유비쿼터스 컴퓨팅 환경을 위한 웹 서비스 기반의 상황인지 워크플로우 언어

(Context-Aware Workflow Language based on Web Services for Ubiquitous Computing Environments)

한 주 현 † 김 은 회 † 최 재 영 †† 조 위 덕 †††
(Joohyun Han) (Eunhoe Kim) (Jaeyoung Choi) (Weduke Cho)

요 약 유비쿼터스 컴퓨팅 환경에서의 서비스들은 사용자가 제공하는 정보뿐만 아니라 사용자가 있는 환경이 제공하는 많은 정보들을 이용하여 동적으로 변화하는 환경에서 사용자에게 적합한 서비스를 자동으로 제공해야 한다. 비즈니스 및 분산 컴퓨팅 환경에서 사용되는 워크플로우는 여러 작업들을 일련의 작업 절차 규칙에 의해 연계시켜 서비스의 자동화를 지원한다. 따라서 서비스의 자동화에 사용되어 왔던 워크플로우를 유비쿼터스 컴퓨팅에 적용하여 상황인지 서비스를 지원하기 위해서는 컨텍스트 정보를 서비스의 전이조건으로 명시해야 한다. 이를 위하여 본 논문에서는 워크플로우를 유비쿼터스 컴퓨팅에 적용하고 워크플로우의 상태 전이 제약조건에 컨텍스트 정보를 명시하기 위한 구조적인 컨텍스트 모델을 제시한다. 또한 이 구조적인 컨텍스트 모델을 사용하기 위해서, 이질적이면서 다양한 플랫폼, 프로토콜 및 언어에 독립적인 표준화된 인터페이스를 제공하는 웹 서비스 기반의 워크플로우 언어인 uWDL을 제안한다. uWDL은 유비쿼터스 환경의 워크플로우 엔진에 의해 해석되고 실행되어 사용자에게 상황인지 기반의 자동화된 서비스를 제공한다.

키워드 : 유비쿼터스 컴퓨팅, 상황인지, 웹 서비스, 워크플로우

Abstract The services for ubiquitous computing environments have to provide automatically user-specific adaptive services in dynamically changed environments with many informations provided by both a user and his/her environment. Workflows used in business and distributed computing environments support service automation by connecting many tasks with rules or orderings of tasks. Therefore we must specify context information on transition condition to support context-aware services by adapting a workflow to ubiquitous computing environments. In this paper, we present a structural context model to specify context information on transition constraints of the workflow. And we propose an uWDL (Ubiquitous Workflow Description Language) based on web services, which provides web service interfaces which are standardized and independent on heterogeneous and various platforms, protocols, and languages. The uWDL can be interpreted and executed by a workflow engine, and provide users autonomic services based on context-awareness.

Key words : Ubiquitous Computing, Context Awareness, Web Services, Workflow

· 본 논문은 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크 원천기반기술개발사업의 지원으로 연구되었습니다(05A3-F1-10주).

† 비 회 원 : 숭실대학교 컴퓨터학과
jhhan@ss.ssu.ac.kr
ehkim@ss.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
choi@ssu.ac.kr
††† 종신회원 : 아주대학교 전자공학부 교수, 유비쿼터스컴퓨팅사업단 단장
chowd@ajou.ac.kr

논문접수 : 2004년 5월 3일
심사완료 : 2005년 8월 10일

1. 서 론

유비쿼터스는 "언제 어디에든 존재한다[1]"라는 의미의 라틴어로 1988년 처음으로 제록스 팔로 알토(Palo Alto) 연구소의 마크 와이저(Mark Weiser)에 의해 소개되었다. 그는 유비쿼터스 컴퓨팅을 "현실 세계의 어디서나 네트워크와 연결된 컴퓨터를 눈에 띄지 않는 형태로 언제 어디서나 사용 가능하며 사용자 상황에 따라 서비스가 변하는 환경[2]"이라고 정의하였다. 이것은 '컴퓨팅 환경이 현실 세계의 사물과 환경 속으로 스며들어

일상생활에 통합되는 것'[3]을 의미한다. 이러한 유비쿼터스 컴퓨팅 환경에서는 기존의 분산 컴퓨팅 환경과 달리 환경에 대한 지각 능력이 필요하다. 이를 위해 환경을 센싱하여 사용자, 디바이스, 사물 등의 위치, 상태, 행동과 같은 컨텍스트 정보를 얻고, 유비쿼터스 환경의 서비스들은 이러한 정보를 바탕으로 사용자에게 가장 적절한 상황인지 서비스를 제공해야 한다.

비즈니스 및 분산 컴퓨팅 환경의 서비스들은 서로 연계된 작업의 흐름을 가지는 워크플로우 모델을 가진다. 워크플로우 모델은 일련의 작업 절차 규칙에 의해 서비스의 자동화를 지원한다. 유비쿼터스 환경의 사용자들은 동적으로 변화하는 환경에서 사용자가 원하는 서비스들을 사용자가 원하는 형태로, 사용자가 원하는 시간에, 사용자의 직접적인 개입 없이 서비스받기를 원한다[4,5]. 유비쿼터스 컴퓨팅 환경에서 이러한 사용자의 요구 조건을 만족하는 서비스 자동화를 구축하기 위해서는 비즈니스와 분산 컴퓨팅 환경의 서비스 자동화에 적용하던 워크플로우 모델을 유비쿼터스 컴퓨팅 환경에 적용해야 할 필요성이 있다. 유비쿼터스 컴퓨팅 환경을 위한 워크플로우의 요구사항은 크게 두 가지로 분류할 수 있다. 첫째, 유비쿼터스 환경의 워크플로우는 동적으로 변화하는 환경에서 사용자 상황에 맞는 적합한 서비스를 제공해야 한다. 둘째, 유비쿼터스 환경은 다양한 이질적인 플랫폼, 프로토콜, 응용 프로그램으로 구성되어 있으므로, 이러한 환경의 응용 프로그램들을 통합, 관리, 실행하기 위해서는 플랫폼과 언어에 독립적인 표준 서비스 인터페이스가 요구된다[6].

본 논문에서는 워크플로우가 주변 상황을 스스로 인지하여(Context Awareness), 상태를 전이할 수 있도록(Transition Condition) 워크플로우의 상태 전이 제약 조건에 컨텍스트를 명시하기 위한 구조적 컨텍스트 모델(Structural Context Model)을 정의하고, 이를 기반으로 하는 워크플로우 언어인 uWDL(Ubiquitous Workflow Description Language)을 제안한다. 구조적인 컨텍스트 모델은 유비쿼터스 컴퓨팅 환경에서 발생하는 컨텍스트 정보를 지식 구조 관점에서 표현하며, 복잡한 컨텍스트 정보들을 기술할 수 있는 정보 구조를 지원하기 때문에 워크플로우 언어에서 상태 전이 제약 사항을 명시할 때 풍부하게 컨텍스트를 표현할 수 있다. 또한 uWDL은 웹 서비스의 표준화된 서비스 인터페이스를 기반으로 하기 때문에 유비쿼터스 환경처럼 이질적이고 다양한 플랫폼에서 동작하는 응용 프로그램들을 통합, 관리 및 실행하기에 적합하다.

논문의 구성은 다음과 같다. 2장에서는 유비쿼터스 컴퓨팅 환경에서 워크플로우 언어가 컨텍스트 모델을 필요로 하는 이유에 관해서 설명한다. 3장에서는 유비쿼터스 환경의 워크플로우에서 컨텍스트를 표현하기 위한 구조적인 컨텍스트 모델을 제안한다. 4장에서는 3장에서 제시한 구조적인 컨텍스트 모델을 이용하여 웹 서비스 기반의 워크플로우 언어(uWDL)을 설계한다. 5장에서는 유비쿼터스 환경의 사무실에서 회의 준비 시나리오를 uWDL을 이용하여 작성함으로써 본 논문에서 제시하는 구조적인 컨텍스트 모델이 어떻게 적용되는지를 설명한다. 6장에서는 관련연구와 본 연구의 상황인지 워크플로우를 비교 분석한다. 7장에서는 본 논문의 결론을 내린다.

2. 관련연구

현재까지 유비쿼터스 환경을 위한 응용 프로그램 중에 상황인지 기능을 수행하는 많은 연구들이 진행되고 있다. PARCTAB[16], Aura[17], One.world[18] 등과 같은 상황인지 시스템의 응용 프로그램들은 if-then의 규칙들을 이용하여 context-triggered action을 정의하였으며, 실행 시간에 publish-subscribe 메커니즘을 이용한 이벤트 기반구조의 지원을 받아 context-triggered action들이 실행되어 상황인지 서비스를 지원하게 된다. 그러나 이러한 연구들은 하나의 서비스 내에서 사용자의 상황을 인식하기 위한 연구들이고, 서로 다른 서비스들의 흐름에 대한 연구는 아니었다. 유비쿼터스 환경에는 서로 다른 목적의 수많은 서비스들이 존재하고, 그것들은 상황 정보에 따라 선택, 반복 혹은 병렬적으로 수행되어야 한다. uWDL은 워크플로우 언어이면서 주어-동사-목적어 기반의 context-triggered action을 지원하기 때문에, 상황 정보에 따라 해당 서비스를 선택, 반복, 병렬적으로 수행할 수 있다.

Gaia[19]에서는 유비쿼터스 응용 프로그램들이 컨텍스트 정보를 교환하면서 통신할 수 있는 서비스 환경을 지원한다. 그러나 코바 기반이기 때문에 특정 프로토콜에 의존적인 단점을 가지고 있다. 그러나 본 논문에서 제안한 유비쿼터스 환경의 워크플로우는 웹 서비스 기반이므로 플랫폼, 언어에 독립적인 특성을 제공한다. 또한 Gaia의 스크립팅 언어인 LuaOrb는 유비쿼터스 환경의 컨텍스트를 기술하고 프로그램하는 기능을 제공하지만 프로그램을 초기화하고 인스턴스화하는 과정이 순차적으로 기술되어 있어, 워크플로우를 이용해 표현할 수 있는 의존성, 순서성, 동시성을 표현할 수 없다는 제약이 있다.

기존의 워크플로우 언어인 BPEL4WS[9], WSFL[10], XLANG[15]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 기능

을 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.

3. 유비쿼터스 기반의 워크플로우 언어를 위한 요구사항

유비쿼터스 컴퓨팅 환경에서 서비스들은 컨텍스트를 통하여 상황을 파악하고 적절한 서비스를 선택하게 된다. 센서로부터 발생하는 컨텍스트는 저수준의 정보만을 가지고 있기 때문에 응용 프로그램에서 바로 사용하기에는 적절치 않다. 따라서 저수준 컨텍스트는 온톨로지 기반의 처리 과정을 거쳐 고수준의 컨텍스트로 변환되어야 한다. 그림 1은 유비쿼터스 환경에서 웹 서비스 기반 워크플로우 언어들이 서비스 전이 조건을 위해 온톨로지에 의해 번역된 고수준의 컨텍스트 정보를 이용하는 과정을 나타낸다.

DAML+OIL[7] 및 OWL[8]과 같은 온톨로지 언어들은 저수준의 컨텍스트를 온톨로지 기반의 컨텍스트로 표현하고, 추론 엔진에 의해 번역되어 고수준의 컨텍스트 정보인 시멘틱 컨텍스트(Semantic Context)로 변환된다. 이와 같은 고수준 컨텍스트 정보를 웹 서비스 기반의 워크플로우 언어에서 서비스의 전이 조건으로 사용하기 위해서는 다음과 같은 요구 조건이 필요하다. 첫째, 온톨로지 기반의 추론 엔진을 통해 생성된 시멘틱 컨텍스트 정보를 표현할 수 있는 컨텍스트 모델이 필요하다. 컨텍스트 모델은 추론을 통해 생성된 고수준의 컨텍스트 정보와 이를 사용하는 워크플로우 및 응용 프로그램간의 인터페이스 역할을 한다. 즉, 워크플로우 언어는 컨텍스트 모델을 통해서 고수준의 컨테스트 정보를 얻게 된다. 둘째, 워크플로우 언어들은 컨텍스트 모델에 의해 기술된 컨텍스트 정보를 이용할 수 있는 원소(element)를 포함해야 한다. 현재 BPEL4WS[9] 및 WSFL[10]에서는 XML 문서 사이의 정보 공유 및 전이 조건(transition condition)을 위해 XML 링크 기능(XPath, XLink, XPointer)을 사용한다. 일반적으로 XML 링크 기능은 하나의 XML 문서 내에서 사용된 XML 링크 포인트를 다른 XML 문서에 대한 트리 형태의 구조 정보를 통해 특정 위치의 원소에 접근한다. 그러나 XPath를 사용하여 얻을 수 있는 정보는 특정 원소의 값(value)이며, 이와 같은 단순 정보를 이용하여 고수준의 컨텍스트를 표현하거나 서비스의 전이 조건으로 사용하는데는 한계가 있다. 유비쿼터스 환경에서 발생하는 컨텍스트 정보는 특정 엔티티(entity)들의 관계에 의해 표현되며, 또한 엔티티의 값(value)뿐만 아니라 엔티티의 타입(type) 정보 또한 중요하게 작용된다. 예를 들어 "철수가 학교에 간다"라는 의미정보를 갖는 고수준의 컨텍스트 정보가 있을 경우, 철수라는 값이 사람을 의미할 수도 있지만 강아지 이름일 수 도 있다. 이를 구별하기 위해서는 철수라는 값의 타입이 'Person' 타입인지 'Dog' 타입인지를 구별할 수 있는 타입 정보가 필요하다. 따라서 기존의 워크플로우 언어는 온톨로지 기반의 컨텍스트 정보를 이용할 수 있는 컨텍스트 모델이 필요하며, 이를 전이 조건에 사용하기 위한 원소가 요구된다.

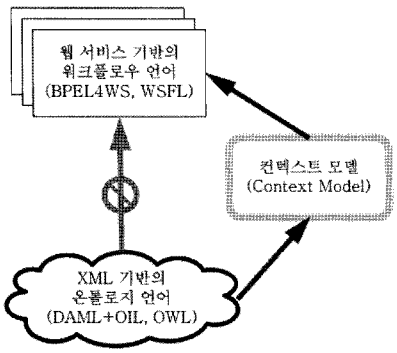


그림 1 워크플로우 언어에서 고수준 컨텍스트 정보를 이용하는 과정

4. 유비쿼터스 환경에서의 워크플로우를 위한 구조적인 컨텍스트 모델

4.1 유비쿼터스 환경에서 워크플로우를 위한 컨텍스트

유비쿼터스 환경에서 컨텍스트란 "사용자 또는 응용 프로그램과 상호 작용을 갖는 사람, 장소, 사물에 대한 상황 또는 상태 정보[11]"를 의미한다. 이러한 컨텍스트 정보를 사용하거나 컨텍스트에 알맞은 기능을 수행하는 응용 프로그램이나 시스템을 상황인지 응용 프로그램(context-aware application) 또는 상황인지 시스템(context-aware system)이라고 한다[5,8]. 유비쿼터스 환경의 모든 서비스는 사용자의 상황에 맞는 가장 적절

한 서비스를 제공하기 위하여 상황인지 서비스(context-aware services)를 지향한다.

일반적으로 워크플로우에서는 세 가지 종류의 데이타가 사용된다. 데이타 종류로는 워크플로우 제어 데이타, 응용 프로그램 데이타, 워크플로우 관련 데이타가 있다. 워크플로우 제어 데이타는 워크플로우 관리 시스템이나 엔진이 내부적으로 사용하는 데이타이다. 예를 들면, 각 서비스의 상태 정보나 아이디(ID), 서비스 결함이 발생한 지점과 복구지점 등과 같은 정보이다. 응용 프로그램 데이타란 워크플로우 서비스에서 사용하는 데이타이다. 워크플로우 관련 데이타는 워크플로우의 상태 전이를 결정하기 위해서 사용하는 데이타를 말한다. 워크플로우의 상태 전이란 한 서비스가 완전히 끝나고 제어권을 다른 서비스에 넘기는 것을 말하며 상태 전이 조건에 의해 다음 서비스가 결정된다[12].

유비쿼터스 환경의 워크플로우가 상황인지 서비스를 제공하기 위해서는 컨텍스트 정보에 따라 적절한 서비스가 선택되어 실행되어야 한다. 즉 워크플로우 서비스들은 유비쿼터스 환경의 컨텍스트 정보를 이용해 명시한 조건에 따라 선택적으로 실행되거나 병렬적으로 혹은 반복적으로 실행되기도 하며, 서비스 순서에 영향을 미치는 시작과 종료 조건이나 데드라인 같은 제약사항 등에 컨텍스트 정보를 명시해야 한다. 따라서 워크플로우의 상태 전이 조건에 유비쿼터스 환경의 컨텍스트 정보를 표현하기 위한 구조적인 컨텍스트 모델을 그림 2와 같이 제안한다.

4.2 워크플로우의 상태 전이 조건에 사용되는 구조적인 컨텍스트 모델

구조적인 컨텍스트 모델은 온톨로지 기반의 고수준 컨텍스트 정보를 워크플로우에서 사용할 수 있도록, 컨텍스트의 의미 정보를 표현하는 단순한 기술 방법인 RDF(Resource Description Framework)[13]를 이용하여 구조적으로 표현한다. 이와 같이 표현된 구조적인 컨텍스트 모델은 워크플로우가 컨텍스트 정보에 따라 적절한 서비스를 선택할 때, 보다 고수준의 상황 정보를 제공한다. 또한 온톨로지와 워크플로우라는 서로 다른 목적과 형태의 시스템을 연결하는 인터페이스 역할을 한다. 다음은 구조적인 컨텍스트 모델을 구성하고 있는 주요 요소들을 설명하고 있다.

4.2.1 컨텍스트와 엔티티

유비쿼터스 환경의 워크플로우에서 사용하는 컨텍스트의 타입은 워크플로우 도메인에 따라서 그 종류가 서로 다르며 매우 다양하다. 일반적으로 사용되는 컨텍스트의 타입은 크게 시간, 위치, 아이덴티티(Identity), 엑티비티(Activity) 등으로 분류할 수 있으며, 컨텍스트 정보의 주체를 엔티티(Entity)라고 한다. 유비쿼터스 환

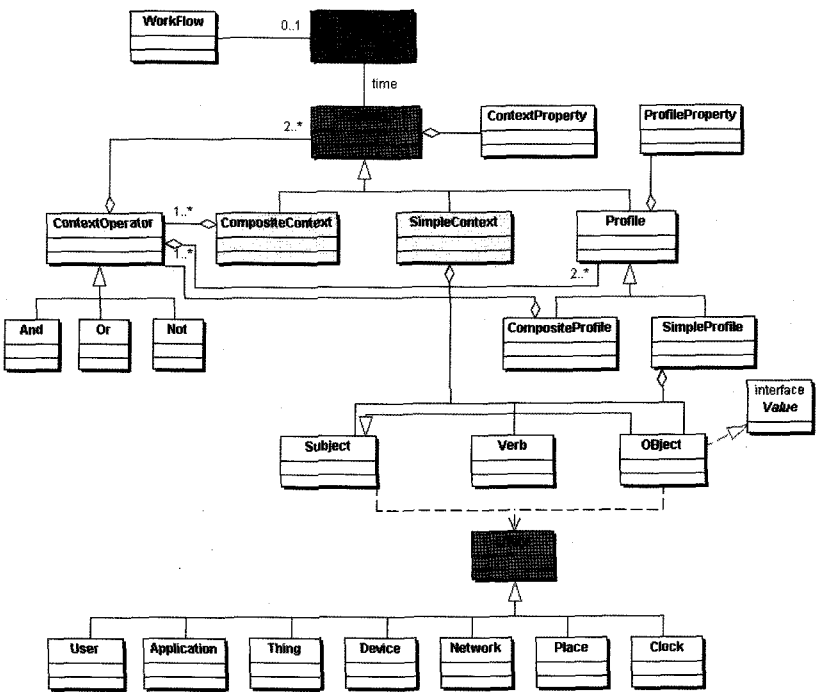


그림 2 워크플로우를 위한 구조적인 컨텍스트 모델

경에서 모든 객체는 모두 엔티티가 될 수 있고, 엔티티의 종류는 워크플로우의 도메인에 따라 좌우된다. 예를 들면, 사람, 응용 프로그램, 사물, 디바이스, 네트워크, 장소, 시스템, 시계 등, 인간의 삶속의 모든 환경을 구성하는 유형 또는 무형의 객체들이 컨텍스트 정보를 위한 엔티티에 해당한다. 이러한 엔티티들로부터 사람의 위치 정보, 사람의 행동 정보, 디바이스들의 상태 정보, 근접해 있는 사물 정보, 스케줄 정보, 컴퓨팅 장비에서 실행되는 응용 프로그램 정보, 네트워크의 대역폭 정보, 온도, 조명의 밝기, 소음 정보 등의 컨텍스트 정보를 얻는다. 이러한 컨텍스트 정보는 물리적인 환경의 엔티티를 센싱함으로써 얻거나 또는 프로파일 정보로부터 얻을 수 있다.

프로파일 정보는 엔티티에 대한 명확하게 주어진 정보를 말한다. 환경을 센싱하여 얻는 컨텍스트 정보는 시간에 따라 매우 가변적인 특성이 있는 반면, 프로파일 정보는 변동이 없거나 변동 주기가 매우 느리다[14]. 특히 워크플로우에서는 사용자의 스케줄에 따라 기본적인 워크플로우를 결정하게 되고, 워크플로우가 참조해야 할 사용자의 기본적인 기호 정보 등이 필요하므로 이러한 정보들을 프로파일 정보로 제공한다.

4.2.2 단순 컨텍스트와 합성 컨텍스트

컨텍스트 정보는 단순한 환경에 대한 사실을 표현하는 저수준의 컨텍스트로 부터, 이러한 저수준의 컨텍스트를 조합하고 인공지능적인 처리 과정을 거쳐 사용자의 의도와 같은 고수준의 컨텍스트 정보까지 그 수준이 매우 다양하다. 유비쿼터스 환경의 컨텍스트 정보들을 실제로 구현할 때 온톨로지를 사용한다[8]. 대부분의 온톨로지 언어들은 RDF에 기반을 둔다. RDF는 자원에 대한 메타 정보를 기술하기 위한 언어로서, '주어-동사-목적어' 트리플의 형식으로 자원을 표현한다. 기본적으로 단순한 컨텍스트 정보와 단순한 프로파일 정보는 RDF로 기술하고, 합성 컨텍스트를 표현할 때는 온톨로지의 구문을 이용한다. 또한 자원을 서술할 때 사용하는 '동사'들도 컨텍스트 타입에 따라 다양하므로 온톨로지를 이용하여 그 단어와 의미들을 명확하게 정의하여 사용한다.

본 논문에서는 구조적인 컨텍스트 모델은 컨텍스트 정보를 단순한 정보 표현도 가능하고, 인공지능적인 처리도 가능한 가장 단순한 표현 방법인 '주어-동사-목적어'로 표현한다. 또한 엔티티의 상태나 상황을 설명하는 지식으로서 컨텍스트를 표현한다. 예를 들면 온도의 타입을 가진 컨텍스트를 표현할 때 그 컨텍스트의 타입은 온도이며 컨텍스트 정보는 "301호는 20도이다"라고 표현한다. 또한 엑티비티 타입을 컨텍스트로 표현할 때, 그 컨텍스트 정보는 "앤이 프리젠테이션을 하고 있다"와 같이 표현한다. 이러한 '주어-동사-목적어'로 구성된 컨텍스트 정보를 단순 컨텍스트라고 한다.

컨텍스트 정보에서 '주어-동사-목적어'로 표현이 불가능한 컨텍스트 정보가 있다. 예를 들면 "앤이 301호에서 프리젠테이션을 하고 있다"는 컨텍스트 정보는 '주어-동사-목적어'로 표현이 불가능하다. 이러한 컨텍스트 정보를 합성 컨텍스트라고 하며, 단순한 컨텍스트 정보를 and, or, not 연산자로 묶어서 표현한다. 위의 켄텍스트 정보는 and 연산자를 이용하여 "앤이 301호에 있다. 그리고 앤이 프리젠테이션하고 있다"로 표현한다.

4.2.3 컨텍스트 히스토리

워크플로우 시스템은 해당 도메인에 따라 다양한 워크플로우들을 가진다. 사무실의 워크플로우 시스템을 예로 들면, 회의 워크플로우, 출장 예약 워크플로우, 자료 검색 워크플로우, 고객 면담 워크플로우 등과 같은 다양한 워크플로우가 존재할 수 있다. 따라서 컨텍스트 정보는 각 워크플로우마다 따로 구분하여 관리되어야 한다. 또한 컨텍스트 정보는 시간을 기준으로 과거, 현재, 미래의 컨텍스트 정보로 구성된다. 과거 컨텍스트 정보는 추론이나 학습, 데이타 마이닝 같은 처리를 통해 고수준의 컨텍스트를 유추하기 위하여 사용된다. 현재 컨텍스트 정보는 워크플로우의 상태 전이에 가장 직접적인 영향을 미치는 요소이다. 미래 컨텍스트 정보는 워크플로우의 각종 장비, 장소 등의 예약 기능을 지원하기 위하여 사용된다. 워크플로우를 위한 구조적인 컨텍스트 모델에서는 시간에 따른 과거, 현재, 미래 컨텍스트를 통칭하여 컨텍스트 히스토리라고 한다.

5. 유비쿼터스 환경을 위한 웹 서비스 기반의 상황인지 워크플로우 언어

현재의 워크플로우 언어들은 웹 서비스를 기반으로 서로 다른 서비스 간에 발생하는 데이타의 흐름을 명시하고 있다. 웹 서비스 기반의 워크플로우 언어인 WSFL (Web Services Flow Language)은 IBM에서 독자적으로 개발하고 있는 언어로, WSDL(Web Services Definition Language)을 기반으로 서비스들의 연관 관계를 명시할 수 있다. 또한 Microsoft, BEA와 함께 BPEL4WS로의 표준화 작업이 진행되고 있다[10]. Microsoft는 웹 서비스 사이에서 발생하는 메시지 교환의 행동(behavior)을 표현하기 위해서 XLANG(Web Services for Business Process Design)을 제안하였다. XLANG은 프로세스 인스턴스들의 상태를 추적할 수 있고, 메시지의 흐름에 있어서 프로토콜의 정확성을 증대시키기 위한 자동화 프로토콜 엔진 기반의 서비스를 제공한다[15].

그러나 이와 같은 워크플로우 언어들은 센서로부터

발생되어 온톨로지에 의해 번역된 고수준 컨텍스트 정보를 이용하여 서비스를 선택할 수 있는 원소를 포함하고 있지 않다. BPEL4WS와 WSFL에서 전이 조건으로 사용하고 있는 transitionCondition 속성은 XPath라는 XML 링크 기능을 사용하여 특정 서비스의 입/출력 메시지의 값에 따라 서비스의 분기를 결정한다. 그러나 컨텍스트 정보는 특정 서비스의 입/출력 정보와는 무관하게 센서로부터 발생하기 때문에 기존의 워크플로우 언어를 통해서는 유비쿼터스 환경의 컨텍스트를 서비스의 전이 조건으로 사용하는데 한계가 있다. 이를 위해 본 논문에서는 기존의 웹 서비스 기반의 워크플로우 언어들의 기능을 간소화하면서, 유비쿼터스 환경에서 발생하는 컨텍스트 정보를 서비스의 전이 조건으로 사용할 수 있는 언어인 uWDL(Ubiquitous Workflow Description Language)을 제안한다. uWDL은 구조적인 컨텍스트 모델을 기반으로 컨텍스트를 표현할 수 있는 원소를 추가하므로써 효과적으로 서비스의 흐름 관계를 기술할 수 있는 언어로, 센서에 의해 감지되는 정보들이 온톨로지에 의해서 고수준의 컨텍스트, 프로파일 및 이벤트 등의 정보로 제공되면, 이와 같은 정보를 기반으로 서비스의 흐름을 결정할 수 있다. 또한 온톨로지 형태의 추론 서비스를 이용하기 위한 지식 기반의 서술 기능도 포함하고 있다. 그림 3은 uWDL 스키마 구조를 나타낸다.

uWDL은 크게 <source>, <sink>, <node>, <link> 원소로 구성된다. <source>와 <sink>는 워크플로우의 시작과 끝을 나타낸다. <node> 원소는 uWDL이 웹 서비스 기반의 언어이기 때문에 이 원소를 이용하여 유비쿼터스 환경의 웹 서비스를 가리키며, 이는 웹 서비스의 하나의 오퍼레이션(operation)에 해당된다. <link> 원소는 서비스간의 흐름을 결정하는 하위 원소인 <condition>을 가지며, 이는 다시 <context>, <profile> 및 <event> 등의 하위 원소를 통해서 유비쿼터스 환경에서 발생하는 컨텍스트, 프로파일 및 이벤트 등을 표현한다. 또한 <link> 원소의 하위 원소인 <export>의 속성 값에 따라 "Control"과 "Data" 링크로 구분되며, 각각은 서비스의 흐름과 데이타의 이동을 명시한다.

5.1 <node> 원소

<node> 원소 그림 4의 EBNF(Extended Backus Naur Form) 표기법은 표 1과 같다.

<node> 원소는 웹 서비스 기반의 유비쿼터스를 위한 서비스들 중에 하나의 기능을 제공하는 오퍼레이션을 지칭한다. 웹 서비스는 자신의 프로그램 또는 서비스를

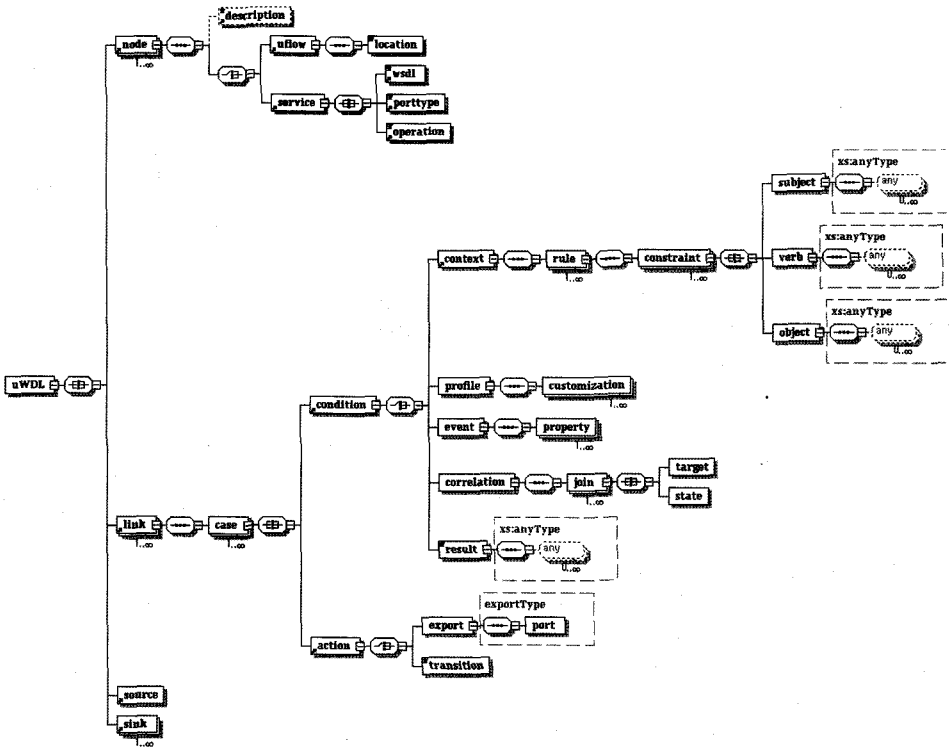


그림 3 uWDL 스키마 구조

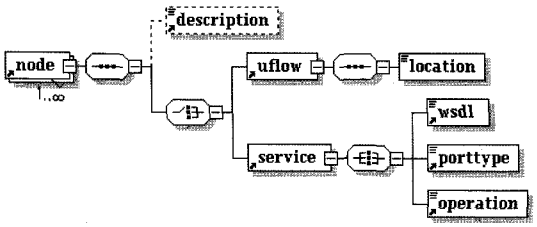


그림 4 <node> 원소

표 1 <node> 원소의 EBNF 표기법

```
Node        ::= Description?, (Uflow | Service)
Uflow       ::= Location
Service     ::= Wsdl, Porttype, Operation
```

명시하기 위해서 WSDL을 이용하여 자신의 PortType 및 Operation 등의 구체적인 서비스를 명시하게 되고, 이는 <service> 원소의 <wsdl>, <porttype>, 그리고 <operation> 원소를 통해서 원하는 서비스의 위치, 서비스 타입 및 오퍼레이션에 대한 정보를 기술하게 된다. 또한 <uflow> 원소는 uWDL로 기술된 다른 워크플로우 기술 문서를 가리킨다.

5.2 〈link〉 원소

<link> 원소(표 2, 그림 5)는 uWDL 언어의 가장 중요한 부분으로 유비쿼터스 환경에서 발생할 수 있는 컨텍스트, 프로파일 및 이벤트 등을 명시하며, 이에 따라 서비스의 흐름이 변화되도록 정의한다. <link> 원소는

표 2 <link> 원소의 EBNF 표기법

```
Link        ::= Case+
Case        ::= Condition, Action
Condition   ::= Context | Profile | Event | Correlation | Result
Context     ::= Rule+
Rule        ::= Constraint+
Constraint  ::= Subject, Verb, Object
Profile     ::= Customization+
Event       ::= Property+
Correlation ::= Join+
Join        ::= Target, State
Action      ::= Export | Transition
Export      ::= Port
```

크게 <condition>과 <action> 원소로 구분된다. <condition> 원소는 <context>, <profile> 및 <event> 등을 통하여 해당 노드의 컨텍스트, 프로파일 및 이벤트의 상태를 명시하며, 상태가 참(true)일 경우 <action>에서 명시하는 행동을 수행한다. <action> 원소는 <export>와 <transition>으로 구분되며, <export>의 속성(attribute)을 통해 제어 링크(control link)와 데이타 링크(data link)로 나뉜다. 그리고 <transition>은 현재 노드의 상태(state) 변화를 명시하기 위해서 사용된다.

<condition> 원소는 컨텍스트, 프로파일 및 이벤트 정보 등에 따라 적절한 서비스가 선택되도록 결정해주는 역할을 한다. 원소 구성요소 중에 중요한 원소로는 <context>와 <profile> 원소를 들 수 있다. <context> 원소는 온톨로지 및 추론 서비스에 의해 생성된 고수준 컨텍스트(High-level context) 정보를 구조적 표기법을 이용하여 명시할 수 있는 <constraint> 원소를 포함한

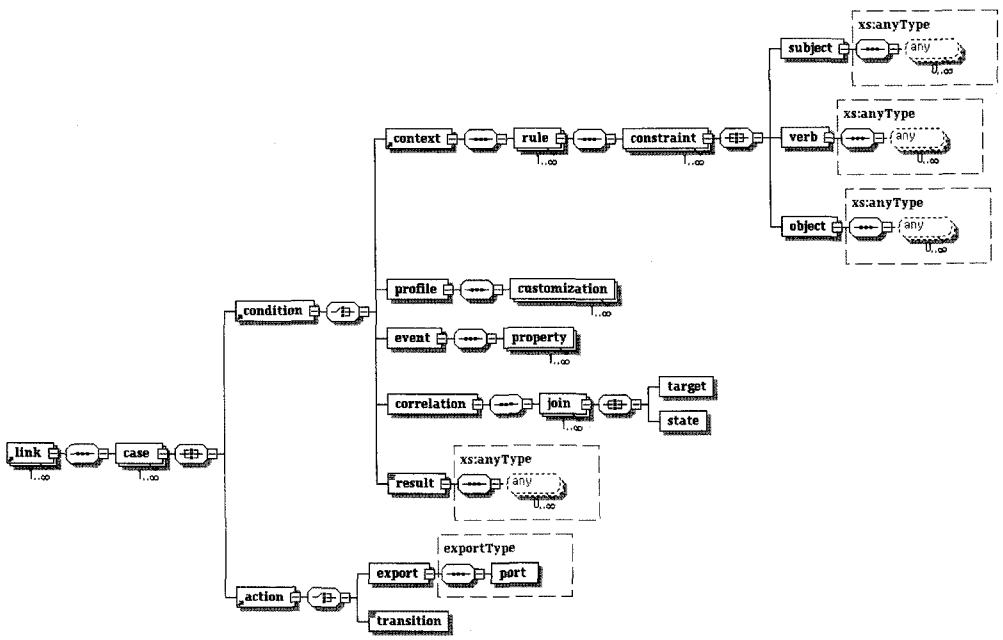


그림 5 <link> 원소

표 3 <constraint> 원소를 이용한 컨텍스트 표현

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

다. <constraint> 원소는 하위 원소로 구조적인 컨텍스트 모델에 기반을 둔 주어(subject), 동사(verb), 목적어(object)를 가지며, 표 3과 같이 "A의 위치가 301호이다"를 표현할 때 사용된다.

<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. <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 사용된다는 것을 알 수 있다. <event> 원소는 컨텍스트 및 프로파일의 상태변화를 처리하기 위해 사용되며 changeOfLocation, changeOfProfile 등과 같이 표현된다. 또한 다른 서비스와의 협업을 위한 <corelation> 원소를 가지고 있다.

6. uWDL에서 컨텍스트 비교를 위한 알고리즘

유비쿼터스 컴퓨팅 환경에서 존재하는 서비스가 사용자의 상황 정보에 따라 전개되기 위해서는 센서 네트워크에서 발생되는 상황 정보와 서비스 시나리오에 기술된 상황 정보와의 비교 방법이 요구된다. 이와 같은 서비스 분기를 자동으로 처리하기 위해서는 uWDL로 기술된 시나리오 문서를 파싱하여 생성한 DIAST(Document Instance Abstract Syntax Tree)[20,21] 형태의 컨텍스트 정보와 유비쿼터스 미들웨어로부터 전달되어 구조적 컨텍스트 모델 형태로 변형된 컨텍스트 정보를 비교해야 한다. DIAST는 uWDL로 기술된 시나리오 문서의 문법 구조를 나타내며, 구조적 컨텍스트 모델 형태의 컨텍스트 정보와 시나리오에서 기술하고 있는 컨텍스트 정보를 비교하기 위해 사용되는 자료구조이다.

구조적 컨텍스트 모델 및 DIAST에서 표현하고 있는 컨텍스트는 {주어, 동사, 목적어} 형태의 엔티티들에 대한 집합으로 표현할 수 있다. 즉, 센서 네트워크로부터 제공되는 컨텍스트 정보는 구조적 컨텍스트 모델 형태의 {주어, 동사, 목적어}로 구성된 엔티티들에 대한 집합으로 표현할 수 있으며, DIAST에서 <constraint> 원소로 표현되는 컨텍스트 정보 또한 {주어, 동사, 목적어} 형태의 엔티티들에 대한 집합이다. 따라서 본 논문에서는 센서 네트워크로부터 전달된 컨텍스트 정보가 구조적 컨텍스트 모델로 구체화된 컨텍스트를 OC = {(OCs_type, OCs_value), (OCv_type, OCv_value), (OCo_type, OCo_value)}로 표현하며, DIAST에서 표현하고 있는 컨텍스트는 UC = {(UCs_type, UCs_value), (UCv_type, UCv_value), (UCo_type, UCo_value)}로 표현한다. 다시 말해서, OC는 구조적 컨텍스트 모델로 표현된 컨텍스트를 의미하며, OCs, OCv, OCo는 각각 주어, 동사, 목적어를 의미하는 엔티티 정보이다. UC는 uWDL로 기술된 시나리오상의 컨텍스트를 의미하며, UCs, UCv, UCo는 각각 주어, 동사, 목적어 형태로 표현된 엔티티 정보이다. 엔티티는 컨텍스트의 타입과 값 정보를 위해 쌍으로 나타내며, OC들과 UC들의 집합 OCS는 OCS = {OC1, OC2, OC3, …, OCi}와 UCS = {UC1, UC2, UC3, …, UCi}로 표현될 수 있다. 표 4는 UC와 OCS를 이용하여 서비스 분기를 결정하기 위한 컨텍스트 비교 알고리즘의 일부분이다. 이는 uWDL로

표 4 UC A와 OCS B를 비교하는 알고리즘

```
Boolean MatchContext(UC A, OCS B) {
  int i; /* For the index of context in A, B each context set */
  /* Repeatedly comparing contexts in A, B context set */
  for each i in OCS B {
      if ((A.UCs_type == Bi.OCs_type && A.UCs_value == Bi.OCs_value) &&
          (A.UCv_type == Bi.OCv_type && A.UCv_value == Bi.OCv_value) &&
          (A.UCo_type == Bi.OCo_type && A.UCo_value == Bi.OCo_value)){
            return True;          /* Found context match */
      }
  } /* End for */
  return False;    /* Return matchresult */
}
```

기술된 컨텍스트 정보인 A와 센서로부터 전달된 컨텍스트 정보가 구조적 컨텍스트 모델로 객체화된 B 집합간의 타입과 값을 {주어, 동사, 목적어}별로 비교하여 상황과 일치하는지의 여부를 판단한다.

7. 시나리오를 통한 검증

이제까지 소개한 uWDL은 유비쿼터스 환경의 모든 서비스들을 컨텍스트 및 프로파일 정보에 따라 서비스를 결정할 수 있는 워크플로우 언어이다. 또한 웹 서비스 기반의 언어이기 때문에 언어, 프로토콜 및 플랫폼에 상관없이 서비스들을 사용할 수 있다는 장점이 있다. 이와 같은 특징을 검증하기 위해 유비쿼터스 환경의 서비스들 중에서 회의 준비를 위한 시나리오를 구상하였다. 시나리오의 목적은 "자신의 일정 계획에 따라 회의를 자동으로 준비해 주는 서비스 구축"이며, 시나리오는 다음과 같다. "회사원 A씨는 출근 후 자신의 노트북을 이용하여 일정 계획에 아침 10시에 301호 회의실에서 발표가 있음을 기록한다. 발표 준비를 하던 A씨는 9시 40분경에 회의를 위해 301호 회의실로 이동한다. 301호 회의실 문 위쪽에는 RFID 센서가 설치되어 있어 A씨의 기본 정보(이름, 노트북 IP 주소 등)를 서버로 전송한다. 서버에서는 전송된 A씨의 IP 주소를 이용하여 일정 계획 서비스로부터 일정 계획 정보를 획득한다. 그리고 현재 시간(Time), 위치 정보(Location) 및 현재 상황(Situation) 등을 비교하여 참(true)일 경우 A씨의 발표 자료를 다운로드하여 해당 프로그램으로 실행한다."

시나리오 개발자는 설계하고자 하는 내용을 uWDL 편집기를 이용하여 기술하게 된다. 이때 상황 정보인 컨텍스트 및 프로파일에 따라 어떤 서비스를 선택할지를 결정해야 한다. 이는 uWDL의 <subject>, <verb>, <object> 원소를 이용하여 컨텍스트 및 프로파일의 관계를 기술함을 의미한다. 표 5는 uWDL로 기술된 회의 준비 시나리오 문서 중에 사용자 검증에 해당하는 내용이다. <node> 원소는 웹 서비스로 등록된 User_Search_Service의 verifyingUser라는 오퍼레이션을 의미한다. <context> 원소는 센서로부터 전달되는 컨텍스트 정보를 표현하기 위하여 <constraint> 원소의 <subject>, <verb>, <object> 하위 원소를 이용하여 Location, Situation, Date 조건을 검사한다. 컨텍스트 정보는 4장에서 제안한 구조적 컨텍스트 모델 형식의

표 5 uWDL로 기술된 회의 준비 시나리오 문서의 일부분

```
<?xml version="1.0" encoding="UTF-8"?>
<uWDL version="0.1" log="0">
  <source>--where--</source>
  <node name="User_Search_Service">
    <service>
      <wsdl>http://cs.ssu.ac.kr/axis/services/user_search_service.wsdl</wsdl>
      <portType>UserSearchService</portType>
      <operation>verifyingUser</operation>
    </service>
  </node>
  <link nodeName="User_Search_Service">
    <case type="default">
      <condition>
        <context>
          <rule>
            <constraint>
              <subject type="PersonType">A</subject>
              <verb type="LocationType">Location</verb>
              <object type="RoomType">301</object>
            </constraint>
            <constraint composite="AND">
              <subject type="PersonType">A</subject>
              <verb type="SituationType">Situation</verb>
              <object type="ConferenceType">presentation</object>
            </constraint>
            <constraint composite="AND">
              <subject type="ConferenceType">presentation</subject>
              <verb type="DateType">Date</verb>
              <object type="TimeType">10:00</object>
            </constraint>
          </rule>
        </context>
      </condition>
      <action>
        <export type="data">
          <port>Download_Files:ip_address</port>
        </export>
      </action>
    </case>
  </link>
  <sink>result</sink>
</uWDL>
```

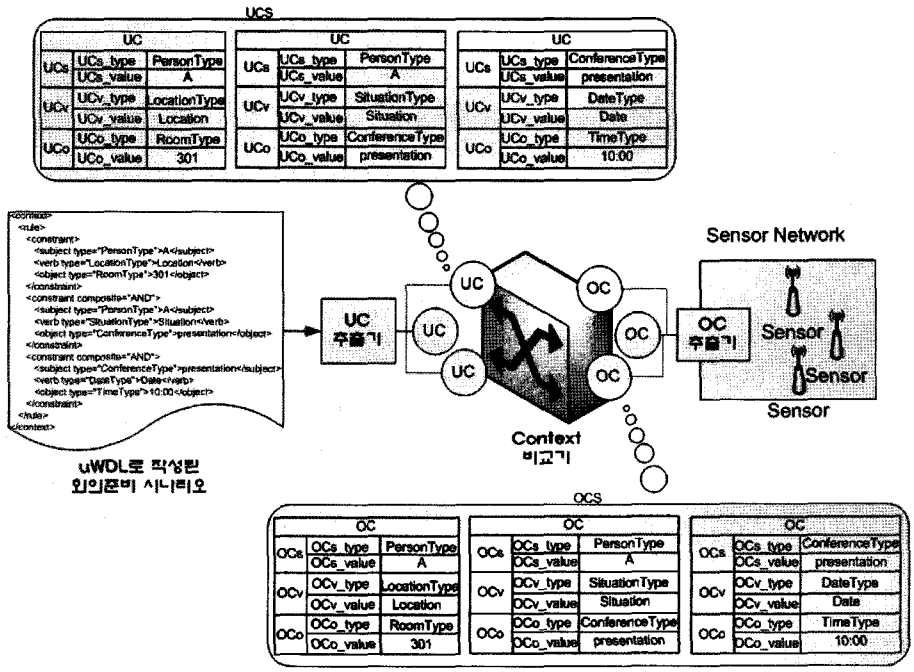


그림 6 uWDL로 작성된 회의준비 시나리오의 검증 과정

객체로 전달되며, 이 정보를 이용하여 현재 상황과 시나리오에서 기술한 컨텍스트 정보가 일치하는지를 판별하여 <conditon>이 참(true)일 경우 <action> 원소의 Download_Files 서비스를 호출한다.

그림 6에서는 6장에서 언급한 컨텍스트 비교 알고리즘을 이용하여 표 5의 회의 준비 시나리오 문서에서 기술된 컨텍스트 정보를 실제 유비쿼터스 컴퓨팅 환경에서 발생하는 컨텍스트 정보와 비교하였다. 이를 통하여 uWDL로 기술된 시나리오 문서가 실제 상황 정보에 따라 적합한 서비스를 호출할 수 있는지를 검증하고 있다. uWDL 시나리오 문서에서 기술된 상황 정보는 개별적인 엔티티들인 UC(Ubiquitous scenario Context) 컨텍스트 모델로 추출된다. 이때, 추출된 각 UC들은 컨텍스트 비교기에 의해 하나의 UCS (UC Set)를 구성한다. 이때, 컨텍스트 비교기는 유비쿼터스 컴퓨팅 환경에 존재하는 센서 네트워크로부터 발생되는 OC(Objectified Context)들의 의미정보와 UCS를 구성하는 각 UC들의 의미정보 사이의 비교를 통해 uWDL로 표현된 컨텍스트 정보가 현재의 상황과 일치하는지를 판단한다. 컨텍스트 비교기는 uWDL 시나리오의 컨텍스트를 구성하는 각 UC와 OC들간의 컨텍스트 비교를 위해 표 4의 컨텍스트 비교 알고리즘을 사용한다. 즉, 컨텍스트 비교기는 알고리즘을 통해 UC에 부합하는 OC의 발생 여부를 확인할 수 있으며, 그 결과에 따라 uWDL 시나리오에 기술된 컨텍스트 기반 서비스 분기를 위한 정보를 제공할 수 있다.

8. 결론 및 향후 연구과제

본 논문에서는 사용자에게 상황인지 자동화 서비스를 지원하기 위해 유비쿼터스 컴퓨팅 환경에 워크플로우를 적용하고, 컨텍스트 정보에 따라 적합한 서비스를 제공하기 위해 구조적인 컨텍스트 모델을 설계하였다. 구조적인 컨텍스트 모델은 유비쿼터스 환경의 컨텍스트 정보를 지식기반 관점에서 표현하여 복잡한 컨텍스트 정보들을 쉽게 표현할 수 있다. 또한 구조적인 컨텍스트 모델을 워크플로우의 상태 전이 조건에 이용하는 워크플로우 언어인 uWDL을 제안하였다. uWDL은 상황에 적합한 서비스들을 선택하거나 반복 또는 병렬적으로 처리할 수 있으며, 서비스들의 흐름을 정의하고 흐름에 영향을 미치는 상태 전이조건이 상황인지 기반으로 이루어질 수 있도록 구조적인 컨텍스트 모델을 사용한다.

uWDL은 웹 서비스 기반의 워크플로우 언어이므로 유비쿼터스 환경처럼 이질적이고 다양한 서비스들에 대한 통합, 관리 및 실행을 가능하게 한다. 유비쿼터스 환경처럼 컴퓨팅 환경 뿐만 아니라 센서, 사용자 정보 및 환경으로부터 얻을 수 있는 모든 정보들을 이용하는 서비스들은 다양한 형태의 플랫폼, 프로토콜 및 언어로 개발되며, 이를 통합하기 위해서는 표준화된 인터페이스를

제공하는 웹 서비스 기반의 워크플로우 개발이 요구된다.

본 논문은 유비쿼터스 컴퓨팅 환경에서 발생하는 컨텍스트 정보를 uWDL 언어에서 사용자에게 적합한 서비스를 선택하는 수단으로 사용하였다. 이를 위하여 본 논문의 구조적인 컨텍스트 모델에서는 {주어, 동사, 목적어}로 표현되는 컨텍스트를 정의하였으며, 이 컨텍스트 정보를 사용하여 상황인지 서비스를 제공하는 방법을 제안하였다. 컨텍스트 정보는 고수준의 컨텍스트 정보를 유추하기 위하여 인공지능적인 처리를 필요로 하는데, 인공지능 관점에서 {주어, 동사, 목적어}는 지식을 표현하는 기본적인 표현법이다. 또한 {주어, 동사, 목적어} 표현은 시멘틱 웹의 여러 온톨로지 언어의 가장 기본적인 자원 표현 방법이기도 하다. 그러나 uWDL에서는 컨텍스트 자체에 대한 전체적인 온톨로지 정보를 사용하지 않고, 온톨로지에서 자원을 기술하기 위한 {주어, 동사, 목적어}의 트리플 정보만을 필요로 하므로, 온톨로지에 대한 구체적인 언급은 추후에 다른 논문에서 다루고자 한다.

참 고 문 헌

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

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

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

[4] D. Saha, A. Mukherjee, "Pervasive Computing: A Paradigm for the 21st Century," IEEE Computer, IEEE Computer Society Press, pp.25-31, 2003.

[5] Guanling Chen, David Kotz, "A Survey of Context-Aware Mobile Computing Research," Technical Report, TR200381, Dartmouth College, 2000.

[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] 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.

[8] Deborah L. McGuinness, Frank van Harmelen (eds.), "OWL Web Ontology Language Overview," W3C Recommendation, 2004.

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

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

[11] Anind k. Dey, "Understanding and Using Context," Personal and Ubiquitous Computing, Vol 5, Issue 1, 2001.

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

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

[14] Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," Pervasive 2002, LNCS 2412, pp.167-180, 2002.

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

[16] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis and M. Weiser, "The ParcTab Ubiquitous Computing Experiment," Technical Report CSL-95-1, Xerox Palo Alto Research Center, 1995.

[17] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing," IEEE Pervasive Computing, 2002.

[18] Robert Grimm, "System support for pervasive applications," Phd thesis, University of Washington, 2002.

[19] Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, Klara Nahrstedt, "Gaia: A Middleware Infrastucture to Enable Active Spaces," IEEE Pervasive Computing, pp.74-83, 2002.

[20] Aho, A., V., Sethi, R., Ullman, J., D.: Compilers: Principles, Techniques and Tools. Addison-Wesley, (1986).

[21] Bates, J., Lavie, A.: Recognizing Substring of LR(K) Languages in Linear Time. ACM TOPLAS. Vol.16. No.3. pp.1051-1077, (1994).

한 주 현

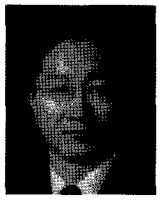
2000년 숭실대학교 컴퓨터학부(공학사) 2002년 숭실대학교 대학원 컴퓨터학과(석사). 2002년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 유비쿼터스 컴퓨팅, 분산/병렬처리, 시스템 소프트웨어, 시스템 보안

김 은 희

1993년 숭실대학교 전자계산학과(학사) 1998년 숭실대학교 대학원 전자계산학과(석사). 1998년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 유비쿼터스 컴퓨팅, 시스템 소프트웨어, 분산/병렬처리

최 재 영

정보과학회논문지 : 시스템 및 이론
제 32 권 제 6 호 참조



조 위 덕

1981년 서강대학교 전자공학과(학사) 1983년 한국과학기술원 전기 및 전자공학과(석사). 1987년 한국과학기술원 전기 및 전자공학과(박사). 1983년 3월~1990년 3월 금성전기(현LG전자) 기술연구소 DSP 연구실장. 1990년 3월~1991년 10월 한국생산기술연구원 전자정보시스템연구부 팀장/조교수 1991년 11월~2003년 9월 전자부품연구원 시스템연구본부 본부장. 2003년 1월~현재 유비쿼터스컴퓨팅사업단 단장, 아주대학교 전자공학부 교수. 관심분야는 유비쿼터스 컴퓨팅/네트워크, 센서 네트워크, Post-PC(차세대 Smart PDA), 디지털방송/이동통신 연계 융합플랫폼기술, 무선인터넷응용기술