언어에서 명시된다.

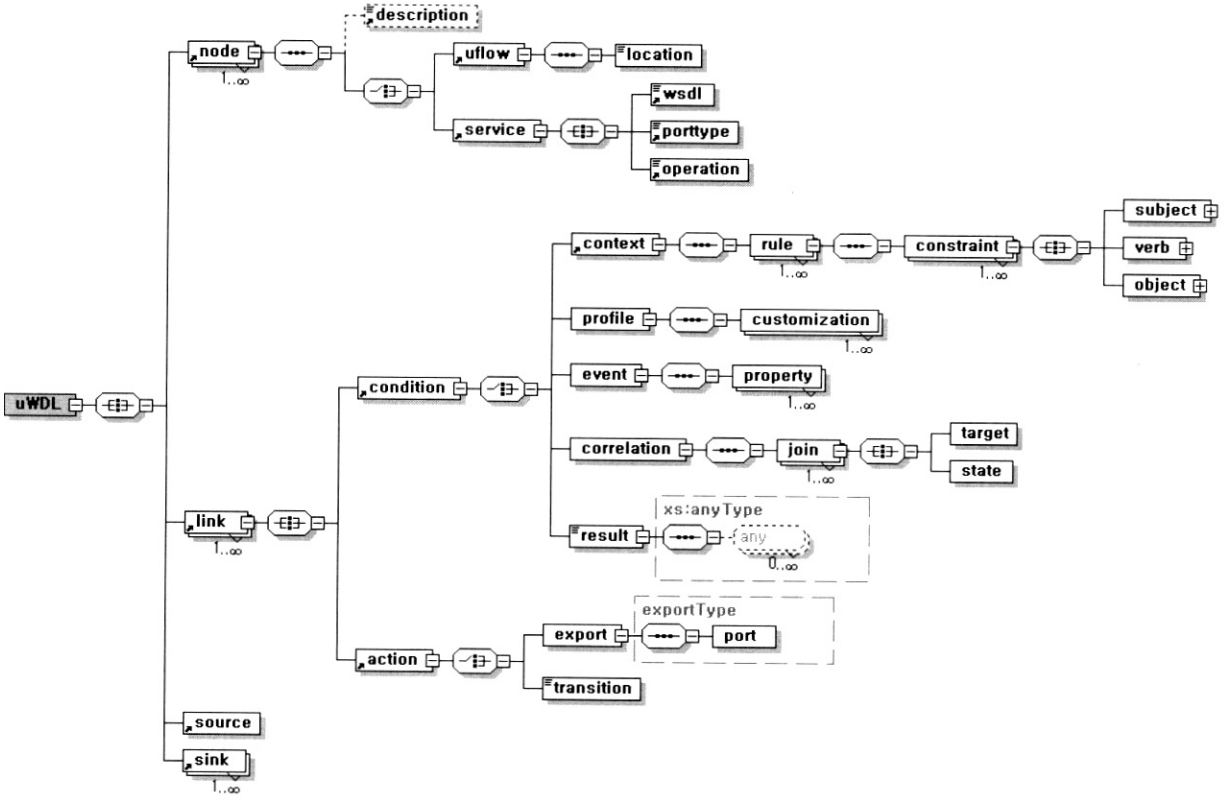


(그림 1) 워크플로우 발전 방향

(그림 1)은 워크플로우의 발전 방향과 그에 따른 워크플로우의 요구사항 및 기능의 복잡도를 나타낸다. 워크플로우 모델이 초기에는 주로 비즈니스 프로세스에 적용되었다. 비즈니스 프로세스란 일반적으로 기능적인 역할과 관계를 정의하는 조직 구조의 맥락에서 업무 목표나 정책적인 목적을 총괄적으로 실행하는 일련의 연관된 업무 절차이다. 다양한 비즈니스 프로세스의 효율성 향상에 기여했던 워크플로우 개념이 최근에는 분산 컴퓨팅에 활발히 적용되고 있고, 나아가서는 그리드 컴퓨팅에도 워크플로우를 적용하려는 시도가 활발히 이루어지고 있다[5]. 유비쿼터스 컴퓨팅 환경의 워크플로우는 기존 컴퓨팅 환경과는 달리 사용자의 요구사항뿐만 아니라 사용자가 속해있는 환경으로부터 발생하는 상황 정보에 따라 서비스를 선택해야 하기 때문에 기능의 복잡도가 더욱 증가하게 된다. 따라서 현재의 워크플로우 모델은 유비쿼터스 컴퓨팅 환경처럼 동적으로 변화하는 환경에서 상황인지 기반의 서비스 자동화를 제공하기 위해 진화되어야 한다. 이를 위해 유비쿼터스 환경의 워크플로우는 서비스의 흐름을 명시하는 워크플로우에서의 상태를 전이시키기 위한 제약조건에 상황 정보인 컨텍스트를 전이조건으로 명세함으로써 워크플로우가 상황인지 서비스를 제공할 수 있다.

2.2 웹 서비스의 필요성

웹 서비스는 인터넷이나 네트워크상에 떨어져 있는 객체의 함수를 호출할 수 있는 미들웨어로 새로운 기술을 의미하지는 않는다. 그러나 기존의 기술과 다른점은 플랫폼 중립적인 표준들, 예를 들어 HTTP 및 XML을 사용하는데 있으며, 이와 같은 특징은 서비스가 언어, 프로토콜 및 플랫폼에 상관없이 사용될 수 있도록 한다[6]. WSDL은 서비스를 기술하는 언어로 서비스 제공자가 제공하려는 웹 서비스 기능을 XML 문법을 이용하여 묘사하는 기술을 제공한다. 유비쿼터스 컴퓨팅 환경의 워크플로우는 다양한 플랫폼과 프로토콜로 구성되고 서로 다른 언어들에 의해 개발된 수많은 응용 프로그램을 통합, 관리 및 실행되어야 한다. 따라서 워크플로우는 표준화되고 이질적인 플랫폼, 프로토콜 및 언어



(그림 2) uWDL 스키마 구조

에 상관없는 웹 서비스 인터페이스를 필요로 한다. 웹 서비스의 플랫폼 중립적인 표준들의 특성은 유비쿼터스 컴퓨팅 환경에서 워크플로우의 필요성을 만족시킨다.

3. 웹서비스 기반의 유비쿼터스 워크플로우 언어

현재 BPEL4WS[7], WSFL[8], XLANG[9]과 같은 웹 서비스 기반의 워크플로우 언어들은 유비쿼터스 컴퓨팅 환경에서 발생하는 다양한 컨텍스트, 프로파일 및 이벤트 정보에 따라 서비스를 분기하는 기능을 포함하지 않는다. 그렇기 때문에 기존의 워크플로우 언어를 이용하여 유비쿼터스 환경에 필요한 서비스들의 연관관계를 효과적으로 표현하는데 한계가 있다. 또한 유비쿼터스 환경은 이질적인 특성을 지니기 때문에 이와 같은 특성을 극복할 수 있는 형태의 미들웨어가 필요하다.

uWDL(Ubiquitous Workflow Description Language)은 웹 서비스 기반의 워크플로우 언어이기 때문에 유비쿼터스 컴퓨팅과 같이 시스템 환경이 이질적이면서 다양한 목적의 서비스들을 통합하는데 유용하다. 또한 기존의 워크플로우 언어가 표현하지 못했던 컨텍스트, 프로파일, 이벤트 정보를 서비스의 전이조건에 추가하여 상황 정보에 따라 워크플로우의 흐름 관계를 효과적으로 기술할 수 있다. uWDL은 센서에 의해 감지되는 정보들이 온톨로지[10, 11]에 의해 고수준의 컨텍스트, 프로파일 및 이벤트 등의 정보로 제공되면, 이 정보를 이용하여 서비스의 흐름을 결정하기 위해 컨텍스트 및 프로파일 정보를 {주어, 동사, 목적어} 형태로 표현한다. 또한 온톨로지 형태의 추론 서비스를 이용하기 위한 규칙 기반의 서술 기능도 포함하고 있다. (그림 2)는 uWDL 스키마 구조를 나타낸다.

uWDL은 크게 <node>, <link>, <source>, <sink> 원소들로 구성된다. <node> 원소는 웹 서비스 형태로 개발된 각각의 서비스를 가리키며, <link> 원소는 상황 정보인 컨텍스트, 프로파일, 이벤트 등에 따라 서비스를 결정하기 위해 사용된다. 또한 <source>와 <sink> 원소는 서비스의 시작과 끝을 의미한다. 이와 같은 원소들 중에 중요한 의미를 지니고 있는 <node>와 <link> 원소에 대해 살펴보면 다음과 같다.

3.1 <node> 원소

<node> 원소는 웹 서비스의 하나의 기능을 의미하는 오퍼레이션(Operation)을 나타낸다. 웹 서비스는 서비스의 기능을 명시하기 위해 WSDL 표준을 이용하여 자신의 포트타입(PortType) 및 오퍼레이션 등의 구체적인 서비스를 명시하게 되고, 이를 이용하기 위해 uWDL에서는 <service> 원소의 <wsdl>, <porttype>, <operation> 원소들로 원하는 서비스의 위치, 서비스 타입 및 오퍼레이션에 대한 정보를 기술하게 된다. 또한 <uflow> 원소는 uWDL로 기술된 다른 워크플로우 기술문서를 가리킨다.

3.2 <link> 원소

<link> 원소는 uWDL 언어의 가장 중요한 부분으로 유비쿼터스 환경에서 발생할 수 있는 컨텍스트, 프로파일 및 이벤트 등을 명시하며, 이에 따라 서비스의 흐름이 결정되도록 정의한다. (그림 2)에서와 같이 <link> 원소는 크게 <condition>과 <action> 원소로 구분된다. <condition> 원소는 <context>, <profile> 및 <event> 원소를 통하여 해당 노드의 컨텍스트, 프로파일 및 이벤트의 상태를 규칙 기반의 서술방식을 통하여 명시하며, 상태가 참(true)일 경우 <action>에서 기술하는 행동을 수행한다. <action> 원소는 <export>와 <transition>으로 구분되며, <export>의 속성에 따라 제어 링크(control link)와 데이터 링크(data link)로 나뉜다. 그리고 <transition>은 현재 노드의 상태 변화를 명시하기 위해서 사용된다.

<condition> 원소는 컨텍스트, 프로파일 및 이벤트 정보 등에 따라 적절한 서비스가 선택되도록 결정해주는 역할을 한다. 중요한 하위 원소로는 <context>와 <profile> 원소를 들 수 있다. <context> 원소는 온톨로지 및 추론 서비스에 의해 생성된 고수준의 컨텍스트 정보를 워크플로우 서비스의 전이조건으로 명시하기 위해 <constraint> 원소를 사용한다. <constraint> 원소는 온톨로지 기반구조인 RDF(Resource Description Framework)[10] 표기법과 유사한 형태로 컨텍스트 정보를 기술하기 위해 하위 원소로 주어(subject), 동사(verb), 목적어(object) 원소를 가진다. 현재 컨텍스트 정보를 표현하는 방법에는 5W1H[12] 등 여러 표기법이 있지만, {주어, 동사, 목적어} 구조는 인간의 사고방식에 가까운 표기법이면서 컴퓨터가 이해하고 처리할 수 있는 구조를 가지고 있기 때문에 수많은 컨텍스트 정보를 의미 있는 지식 정보로 표현하는데 유용하다. 또한 온톨로지 형태로 표현된 컨텍스트 정보를 손쉽게 이용하기 위해서는 {주어, 동사, 목적어} 형태의 표기법이 효과적이다. 구체적으로 온톨로지와의 연동을 위해 <subject>, <verb>, <object> 원소의 "type" 속성은 온톨로지의 클래스에 해당하며, 원소의 값은 온톨로지의 인스턴스에 해당한다. (그림 3)은 "A의 위치가 301호이다"를 <constraint> 원소를 이용하여 표현한 내용이다.

<subject>와 <object> 원소는 구조적인 컨텍스트 모델의 엔티티(entity)에 의해 제공된다. 엔티티는 컨텍스트 정보의 주체를 의미하며, 유비쿼터스 환경에서 모든 객체는 엔티티가 될 수 있다. 그러므로 <constraint> 원소는 두 개의 엔티티인 <subject>와 <object>의 관계를 통해서 컨텍스트를 표현한다. 또한 <constraint> 원소의 composite 속성은 and, or, not의 속성 값을 가지며, 이를 통하여 단순 컨텍스트들의 관계를 보다 고수준인 합성 컨텍스트로 기술할 수 있다.

```
<constraint>
        <subject type="PersonType">A</subject>
        <verb type="LocationType"/>
        <object type="RoomType">301</object>
</constraint>
```

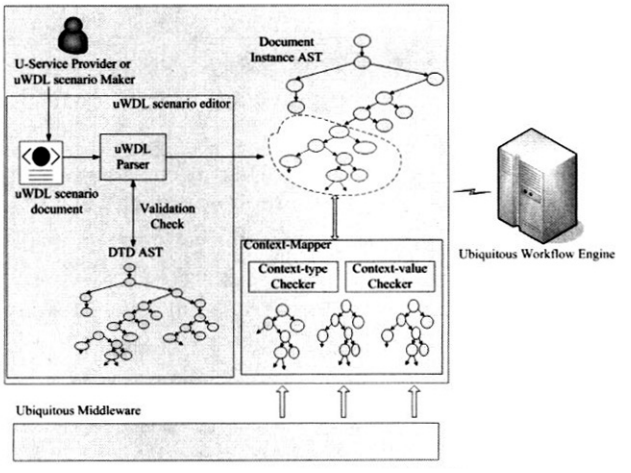
(그림 3) <constraint> 원소를 이용한 컨텍스트 표현

<rule> 원소는 <constraint> 원소들의 집합을 의미하며, 하나의 상황을 결정하기 위한 가장 상위의 표현 방식을 의미한다. <profile> 원소는 사용자의 특성 또는 사용자로부터 입력된 컨텍스트 정보를 표현하기 위해 필요하다. 이는 <context> 원소에 비해 정형화된 컨텍스트의 표현에 사용됨을 알 수 있다. <event> 원소는 현재 수행되고 있는 서비스에서 발생하는 event 정보를 처리하기 위해 사용되며, event 정보는 changeOfLocation, changeOfProfile 등과 같이 표현된다. 또한 다른 서비스와의 협업을 위한 <correlation> 원소를 가지고 있다.

4. uWDL을 위한 컨텍스트 처리기 구조

유비쿼터스 환경을 위한 워크플로우 서비스를 개발하려는 시나리오 개발자는 uWDL 편집기를 이용하여 시나리오 문서를 작성하게 된다. 이처럼 시나리오 기반의 워크플로우 서비스를 제공하기 위해서는 uWDL로 기술된 시나리오 문서에 기술된 컨텍스트 정보와 센서로부터 전달되는 컨텍스트를 비교하여 상황에 맞는 서비스를 실행할 수 있는 기능이 필요하다. 이를 위해 본 논문에서는 uWDL 시나리오 문서를 해석하고 컨텍스트 정보를 처리하기 위한 uWDL 컨텍스트 처리기를 설계하였다.

(그림 4)는 uWDL 컨텍스트 처리기의 전체 구조이다. uWDL 컨텍스트 처리기는 크게 uWDL 시나리오 문서를 파싱하기 위한 uWDL 파서와 파싱 정보 중 컨텍스트 정보를 처리하기 위한 컨텍스트 비교기(context mapper)로 구성된다. 컨텍스트 처리기의 수행과정은 다음과 같다. 시나리오 개발자는 uWDL 시나리오 편집기를 이용하여 시나리오 문서를 작성한다. 작성된 시나리오 문서는 uWDL 파서에 의해 번역되어 DIAST(Document Instance Abstract Syntax Tree) 형태로 저장된다. DIAST는 uWDL 시나리오 문서의 문법적 구조 정보와 내용 정보를 트리 형태로 표현한 자료구조이다. 이때 컨텍스트 비교기는 센서 네트워크로부터 수집된 컨텍스트 정보와 DIAST의 컨텍스트 정보를 비교하여



(그림 4) uWDL 컨텍스트 처리기 구조

워크플로우 서비스의 전이 조건으로 사용한다. 이와 같은 컨텍스트 처리기 구조를 구체적으로 이해하기 위해서 다음 절에서는 중요한 구성요소인 컨텍스트와 엔티티, 컨텍스트 트리플릿 및 컨텍스트 비교기에 대해 살펴본다.

4.1 컨텍스트와 엔티티

유비쿼터스 환경에서 컨텍스트란 "사용자 또는 응용 프로그램과 상호 작용을 갖는 사람, 장소, 사물에 대한 상황 또는 상태 정보[13]"를 의미한다. 유비쿼터스 환경의 워크플로우에서 사용되는 컨텍스트의 타입은 워크플로우 도메인에 따라 그 종류가 서로 다르며 매우 다양하다. 일반적으로 사용되는 컨텍스트의 타입은 크게 시간, 위치, 아이덴티티(Identity), 엑티비티(Activity) 등으로 분류할 수 있다. 또한 (그림 4)에서, uWDL 시나리오 문서에 기술된 컨텍스트 정보와 유비쿼터스 미들웨어로부터 객체화된 컨텍스트 정보는 엔티티[14] 형태로 표현된다. 엔티티는 컨텍스트를 표현하기 위해 사용되며, 컨텍스트는 특정 도메인에 포함되는 엔티티들의 타입과 값을 통해 기술된다. 엔티티의 타입과 값을 동시에 사용하는 이유는 센서 네트워크로부터 생성되는 어떤 엔티티 객체는 하나의 도메인 내에서 같은 타입을 가질 수 있다. 이것은 같은 도메인 내에서 발생하는 서로 다른 의미의 엔티티 객체를 타입만으로는 구별할 수 없다는 것을 의미한다. 이와 같은 컨텍스트 비교의 모호성을 해결하기 위해, uWDL 시나리오에서의 컨텍스트는 <constraint>의 <subject>, <verb>, 그리고 <object> 원소들의 타입과 값으로 표현되며, 센서 네트워크로부터 생성되는 엔티티 정보 또한 표현하려는 상황 정보의 타입과 값으로 객체화된다.

4.2 컨텍스트 트리플릿(Context Triplet)

uWDL 컨텍스트 처리기에 포함된 uWDL 파서는 문서 파싱 과정을 통해 uWDL 시나리오 문서를 해석하고 결과로써 DIAST를 생성한다. uWDL 파서가 생성하는 DIAST는 uWDL 시나리오 문서에 기술된 컨텍스트의 문법 검사와 내용 검사를 위한 유용한 자료구조로 사용된다. 이때, uWDL 시나리오 문서에 기술된 컨텍스트 정보는 <subject>, <verb>, 그리고 <object> 원소의 순차적 의미구조인 컨텍스트 트리플릿(triplet) 형태로 표현된다.uWDL 시나리오 문서에서 기술되는 컨텍스트 정보는 <constraint> 원소로 표현되는 하나 이상의 컨텍스트 트리플릿을 가진다. 또한 uWDL 시나리오 문서에 표현된 컨텍스트 정보는 uWDL 파서가 파싱 후 생성하는 DIAST의 일정 부분에 부분트리(subtree)로 표현된다. (그림 4)의 DIAST에서 점선으로 표현된 부분트리는 uWDL 시나리오 문서에 기술된 컨텍스트 정보를 나타낸다.

4.3 컨텍스트 비교기(Context Mapper)

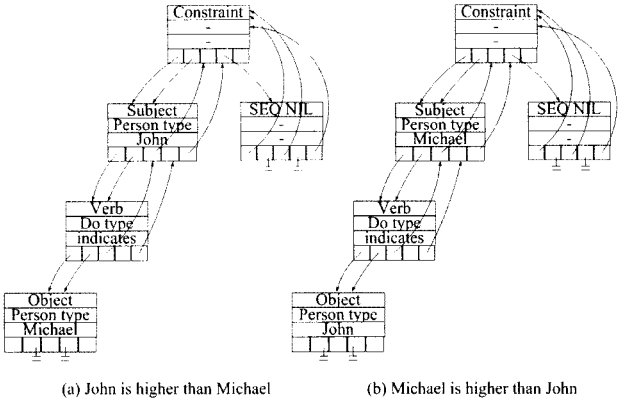
컨텍스트 비교기는 파서가 생성한 DIAST를 통해 uWDL 시나리오 문서에 기술된 컨텍스트 정보와 센서 네트워크로부터 생성된 엔티티 객체들을 비교한다. 이를 위해, 컨텍스트 비교기는 센서 네트워크로부터 생성된 상황 정보인 엔티티 객체들로부터 타입과 값을 추출하고, 추출된 엔티티의 타입과 값은 DIAST의 <constraint> 원소가 포함하는 각각의 <subject>, <verb>, <object> 원소의 타입 및 값과 비교된다. 만약 <constraint> 원소의 <subject>, <verb>, <object> 원소들과 추출된 엔티티 정보들 사이의 타입과 값이 일치하는 원소가 발견되면, 컨텍스트 비교기는 센서 네트워크로부터 생성된 현재의 엔티티 정보가 uWDL 시나리오에 기술된 컨텍스트 트리플릿의 한 원소와 일치되는 상황 정보임을 확인한다. 이와 같은 방법으로 컨텍스트 트리플릿의 모든 원소인 <subject>, <verb>, <object> 원소들의 타입과 값이 수집된 엔티티 정보들과 모두 일치하게 되면, 이는 하나의 상황 정보 단위인 컨텍스트 트리플릿 정보가 현재 상황과 일치됨을 의미하고 이에 해당하는 서비스가 호출된다. 예를 들어, "존이 마이클을 가리키고 있다 (John indicates Michael)"라는 컨텍스트 정보를 가정해 보자. 예제 컨텍스트는 uWDL 시나리오의 <constraint> 원소를 이용하여 (그림 5)와 같이 타입과 값을 가지는 컨텍스트 트리플릿으로 표현된다.

이때, 센서 네트워크로부터 다음과 같은 엔티티 객체들이 - *(PersonType, John)*과 *(PersonType, Michael)* - 발생되었다고 가정해 보자. (그림 6)은 uWDL 컨텍스트 비교기가 uWDL에 기술된 컨텍스트와 센서 네트워크로부터 발생된 엔티티 객체를 비교하는 모습을 나타낸다.

예제 컨텍스트를 표현한 <constraint>의 DIAST 부분트리는 (그림 6)의 (a)와 같다. 또한 센서 네트워크로부터 생성된 엔티티 객체 *(PersonType, John)*과 *(PersonType, Michael)*에 대한 DIAST 부분트리 표현은 각각 (그림 6)의 (a)와 (b)와 같이 구성될 수 있다. 이때, (그림 6)의 (a)와 (b)의 <subject> 엔티티는 같은 PersonType의 타입 정보를

```
<constraint>
<subject type=PersonType>John</subject>
<verb type=DoType>indicates</verb>
<object type=PersoType>Michael</object>
</constraint>
```

(그림 5) "John indicates Michael"을 <constraint> 원소를 통해 표현한 컨텍스트 트리플릿



(그림 6) "John indicates Michael"에 대한 컨텍스트 비교

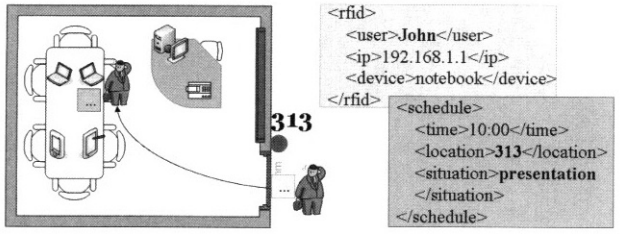
가지므로, 타입 정보만을 이용한 컨텍스트 비교가 진행될 때 비교 모호성이 발생될 수 있다. 즉, 같은 PersonType의 타입 정보를 갖는 엔티티 객체*(PersonType, John)*과 *(PersonType, Michael)*는 모두 <subject> 엔티티로 사용될 수 있으므로, (그림 6)의 (a)와 (b) 형태의 컨텍스트 구성을 나타내는 부분 트리가 가능하다. 이러한 모호성은 단순히 컨텍스트 타입만을 비교함으로써 발생하는 것이다. 그러나 (그림 6)의 (a)와 (b)를 구성하는 <subject> 엔티티는 값-John과 Michael-에 의해 구별된다. 따라서 uWDL 컨텍스트 비교기는 예제 컨텍스트를 표현한 <constraint>의 DIAST 부분트리와 일치하는 *(PersonType, John)*엔티티 객체를 <subject> 엔티티로 갖는 (그림 6)의 (a)를 uWDL에 기술된 서비스를 실행하기 위해 필요한 올바른 컨텍스트로 인식하게 된다.

5. 시나리오를 통한 uWDL의 검증

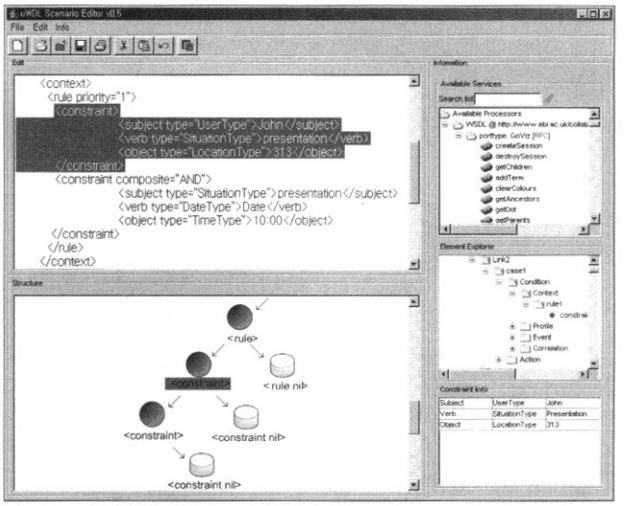
이제까지 소개한 uWDL은 유비쿼터스 환경의 모든 서비스들을 컨텍스트, 프로파일 및 이벤트 정보에 따라 상황에 맞는 서비스의 흐름을 결정할 수 있는 워크플로우 언어이다. 또한 웹 서비스 기반의 언어이기 때문에 언어, 프로토콜 및 플랫폼에 상관없이 서비스들을 사용하는 장점이 있다. 이와 같은 uWDL의 특징을 검증하기 위해 유비쿼터스 환경의 서비스 중에서 회의 준비를 위한 서비스를 구상하였다. 시나리오의 목적은 "자신의 일정 계획에 따라 회의를 자동으로 준비해 주는 서비스를 구축한다"이다. 이에 따라 시나리오를 설계하면 (그림 7)과 같다.

"회사원 A씨는 출근 후 자신의 노트북을 이용하여 일정 계획에 아침 10시에 313호 회의실에서 발표가 있음을 기록한다. 발표 준비를 하던 A씨는 9시 40분경에 회의를 위해 313호 회의실로 이동한다. 313호 회의실 문 위쪽에는 RFID 센서가 설치되어 있어 A씨의 기본 정보(이름, 노트북의 IP 주소 등)를 서버로 전송한다. 서버에서는 전송된 A씨의 IP 주소를 이용하여 일정 계획 서비스로부터 일정 계획 정보를 획득하여 현재 시간 (Time), 위치 정보 (Location) 및 현재 상황 (Situation) 등을 비교하여 참(true)일 경우 A씨의 발표 자료를 다운로드하여 해당 프로그램을 실행한다."

본 논문에서는 uWDL을 이용하여 개발자가 손쉽게 시나리오를 설계할 수 있도록 GUI 기반의 uWDL 시나리오 편집기를 구현하였다. uWDL 시나리오 편집기는 텍스트 편집 창과 편집된 uWDL 시나리오 구조를 트리 구조로 표현해 주는 트리구조 창으로 이루어진다. 또한 편집 창과 트리구



(그림 7) 회의 준비 시나리오



(그림 8) uWDL 시나리오 편집기

조 창은 사용자가 선택하는 특정 uWDL 노드에 대한 타입 및 값을 설정할 수 있는 파라메터(parameter) 창을 제공한다. uWDL 시나리오 편집기는 본 논문에서 설계한 uWDL 컨텍스트 처리기 즉, uWDL 파서와 uWDL 컨텍스트 비교기를 포함한다. 따라서 uWDL 편집기를 통해 사용자가 생성하는 uWDL 시나리오 문서는 uWDL 파서를 통해 DIAST로 변환 생성되어 uWDL 시나리오 편집기의 트리 구조 창에 표시된다. 트리 구조창에 표현된 uWDL 시나리오의 구조정보를 통해 사용자는 생성한 uWDL 시나리오의 전체적인 구조를 빠르고 쉽게 파악할 수 있으며, 특정 노드의 속성 정보를 입력할 때 쉽게 사용할 수 있다. (그림 8)은 uWDL 시나리오 편집기를 이용하여 (그림 7)에서 표현한 예제 시나리오에 대해 uWDL 시나리오 문서와 DIAST의 생성을 보여주며, 파라메터 설정 창을 통해 컨텍스트 정보의 타입과 값을 설정하는 화면을 보여준다.

시나리오 기반의 컨텍스트 비교 과정에 대한 실험을 위해 본 논문에서는 uWDL 컨텍스트 처리기를 이용하여 센서 네트워크로부터 발생되는 컨텍스트 정보를 (SituationType, presentation), (UserType, Michael), 그리고 (LocationType, 313)과 같은 엔티티들로 생성하였다. 그리고 컨텍스트 비교기는 이 엔티티들과 (그림 8)의 회색부분에 해당하는 DIAST 중에 <constraint> 원소의 <subject>, <verb>, <object> 원소들과 각각 비교한다. 이때 (SituationType, presentation)와 (LocationType, 313) 엔티티 정보는 <constraint> 원소의 <verb> 및 <object> 원소와 각각 일치하지만, (UserType, Michael) 엔티티는 시나리오에 의해 생성된 DIAST의 어떠한 <constratint> 부분트리와도 일치하지 않기 때문에 무시된다. 다시 말해 (UserType, John) 엔티티가 발생할 때까지 컨텍스트 비교기는 현재까지 생성된 엔티티 중에는 시나리오에 기술된 <constaint> 원소와 일치하는 엔티티 정보가 발생하지 않은 것으로 인식하여 시나리오에 기술된 해당 서비스를 실행하지 않는다. 만일 서비스를 실행시키기 위해 (UserType, John) 엔티티 정보를 생성하면 컨텍스 비교기는 컨텍스트 비교 결과로써 (UserType, John) 엔티티가 <subject> 원소와 일

치하여 해당 <constraint> 정보가 현재의 상황과 일치함을 판단하고, 시나리오에 기술된 발표용 원고가 313호에 있는 해당 호스트 컴퓨터로 다운로드되어 실행된다.

6. 관련연구

UBL(Universal Business Language)[15], CC(Core Components) & BIE (Business Information Entity)[16]는 ebXML[17] 프레임워크를 기반으로 기업들 간의 정보 교환을 체계적이면서 상호 운용성을 제공할 수 있도록 정형화된 문서양식을 제공하는 것을 목적으로 한다. 이를 위해서 UML과 같은 디자인 도구를 이용하여 Core Component들을 설계하고 기업 간의 Business Context를 적용하여 BIE를 생성하며, 이를 자동화하기 위해 다양한 형식(XML Schema, DTD 등)으로 변환하여 전자상거래에 활용하게 된다. 그러나 이는 e-Business에 사용될 목적으로 설계되었기 때문에, 유비쿼터스 컴퓨팅 환경에서 발생하는 컨텍스트 정보를 효율적으로 기술하는데 한계가 있다. uWDL은 상황 정보인 컨텍스트와 웹 서비스 형태의 서비스들 간의 연관 관계를 효율적으로 기술하기 위한 언어이다. 또한 컨텍스트 정보는 상호 운용성을 제공하기 위해 온톨로지 개념의 단어 형태로 표현되고, 이 정보는 사용자의 상황에 맞는 서비스를 결정하기 위해 워크플로우의 전이조건으로 사용된다.

기존의 워크플로우 언어인 BPEL4WS[7], WSFL[8], XLANG[9]은 비즈니스 및 분산 컴퓨팅 환경에 적용하기 위한 웹 서비스 기반의 언어들이다. 이 언어들은 서비스의 전이 조건으로 이전 서비스의 결과 값과 서비스 및 시스템의 이벤트 정보를 사용한다. 또한 XPath 기능을 통해 다른 XML 형태의 결과 값을 이용할 수 있다. 그러나 이 언어들을 유비쿼터스 컴퓨팅 환경에 적용하는데는 몇 가지 문제점이 있다. 첫째, 워크플로우가 유비쿼터스적인 서비스를 제공하기 위해서는 서비스의 결과 값이나 단순한 이벤트 정보가 아닌 사용자가 처해있는 환경으로부터 발생하는 상황 정보를 서비스의 전이 조건으로 사용해야 한다. 둘째, XPath를 이용하여 컨텍스트 정보를 얻었을 경우 XPath의 개발 목적상 논리 및 조건 연산자만으로 유비쿼터스 환경에서 발생하는 고수준의 상황 정보를 표현하기가 어렵다. 이를 해결하기 위해 uWDL 언어에서는 컨텍스트 정보를 서비스의 전이 조건으로 명시할 수 있는 지식 기반의 {주어-동사-목적어} 트리플 원소를 추가하였으며, 이를 통해 고수준의 컨텍스트 정보에 따라 사용자에게 적합한 서비스를 자동으로 제공할 수 있다.

7. 결 론

본 논문에서는 사용자에게 상황인지 기반의 서비스를 제공하기 위해 유비쿼터스 컴퓨팅 환경에 워크플로우를 적용하였다. 또한 사용자에게 상황에 맞는 서비스를 제공하기 위해 워크플로우의 상태 전이 조건에 컨텍스트, 프로파일 및 이벤트 정보를 기술할 수 있는 워크플로우 언어인 uWDL (Ubiquitous Workflow Description Language)을 제안하였다. uWDL은 BPEL4WS, WSFL, XLANG 등 기존의 언어들과 같은 웹 서비스 기반의 워크플로우 언어이지만, 이들 언어들이 표현하지 못하는 유비쿼터스 환경의 컨텍스트, 프로파일 및 이벤트 정보를 서비스의 분기조건으로 기술할 수 있는 언어이다. 또한 uWDL은 범용 XML 파서에서 제공할 수 없는 기능을 uWDL 전용 파서와 컨텍스트 처리기를 개발하여 효과적으로 유비쿼터스 환경의 컨텍스트 정보를 기술하여 상황에 맞는 서비스를 선택하는데 사용한다.

유비쿼터스 환경처럼 컴퓨팅 환경뿐만 아니라 센서, 사용자 정보 및 환경으로부터 얻을 수 있는 모든 정보들을 이용하는 서비스들은 다양한 형태의 플랫폼, 프로토콜 및 언어로 개발된다. 이를 통합하기 위해서는 표준화된 인터페이스를 제공하는 웹 서비스 기반의 워크플로우 언어 개발이 요구된다. uWDL은 웹 서비스 기반의 워크플로우 언어이며, 유비쿼터스 환경처럼 이질적이고 다양한 서비스들에 대한 통합, 관리 및 실행을 가능하게 한다.

참 고 문 헌

[1] Merriam-Webster OnLine, http://www.merriam-webster.com.

[2] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Communica- tions of the ACM, Vol.36, No.7, pp.75-84, 1993.

[3] M. Weiser, "The Computer for the 21st Century," Sci. Amer, 1991.

[4] WfMC, "The Workflow Management Coalition Terminology & Glossary-Issue 3.0," WFMC-TC-1011, Workflow Management Coalition, pp.44-49, 1999.

[5] Junwei Cao, Stephen A. Jarvis, Subhash Saini, and Graham R. Nudd. Gridflow: Workflow management for grid computing. In Proceedings of the 3st International Symposium on Cluster Computing and the Grid, page 198. IEEE Computer Society, 2003.

[6] Mack Hendricks, Ben Galbraith, Romin Irani, James Mibery, Tarak Modi, Andre Tost, Alex Toussaint, S. Jeelani Basha, Scott Cable, "Professional Java Web Services," WROX Press, pp.1-16, 2002.

[7] Tony Andrews, Francisco Curbera, Yaron Goland, "Business Process Execution Language for Web Services," BEA Systems, Microsoft Corp., IBM Corp., Version 1.1, 2003.

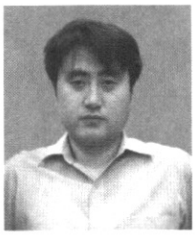
[8] Frank Leymann, "Web Services Flow Language (WSFL 1.0)," IBM Corp., 2001.

[9] Satish Thatte, "XLANG Web Services for Business Process Design," Microsoft Corp., 2001.

[10] W3C, "RDF/XML Syntax Specification," W3C Recommendation, 2004.

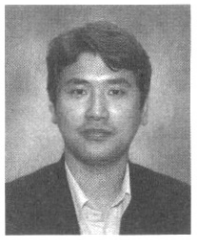
[11] R. Scott Cost, Tim Finin, "ITtalks," A Case Study in the

Semantic Web and DAML+OIL," University of Maryland, Baltimore County, IEEE pp.1094-7167, 2002.
[12] S.Jang, W.Woo, "5W1H: Unified User-Centric Context," The 7th International Conference on Ubiquitous Computing, 2005.
[13] Anind k. Dey, "Understanding and Using Context," Personal and Ubiquitous Computing, Vol.5, Issue 1, 2001.
[14] Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," Pervasive 2002, LNCS 2412, pp.167-180, 2002.
[15] Bil Meadows, Lisa Seaburg, "Universal Business Language 1.0," OASIS Committee Draft, Sep., 2004.
[16] "Core Components Technical Specification V2.01," UN/CEFACT Technical Specification, Nov., 2003.
[17] Anders Grangard, Brian Eisenberg, Duane Nickull, "ebXML Technical Architecture Specification v1.0.4," OASIS, Feb., 2001.



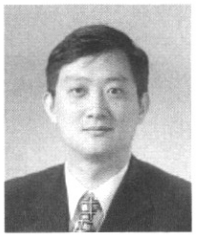
한 주 현

e-mail : jhhan@ss.ssu.ac.kr
2000년 숭실대학교 컴퓨터학부(학사)
2002년 숭실대학교 컴퓨터학과(공학석사)
2002년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 유비쿼터스 컴퓨팅, 분산/병렬처리, 시스템 소프트웨어, 시스템 보안, 한글폰트기술



조 용 윤

e-mail : yycho@ss.ssu.ac.kr
1995년 시립 인천대학교 전자계산학과 (학사)
1998년 숭실대학교 컴퓨터학과(공학석사)
1999년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 유비쿼터스 컴퓨팅, 시스템 소프트웨어, 프로그래밍 언어, 컴파일러, XML, HCI



최 재 영

e-mail : choi@ssu.ac.kr
1984년 서울대학교 제어계측공학과(학사)
1986년 미국 남가주대학교 전기공학과 (공학석사)
1991년 미국 코넬대학교 전기공학부 (공학박사)
1992년~1994년 미국 국립 오크리지연구소 연구원
1994년~1995년 미국 테네시 주립대학교 연구교수
1995년~현재 숭실대학교 컴퓨터학부 부교수
관심분야 : 유비쿼터스 컴퓨팅, 그리드 컴퓨팅, 병렬/분산처리, 클러스터링, 시스템 소프트웨어