# 동적인 사용자 서비스 요구를 지원하는 상황인지 워크플로우 시스템

## (A Context-Aware Workflow System for Supporting Users' Dynamic Service Demands)

최 종 선 † 조 용 윤 †† 최 재 영 †††
(Jongsun Choi) (Yongyun Cho) (Jaeyoung Choi)

**요 약** 유비쿼터스 컴퓨팅 환경에서의 상황인지 서비스는 동적으로 발생하는 사용자의 상황 정보에 따른 서비스 제공을 지향한다. FollowMe와 uFlow와 같은 상황인지 워크플로우 시스템은 사용자의 상황 정보를 서비스의 분기 조건으로 표현한 워크플로우 기반의 상황인지 웹 서비스를 제공한다. 그러나 그들은 워크플로우의 실행 중에도 동적으로 발생할 수 있는 사용자의 서비스 요구에 대해 진행 중인 워크플로우 시나리오에 즉각적으로 적용할 수 있는 방법을 제공하지 못한다.

본 논문에서는 실행 중인 워크플로우 서비스의 중단없이 사용자가 입력하는 새로운 서비스 요구를 초기 워크플로우 시나리오에 동적으로 반영할 수 있는 상황인지 워크플로우 시스템을 제안한다. 제안하는 시스템은 사용자의 새로운 서비스 요구에 대해 실행중인 시나리오에서의 변경 위치를 점진적 파싱(Incremental Parsing)을 통해 정확히 인식하고 초기 시나리오에서 영향을 받는 부분만을 신속히 재구성함으로써 동적으로 발생하는 사용자의 상황정보를 보다 빠르고 효율적으로 초기 시나리오에 적용할 수 있으며, 워크플로우 흐름의 중단없이 계속적인 서비스를 제공할 수 있다. 이를 통해 사용자는 시간과 공간에 관계없이 원하는 서비스를 상황인지 워크플로우 시나리오에 반영할 수 있으며, 기술된 사용자 컨텍스트에 따라 상황인지 서비스의 실행을 보장받을 수 있다.

**키워드** : 유비쿼터스 컴퓨팅, 상황인지 워크플로우 시스템, DItree 동적 재구성

***Abstract*** A context-aware service in ubiquitous computing environments aims to supply services according to users' situation information that is dynamically occurring. The existing context-aware workflow systems, such as FollowMe and uFlow, provide context-aware services based on a workflow, which uses users' situation information as transition conditions of a service. But they can't apply users' new service demands, which may dynamically occur even when a workflow is on going, to a workflow scenario.

In this paper we propose a context-aware workflow system, which can reapply users' new service demands into an initial workflow without interrupting or deleting the workflow. The proposed system can provide context-aware services without interrupting of service by recognizing exactly a place holder that has to be changed in a workflow scenario and by reconstructing only the changed parts through an incremental parsing method. Therefore, a user can immediately apply his new service demands to an on-going workflow scenario, and he can be guaranteed continuous executions of context-aware services according to a workflow scenario, which includes new service demands.

**Key words** : ubiquitous computing, context-aware workflow system, dynamic DItree reconstructing

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
† 학생회원 : 숭실대학교 컴퓨터학부
jschoi@ss.ssu.ac.kr
†† 정 회 원 : 숭실대학교 컴퓨터학부
yycho@ss.ssu.ac.kr
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
choi@ssu.ac.kr
논문접수 : 2006년 8월 29일
심사완료 : 2007년 3월 31일

## 1. 서 론

유비쿼터스 컴퓨팅 환경을 위한 워크플로우는 서비스 전이 조건으로 사용자의 상황정보를 이용하며, 이러한 환경에서의 사용자 상황정보는 인간과 환경과의 상호작용을 통해 컨텍스트로 표현되는 정보이다[1]. 유비쿼터스 컴퓨팅 환경의 워크플로우 시스템은 사용자가 작성

한 시나리오를 바탕으로 실제 유비쿼터스 환경에서 발생하는 컨텍스트에 따라 상황인지 서비스의 실행을 제공한다[2]. 유비쿼터스 환경에서의 상황인지를 지원하기 위한 워크플로우 언어는 서비스 분기 및 선택을 위한 서비스 전이 조건(transition condition)으로써 사용자의 상황 정보를 표현할 수 있어야 한다.

WSFL[3], BPEL4WS[4], XLANG[5]과 같은 기존 웹 서비스 기반 워크플로우 언어들은 서비스의 분기와 선택을 위한 서비스 전이 조건으로 이전 서비스 결과로부터 XML 링크 기능(XPath, XLink, XPoint)[6]의 조건-관계 연산식을 통해 얻어진 값을 이용할 수 있다. 그러나 유비쿼터스 환경에서 발생하는 상황정보는 조건-관계 연산식으로 얻어질 수 있는 단순한 값이 아니고 사용자의 위치 정보 및 시간 정보 등과 같은 복합적인 상황정보의 집합으로 구성된다. 즉, BPEL4WS, WSFL, 그리고 XLANG에서의 XPath나 XPoint의 기능만을 통해서는 유비쿼터스 환경에서의 워크플로우에 존재하는 서비스 간의 분기 및 선택을 위한 충분한 전이 조건을 얻을 수 없다. uWDL(ubiquitous Workflow Description Language)은 유비쿼터스 컴퓨팅 환경에서 발생하는 사용자의 컨텍스트 정보를 서비스 전이 조건으로 표현할 수 있는 상황인지 워크플로우 언어이며, 기존의 웹 서비스 기반의 워크플로우 언어들의 기능을 간소화하였다[7]. 개발자는 uWDL을 이용하여 워크플로우 서비스의 분기 조건으로써 사용자의 상황정보를 기술하는 uWDL 시나리오를 작성할 수 있으며, 생성된 uWDL 시나리오 문서를 기반으로 유비쿼터스 환경에서의 상황인지 서비스를 제공할 수 있는 워크플로우 응용 프로그램 개발이 가능하다.

시나리오 기반의 개발 환경에서 사용자가 초기 시나리오에 기술했던 서비스와 상황정보는 동적으로 변경될 수 있으며, 사용자의 서비스 요구는 초기 시나리오를 작성하는 시점에서 뿐만 아니라 워크플로우 서비스가 실행되는 도중에도 발생할 수 있다. 따라서 유비쿼터스 컴퓨팅 환경에서의 상황인지 워크플로우 시스템은 동적으로 발생하는 사용자의 서비스 요구를 실행 중인 워크플로우 시나리오에 적용하고, 이를 기반으로 사용자에게 즉각적으로 적합한 서비스를 제공할 수 있어야 한다. 그러나 기존의 uWDL 기반 상황인지 서비스 시스템[7]은 동적으로 입력되는 사용자의 새로운 서비스 요구를 초기에 작성하였던 시나리오에 동적으로 적용할 수 있는 방법을 제공하지 않는다. 또한 현재까지 시도되고 있는 유비쿼터스 환경에서의 상황인지 워크플로우 시스템[7,9,10]은 사용자의 서비스 요구를 초기 시나리오의 작성 시점에서만 기술할 수 있기 때문에, 서비스 실행 중에 발생하는 사용자의 서비스 요구를 초기 작성된 시나리오에 동적으로 적용할 수 있는 방법이 없다. 따라서 유비쿼터스 환경을 위한 상황인지 워크플로우 시스템은 진행 중인 상황인지 워크플로우 서비스의 중단이나 초기화없이 동적으로 발생하는 사용자의 새로운 서비스 요구를 처리할 수 있는 방법이 요구된다.

본 논문에서는 실행 중인 서비스의 중단없이 새로운 사용자 서비스 요구를 시나리오에 즉각적으로 반영할 수 있는 상황인지 워크플로우 시스템을 제안한다. 제안하는 시스템은 유비쿼터스 환경에서 동적으로 발생하는 사용자의 서비스 요구에 대해 영향을 받는 시나리오의 변경 위치(placeholder)를 정확하게 인식하고, 새로운 서비스 요구에 대하여 영향을 받는 부분만을 변경하여 초기 시나리오를 재구성한다. 워크플로우 시나리오의 변경을 위한 문서의 재파싱은 진행 중인 서비스의 실행과 관련된 부분을 제외함으로써, 실행중인 서비스를 중단시키지 않고 새롭게 구성된 시나리오를 기반으로 계속적인 상황인지 워크플로우 서비스를 제공할 수 있다.

논문의 구성은 다음과 같다. 2장의 관련연구 부분에서는 기존 연구의 고찰을 통하여 기존연구의 문제점 및 요구사항에 대하여 살펴본다. 3장에서는 새로운 요구사항을 지원하기 위하여 본 논문에서 제안하는 시스템에 대한 설계 및 구성에 대하여 설명한다. 4장에서는 제안하는 시스템의 적용을 위한 실험 및 평가를 하며, 마지막으로 5장에서는 결론으로 맺는다.

## 2. 관련연구

### 2.1 워크플로우(Workflow)

WfMC(Workflow Management Coalition)에서는 워크플로우를 "전체 혹은 부분적인 비즈니스 프로세스의 자동화 또는 컴퓨터로 처리되는 간이화(facilitation)[11]"로 정의하고 있다. 워크플로우는 하나의 큰 작업의 수행이 완료될 때까지 일어나는 하위 작업들의 흐름을 표준화된 방법으로 표현한다. 워크플로우는 하나의 큰 작업이 수행 완료될 때까지 일어나는 하위 작업들의 흐름을 XML 기반의 언어를 이용하여 표준화된 방법으로 표현한다. 이 때 워크플로우의 하위 작업들 간에는 의존성이나 수행 순서, 동시 수행 가능 여부 등의 다양한 관계가 나타나며, 이와 같이 큰 작업을 작은 작업의 연관된 흐름으로 표현하면 각 작업에 효율적으로 자원이 배분되어 전체 작업의 효율성을 향상시킬 수 있다[3,7].

워크플로우에서 프로세스는 공통의 목적을 달성하기 위하여 연결된 일련의 동작들의 집합이며, 전이 조건은 하나의 단위 작업에서 또 다른 작업으로의 전이를 위하여 해당 작업의 상태를 평가하기 위한 기준(criteria)이라고 할 수 있다[11]. 워크플로우 관리 시스템은 일련의 워크플로우 로직의 컴퓨터 표기법에 따라 구동되는 실

행순서를 가지는 소프트웨어의 실행을 통하여 워크플로우를 정의, 관리 및 실행해주는 것이다. 이러한 워크플로우 시스템은 워크플로우 언어에 명시된 이러한 전이 조건 또는 제약조건을 워크플로우 작업들의 흐름을 결정하는데 사용한다[3,11].

### 2.2 상황인지 환경에서의 워크플로우 시스템

워크플로우 기술은 소프트웨어 엔지니어링, 헬스케어(healthcare), 금융, 생산, 사무 자동화, 은행 등과 같이 매우 다양한 환경에 퍼져 잘 활용되고 있다[12]. 이러한 워크플로우 기술들은 예측 가능하면서 잘 정의된 비즈니스 프로세스의 경우에 매우 효과적이라는 것이 증명되었다. 그럼에도 불구하고 유동적인 비즈니스 도메인에서 발생하는 요구사항을 기존의 정적인 개념의 워크플로우 기술로는 감당하기 어려운 실정이다[13]. 이러한 문제를 극복하기 위하여 최근 연구 진행 중인 워크플로우 관리 시스템은 유동적인 개념을 포함시키고 있으나, 동적인 요구사항의 처리에 대한 상당부분을 향후 연구 과제를 남겨둔 상태이며, 유비쿼터스 환경에서 이러한 연구를 적용하지는 못하고 있다[14]. 또한 현재 유비쿼터스 환경에서의 워크플로우 시스템은 초기 프로토타입 수준의 연구가 진행 중에 있으며[9], 사용자가 원하는 서비스들의 흐름을 일정 형태로 기술한 시나리오를 기반으로 상황인지 워크플로우 서비스를 제공하는 시스템에 대한 초기 연구가 진행 중에 있다[10]. 그러나 동적인 사용자의 서비스 요구를 서비스의 실행 중에 즉각적으로 적용할 수 있는 상황인지 워크플로우 시스템에 대한 연구는 아직 시도되고 있지 않다.

WorkSco[14]는 포르투갈의 리스본 기술 대학이 주축되어 진행 중인 프로젝트로, 비즈니스 프로세스에서 발생하는 동적인 요구사항에 대한 처리 능력을 갖춘 적응형 워크플로우 관리 기술을 적용하는 것이 목적이다. 이것은 워크플로우를 최대한 가볍게 만들기 위하여 가벼운 기능만을 워크플로우 코어에 두고 반대로 무거운 기능들은 따로 분리하는 마이크로커널과 같이 플러그-인 방식을 사용하는 마이크로 워크플로우 모델[15]에 기반을 두고 있다. WorkSco는 다양한 비즈니스 도메인에서 발생하는 요구사항에 대한 동적인 대응을 위하여 동적 진화(dynamic evolution)[16]와 open point 기반의 적응(adaptation)[14]의 두 가지 접근방식을 사용하고 있으나, 전자의 경우 정책적인 방안을 제시하고는 있지만 다량의 경험적 지식을 필요로 하기 때문에 향후 연구 과제를 남겨두고 있는 상태이며, 후자의 경우는 뚜렷한 방법을 제시하고 있지 못하고 있다. 또한 이 프로젝트는 유동성을 워크플로우 관리 기술에 적용하고 있지만 유비쿼터스 환경에서 이러한 기술을 접목시키지는 못하고 있다.

FollowMe[10]는 중국의 Nanjing 대학에서 현재 연구되고 있는 편재형 컴퓨팅 환경을 위한 OSGi 프레임워크로써 온톨로지 기반의 컨텍스트 모델과 워크플로우 기반의 응용 모델을 OSGi 프레임워크에 통합한 기반구조이다. 이것은 편재형 컴퓨팅 환경에서 존재하는 다양한 도메인에 적용할 수 있는 통일된 형태의 상황인지 워크플로우 기반구조를 제공하며, 다른 도메인의 컨텍스트들과 응용 프로그램들을 접속시켜 다양한 도메인에 최적화될 수 있는 장점이 있다. FollowMe는 다양한 도메인에서 발생하는 사용자의 서비스 요구를 충족시키기 위하여 시나리오 기반의 워크플로우 모델을 사용하고 있으나, 이러한 사전에 기술된 시나리오에 따라 진행되고 있는 워크플로우 서비스의 실행 도중에 동적으로 발생할 수 있는 사용자의 요구를 처리할 수 있는 기능은 제공하지 못하고 있다.

uFlow(Ubiquitous Workflow) 시스템[7]은 기존 웹 서비스 기반 워크플로우 언어들이 서비스 전이 조건으로 컨텍스트를 기술하지 못하는 점을 해결하기 위하여 새로운 유비쿼터스 워크플로우 언어인 uWDL을 기반으로 하는 상황인지 워크플로우 서비스 프레임워크이다. uWDL[8]은 워크플로우가 주변 상황에 대한 상황을 인지하고, 상태를 전이할 수 있도록 워크플로우의 상태 전이 제약 조건에 컨텍스트를 명시하기 위한 구조적 컨텍스트 모델[8]을 기반으로 하고 있으며, 이러한 uWDL을 통해 시나리오에 기술한 컨텍스트에 따라 사용자의 서비스 요구를 처리하고 실행한다. 시나리오에 기반을 두고 있는 uFlow 시스템 또한 FollowMe와 마찬가지로 초기 시나리오에 기술된 서비스에 대한 처리만을 할 뿐 진행 중인 워크플로우 서비스 중도에 동적으로 발생하는 사용자 서비스 요구에 대한 처리 방법을 지원하지 못한다.

앞서 살펴본 연구사례[7,9,10,14]들은 유비쿼터스 환경의 다양한 도메인에서 동적으로 발생할 수 있는 사용자의 서비스 요구를 처리할 수 있는 적절한 대안을 마련하지 못하고 있다.

## 3. 시스템 설계 및 구성

본 논문에서는 사용자가 입력하는 새로운 서비스 요구를 실행중인 워크플로우 시나리오에 반영하여 계속적인 서비스 제공을 보장하는 동적인 상황인지 워크플로우 시스템을 설계하였다.

제안하는 시스템의 전체 구조는 그림 1과 같다. 그림 1에서 상황인지 워크플로우 엔진(Context-aware Workflow Engine)은 입력된 워크플로우 문서를 파싱하기 위한 상황인지 워크플로우 파서(Context-aware Workflow Parser : CWparser), 서비스 실행 중에 사용자의 새로

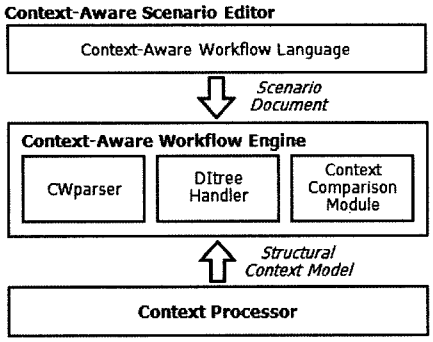



그림 1 상황인지 워크플로우 시스템 구조

운 서비스 요구를 기존 시나리오에 적용하기 위한 시나리오 문서 인스턴스 트리 처리기(Document Instance Abstract Syntax Tree Handler : DItree Handler), 그리고 시나리오에 기술된 서비스 실행 조건에 따라 상황인지 서비스의 실행을 결정하기 위한 컨텍스트 비교 모듈(Context Comparison Module : CCM)로 구성된다.

CWparser는 개발자가 상황인지 워크플로우 언어를 통해 생성한 워크플로우 시나리오 문서를 파싱하여 DItree를 출력하며, 시나리오 문서에 기술된 컨텍스트를 DItree의 RDF 기반 트리플릿(triplet) 형태의 부분트리(sub-tree)로 표현한다. DItree Handler는 사용자로부터 동적으로 발생하는 새로운 서비스 요구를 실행 중인 상황인지 워크프로우 서비스의 중단없이 초기 시나리오에 적용하기 위해 워크플로우 시나리오에 대한 DItree를 재구성한다. CCM은 DItree의 부분트리에 표현된 컨텍스트 정보를 실제 유비쿼터스 환경에서 동적으로 발생하는 사용자의 상황정보와 비교하여 서비스의 실행을 결정한다. 상황인지 워크플로우 시나리오 편집기(Context-aware Scenario Editor : CSeditor)는 상황인지 워크플로우 개발 언어의 문법 구조에 유효한 워크플로우의 작성을 지원하는 그래픽 기반의 편집 환경을 제공한다. 다음은 제안하는 시스템이 포함하는 각각의 구성 모듈에 대해 설명한다.

### 3.1 상황인지 워크플로우 파서(Context-aware Workflow Parser : CWparser)

사용자가 상황인지 워크플로우 언어를 통해 생성한 워크플로우 시나리오 문서의 파싱을 위하여 CWparser는 시나리오 문서의 토큰을 인식하는 어휘 분석기(lexical analyzer)와 일련의 토큰들의 조합이 상황인지 워크플로우 언어의 DTD에 적합한 구문 구조를 갖는지를 검사하는 구문 분석기(syntax analyzer)로 구성된다. 본 논문에서 제안하는 시스템은 상황인지 워크플로우 언어로써 uWDL을 사용한다. uWDL은 서비스의 실행 조건으로써 컨텍스트를 기술할 수 있는 문법적 엘리먼트를 포함하며, 컨텍스트들의 표현을 위하여 RDF 트리플릿 형태로 구조화된 컨텍스트 기술 모델을 사용한다[8]. RDF 트리플릿 형태의 컨텍스트 기술 모델은 컨텍스트를 표현하는 표준 기술로써 이해하기 쉽고 보다 고수준의 컨텍스트를 표현을 위한 온톨로지 기반의 컨텍스트 모델을 적용시키기가 용이하다[1,2]. CWparser는 입력 uWDL 문서에 대해 uWDL DTD AST(Abstract Syntax Tree)를 이용해 파싱을 수행하고, 파싱 결과로써 uWDL 문서 구조 정보인 DItree를 생성한다. CWparser는 uWDL 문서에 기술된 컨텍스트를 RDF 트리플릿 구조의 엘리먼트 노드로 구성된 부분트리로 표현한다. 그림 2는 uWDL 시나리오문서를 기반으로 동적인 상황인지 서비스를 제공하기 위한 CWparser와 컨텍스트 비교 모듈을 포함하는 상황인지 워크플로우 엔진의 구성도이다.

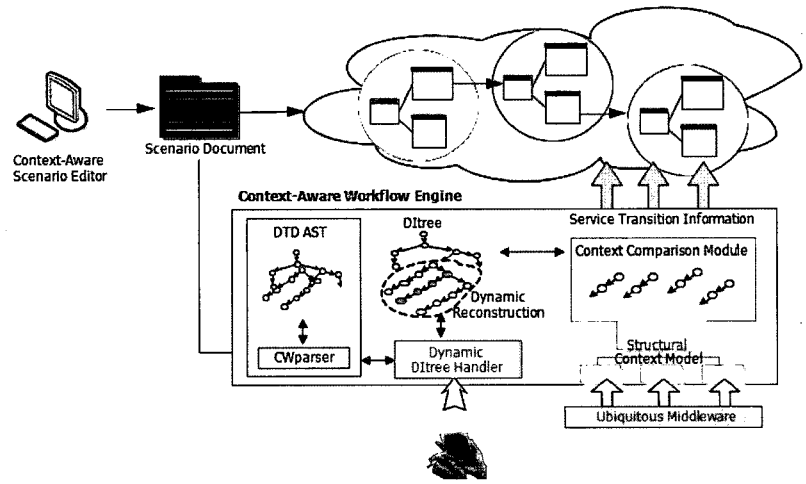



그림 2 상황인지 워크플로우 엔진의 구성도

그림 2의 CCM는 서비스 분기정보 생성을 위하여 유비쿼터스 미들웨어로부터 전달되는 사용자에 대한 상황정보와 시나리오 DItree의 컨텍스트 부분트리를 비교한다. 인스턴스 트리 처리기(DItree Handler)는 사용자가 로컬 컴퓨팅 디바이스나 개인용 이동 단말기를 통하여 동적으로 입력되는 새로운 서비스에 대한 요구사항을 CWparser에 전달한다. CWparser는 사용자가 입력한 새로운 서비스 요구에 따라 초기 시나리오를 재구성한다. 이 때, 진행 중인 서비스의 실행이 끊이지 않고 사용자의 서비스에 대한 요구사항을 시나리오에 반영하기 위하여, CWparser는 초기 시나리오 문서 전체를 처음부터 다시 파싱하지 않고 새로운 서비스 요구에 대해 DItree에서 영향을 받는 부분만을 파싱하여 시나리오를 재구성하기 위한 점진적 파싱(incremental parsing) [17]을 실시한다. CWparser는 워크플로우 서비스의 흐름을 방해하지 않는 점진적 파싱을 통하여 유비쿼터스 환경의 다양한 도메인으로부터 발생할 수 있는 사용자의 서비스 요구를 동적으로 시나리오에 적용할 수 있다.

### 3.2 DItree(Document Instance Tree)

그림 3은 그림 1의 CSeditor를 통해 개발자가 작성한 uWDL 상황인지 워크플로우 서비스 시나리오 문서에 대해 uWDL DTD를 이용하여 CWparser가 생성하는 DItree 구조의 일부분이다. 그림 3의 DItree에서 점선으로 표시된 <constraint> 영역은 워크플로우 서비스를 분기하는데 필요한 전이 조건과 관련된 컨텍스트를 나타낸다. 즉, DItree에서 워크플로우 서비스 문서의 상황정보를 RDF 트리플릿 노드들로 나타내는 곳은 <constraint>를 루트 노드로 갖는 부분트리이다. uWDL 시나리오 문서에 기술된 컨텍스트 정보와 유비쿼터스 환경에서 발생하는 사용자 상황정보는 시간이나 위치와 같은 정보를 그 정보의 값(value)과 유형(type)으로 기술하여 사용자의 특정 상황에 대한 고유한 의미를 갖는다.

예를 들어, 유비쿼터스 환경에서 센서가 감지한 '홍길동'이라는 값을 가지는 컨텍스트는 '홍길동'의 컨텍스트 유형, 즉 '사람' 유형인지 '동물' 유형인지에 따라 그 의미가 명확히 구분될 수 있다. 따라서 CWparser는 표현하려는 컨텍스트의 값과 유형에 대한 정보를 컨텍스트 부분트리의 각 노드 구조 필드에 포함시킨다. CCM은 컨텍스트 부분트리의 노드 필드에 포함된 컨텍스트의 값과 유형 정보를 이용하여 사용자 상황정보와 컨텍스트 비교를 정확히 수행할 수 있으며, 또한 적합한 서비스 분기정보를 생성할 수 있다.

그림 4는 DItree 엘리먼트 노드의 구조와 노드가 이용하는 자료 테이블 구조를 나타내며, DItree 엘리먼트

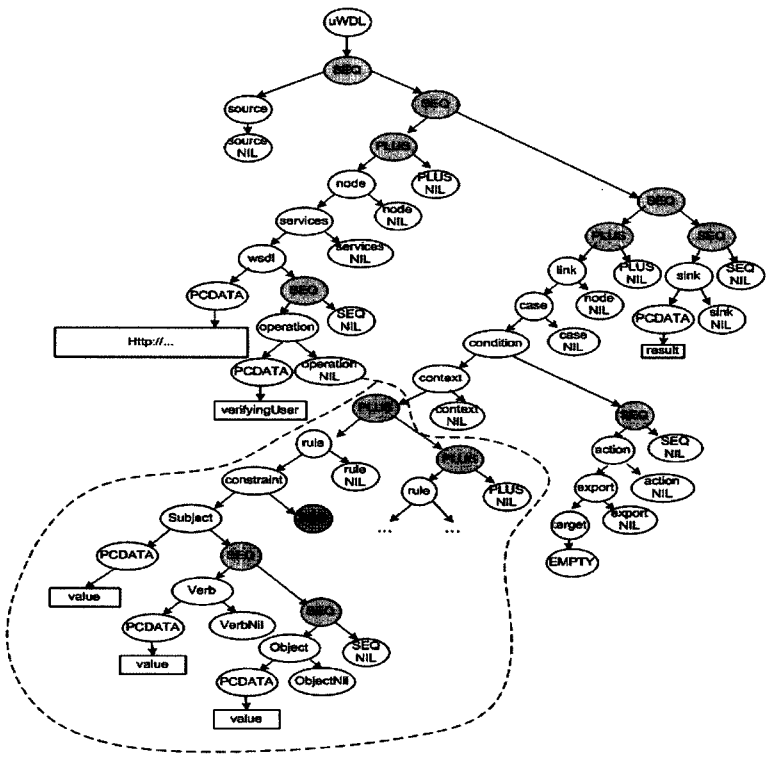



그림 3 uWDL 시나리오에 대하여 CWparser가 생성하는 DItree의 구조도

노드는 8개의 항목으로 구성된다. DItree의 각 노드는 이전 노드와 다음 노드간의 연결 정보를 통하여 전체 트리구조를 구성한다. 예를 들어, 그림 4는 uWDL 시나리오의 컨텍스트 정보를 나타내는 <constraint>의 <subject type="UserType"> John </subject> 문장에 대해 CWparser가 생성하는 DItree의 컨텍스트 부분트리의 노드 정보와 그 사용 테이블 정보를 나타낸다.

그림 4의 예제 노드 구조에서 65의 값을 갖는 Production Number는 각 엘리먼트 노드를 식별할 수 있는 고유의 엘리먼트 번호이다. DItree를 구성하는 각 노드는 일반 엘리먼트 노드와 연산자 노드로 구분된다. 일반 엘리먼트 노드는 uWDL의 DTD에서 표현된 각각의 엘리먼트, PCDATA, EMPTY와 대응된다. 연산자 노드는 uWDL의 DTD를 이루는 엘리먼트의 언어적 특성을 표현하기 위한 메타 문자를 나타낸다. DItree의 각 연산자 엘리먼트에 대해 PLUS, OPTION, STAR, OR, SEQ 연산자 노드로 표현된다. 연산자 엘리먼트는 실질적인 언어를 구성하는 문법 엘리먼트의 일부분이 아닌 그 엘리먼트의 언어적 특성을 나타내기 위한 정보로 사용된다. 예를 들어 그림 3의 DItree에서 <node> 엘리먼트에 대한 부모 엘리먼트 노드는 일반 엘리먼트 노드가 아니고 연산자 엘리먼트인 PLUS 노드이다.

노드 구조에서 87의 값은 해당 엘리먼트가 가지는 attribute 정보를 얻기 위하여 Attr_list 테이블의 해당 레코드를 가리키는 포인터이다. Attr_list 테이블의 Value_list 필드는 Value_list 테이블에 연결되며, 해당 attribute가 가질 수 있는 컨텍스트의 유형 정보를 의미하는 값의 목록을 나타낸다. 또한 컨텍스트 정보를 정확하게 표현하기 위한 유형 정보와 값 정보가 각각 Attr_list와 Opr_tab에 저장되어 있다. 따라서 CCM는 DItree의 컨텍스트 부분트리가 나타내는 컨텍스트 정보를 이용하여 사용자 상황정보와의 정확한 비교가 가능하고 올바른 서비스 분기정보를 생성할 수 있다.

노드 구조에서 89의 값은 PCDATA나 COMMENT 엘리먼트가 가지는 string 정보를 담고 있는 PCDATA_table 테이블 레코드를 가리키는 포인터이다. 또한 엘리먼트 노드의 자료구조는 같은 형태의 엘리먼트 노드 자료 구조를 가지는 자식 엘리먼트와 부모 엘리먼트에 대한 포인터 정보를 가진다. Prod 테이블은 엘리먼트 이름을 나타내는 name 필드, 연산자 엘리먼트 정보를 위한 opr 필드, 그리고 DItree에서 엘리먼트의 순서 정보를 나타내는 link 필드로 구성된다. 따라서 이러한 노드들 간의 구조 정보를 통해 문서의 전체적인 구조 정보와 내용 정보에 대해 알 수 있다.

### 3.3 시나리오 동적 재구성을 위한 점진적 파싱

기존 uFlow가 지원하는 상황인지 워크플로우 서비스는 초기 작성된 시나리오를 기반으로 워크플로우가 실행될 때 사용자로부터 동적으로 발생하는 새로운 서비스에 대해 진행 중인 워크플로우 시나리오에 적용할 수 있는 방법을 제공하지 않는다. 그러나 유비쿼터스 환경에서의 사용자는 언제든지 새로운 서비스를 요구할 수 있으므로, 상황인지 워크플로우 서비스 엔진은 이러한 요구를 지원할 수 있어야 한다.

이를 위해, 본 논문에서 제안하는 CWparser는 시나리오 기반의 서비스 실행 중에 사용자로부터 입력되는 새로운 서비스 요구를 초기 시나리오에 적용하기 위하여 점진적 파싱[17]을 수행한다. 점진적 파싱은 초기 시나리오 문서에 대해 파서가 생성한 기존의 파스트리 정보를 최대한 재사용하여, 유비쿼터스 환경 특성상 동적으로 빈번하게 발생하는 사용자의 서비스 요구사항에 대해 빠르고 효과적인 처리를 통한 상황인지 서비스 지원에 그 목적이 있다. 이를 위하여, CWparser는 새로운 서비스 요구에 대해 DItree에서 새롭게 변경되어야 할 최소한의 부분만을 재파싱하고 나머지 파스 트리부분은 재사용함으로써, 첫째 새로운 서비스의 적용과 실행에 필요한 노력을 최소화하고, 둘째 실행 중인 초기 워크플로우 서비스의 중단없이 새로운 서비스 요구를 처리할 수 있는 장점을 가질 수 있다.

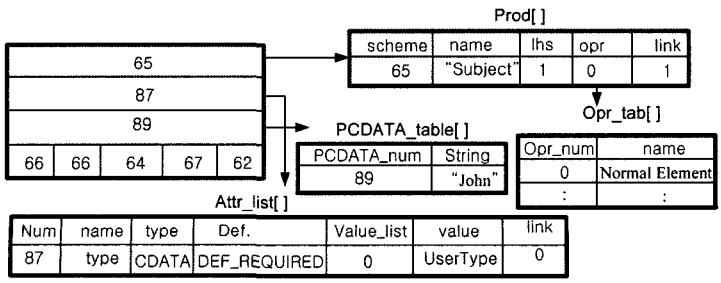
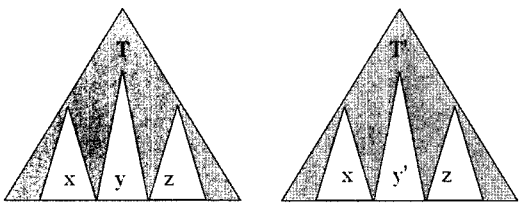



그림 4 컨텍스트 정보를 나타내는 <subject type="UserType"> John </subject> 문장에 대한 DItree의 구조

그림 5는 서비스가 실행되는 도중에 CWparser가 입력되는 사용자의 새로운 서비스 요구에 대하여 시나리오 동적 재구성을 수행하는 점진적 파싱의 개념도이다. CWparser가 수행하는 점진적 파싱은 이전 시나리오에 대한 파스 결과를 최대한 다시 이용하여 새로운 서비스 요구에 대해 구문분석을 통한 새로운 서비스 시나리오의 빠르고 동적인 DItree 재구성에 있다. 그림 5의 (a)는 초기 워크플로우 시나리오에 대해 CWparser가 생성한 DItree T를 나타내며, (b)는 새로운 서비스 요구에 대해 CWparser가 새롭게 재구성한 DItree T'이다. 즉, 그림 5의 T가 워크플로우 시나리오에서의 문장 ω = xyz에 대해 구성된 부분트리일 때, 새로운 서비스 요구에 대한 입력 문장 ω = xy'z에 대한 새로운 DItree는 T'이다.




(a) (b)

그림 5 시나리오 동적 재구성을 위한 점진적 파싱

T와 T'의 DItree는 실제로 새로운 서비스 요구에 대해 영향을 받지 않는 x, z, 그리고 나머지 T의 DItree 정보를 재사용하고 영향을 받는 초기 DItree T의 y 부분에 대해서만 파스 트리를 재구성할 필요가 있다. CWparser는 적어도 새로운 서비스에 대해 재구성할 T'에 대해 y' 전체를 점유하는 T'0 에서부터 T'K까지 연속되는 새로운 부분트리를 생성한다. 새롭게 생성된 부분트리 y'은 새로운 서비스 요구에 대해 개발 언어의 문법에 따라 작성된 파스트리로써, 함수의 반복 호출 형태의 Top-down 방식으로 구현된 CWparser는 시나리오 작성 언어의 문법 구조에 따라 의미적으로 y'를 초기 DItree에 연결되어야 할 삽입 위치(placeholer)를 정확히 알아낼 수 있다.

점진적 파싱을 통해 CWparser는 새로운 서비스 요구에 대해 초기 시나리오 문서에 대한 DItree의 전부를 완전히 새롭게 생성하는 것이 아니라, 부분 트리만을 재생성하여 초기 DItree에 연결함으로써, 전체적인 서비스의 처리 속도와 효율성을 높일 수 있다.

표 1은 CWparser가 새로운 서비스 요구에 대한 부분트리를 생성하고 초기 DItree와의 연결을 통해 시나리오를 동적으로 재구성하기 위하여 사용하는 점진적 파싱 알고리즘이다. 표 1의 점진적 파싱 알고리즘을 통해 CWparser는 새로운 서비스에 대한 부분트리를 루트 노드로 하여 DItree에서 문법적으로 삽입될 수 있는 위치에 연결한다. 삽입 위치를 발견하기 위하여, CWparser는 DItree Handler를 이용하여 DItree를 순회하며, 현재 진행 중인 서비스에 대해 기술하고 있는 부분트리 영역의 루트 노드를 찾는다. 현재 실행 중인 서비스의 부분트리와 다음에 실행할 서비스의 부분트리 사이에 새로운 서비스를 위한 부분트리를 삽입할 수 있는지에 대한 의미적 연결 여부를 파악하고 조건을 만족하면 해당 위치에 삽입을 시도한다. 시스템이 제공하는 언파싱 규칙은 새롭게 생성된 DItree에 대해 출력 규칙인 언파싱 규칙을 통해 개발자나 사용자에게 새로운 서비스 요구가 적용된 시나리오 내용을 화면으로 출력한다.

### 3.4 상황인지 서비스를 실행을 위한 컨텍스트 비교

유비쿼터스 컴퓨팅 환경에서 존재하는 서비스가 사용자의 상황 정보에 따라 실행되기 위해서는 센서 네트워크에서 발생되는 상황 정보에 근거한 서비스 분기정보가 필요하다. 컨텍스트 비교 모듈(CCM)은 CWparser가 생성한 DItree 컨텍스트 부분트리와 사용자 상황정보의 비교를 통해 서비스 분기정보를 생성한다. 유비쿼터스 환경에서 발생하는 사용자 상황정보 및 시나리오에서 표현되는 컨텍스트는 RDF 기반 {주어, 동사, 목적어} 형태의 엔티티(entity)들에 대한 집합으로 표현된다. 즉, 센서 네트워크로부터 제공되는 컨텍스트 정보는 RDF

표 1 점진적 파싱 알고리즘

입력 : 초기 시나리오 문서에 대한 파스 트리와 새로운 서비스에 대한 부분 트리
출력 : 재구성된 시나리오 문서에 대한 파스 트리
방법 :

1. 입력된 부분트리를 위한 초기 루트 노드를 생성
2. 생성된 트리의 맨 마지막 노드의 위치를 저장
3. 마지막 노드를 리스트 노드로 변환하고 구문 분석기로부터 입력 토큰을 받아 추가
4. 추가된 노드를 루트로 하는 트리를 생성하기 위한 Nil 노드 추가
5. 입력된 토큰을 루트로 하는 트리를 생성하고 4에서 생성한 Nil 노드에 추가
6. 구문 분석기를 호출하여 다음 토큰을 읽고 2부터 반복
7. 초기 파스 트리에서 새로운 서비스에 대한 부분트리의 루트 노드가 끼워 들어갈 위치(placeholder)를 찾음
8. 언파싱 규칙을 이용한 새로운 파스트리를 통하여 재구성된 시나리오 문서를 생성

기반의 구조적 컨텍스트 모델을 통해 {주어, 동사, 목적어}로 구성된 엔티티의 집합으로 객체화된다.

또한, CWparser가 uWDL 시나리오에 대해 생성한 DItree에서 <constraint> 원소로 표현되는 컨텍스트 정보 역시 상황인지 워크플로우 개발 언어 문법이 제공하는 {주어, 동사, 목적어} 형태의 엘리먼트를 통해 표현된다. 따라서 본 논문에서는 CCM이 컨텍스트 비교에 이용하는 사용자 상황정보와 시나리오에 기술된 컨텍스트 정보를 표 2와 같이 표현한다. 즉, OC는 RDF 기반의 구조적 컨텍스트 모델을 통해 객체화된 컨텍스트를 의미하며, OCs, OCv, OCo는 각각 주어, 동사, 목적어를 의미하는 엔티티 정보이다. UC는 uWDL 시나리오에 기술된 컨텍스트의 DItree 컨텍스트 부분트리를 의미하며, sNode, vNode, oNode는 DItree의 컨텍스트 부분트리를 구성하는 <subject>, <verb>, <object> 엘리먼트 노드로 표현된 엔티티 정보이다. 예를 들어 "존(John)이 313호 있다"와 같은 컨텍스트 정보에 대한 RDF 기반 컨텍스트 부분트리 형태의 UC는 그림 6과 같다.

시나리오에서 하나의 컨텍스트 표현을 위하여 사용되는 여러 <constraint>들은 UC들의 집합 UCS={UC1, UC2, …, UC*i*}로 표현될 수 있으며, OC들의 집합 OCS는 OCS = {OC1, OC2, …, OC*i*}로 표현될 수 있다. 따라서 CCM은 DItree 컨텍스트 부분트리 노드를 통해 표현되는 UC와 유비쿼터스 미들웨어에 의해 객체화된 OC를 비교하여 서비스 분기정보를 생성한다. 이때, uWDL 시나리오를 통해 한정된 수의 UC로 표현되는 시나리오 컨텍스트 와는 달리 사용자 상황정보를 나타내는 OC는 센서 네트워크로부터 무수히 발생할 수 있다. 따라서 CCM은 무수하게 발생하는 OC들로부터 시나리오에 기술된 UC와 부합하는 OC를 정확하게 선택하기 위한 컨텍스트 비교 알고리즘을 사용한다.

표 3은 CCM이 서비스 분기정보 생성을 위하여 사용하는 컨텍스트 비교 알고리즘이다. 표 2에서 CCM은 OCS와 시나리오에 기술된 컨텍스트 부분트리의 root 노드를 입력받고, 컨텍스트가 일치하는지에 대한 Boolean 값을 출력한다. ocListener로부터 입력받은 OCS에 대해 각 OC를 RDF 트리플릿 형태의 컨텍스트 객체 모델인 OContextNode를 통해 o_node 클래스로 객체화한다.

표 2 컨텍스트 비교를 위한 사용자 상황정보와 시나리오 컨텍스트의 표현

| |
|---|
| OC = { (OCs_type, OCs_value), (OCv_type, OCv_value), (OCo_type, OCo_value) } |
| OC (Objectified Context) : 유비쿼터스 환경에서 발생한 저수준의 사용자 상황정보를 RDF 기반의 컨텍스트 모델을 통해 고수준 상황정보로 객체화 한 컨텍스트 정보 |
| (OCs_type, OCs_value) : OC의 주어로 표현되는 상황정보의 유형과 값 |
| (OCv_type, OCv_value) : OC의 동사로 표현되는 상황정보의 유형과 값 |
| (OCo_type, OCo_value) : OC의 목적어로 표현되는 상황정보의 유형과 값 |
| UC = {(sNode.type, sNode.value), (vNode.type, vNode.value), (oNode.type, oNode.value)} |
| UC (uWDL Scenario Context) : 시나리오에 기술된 컨텍스트를 나타내는 DItree의 <constraint> 서브트리 |
| (sNode.type, sNode.value) : 컨텍스트 부분트리를 구성하는 주어 노드의 유형과 값 |
| (vNode.type, vNode.value) : 컨텍스트 부분트리를 구성하는 동사 노드의 유형과 값 |
| (oNode.type, oNode.value) : 컨텍스트 부분트리를 구성하는 목적어 노드의 유형과 값 |

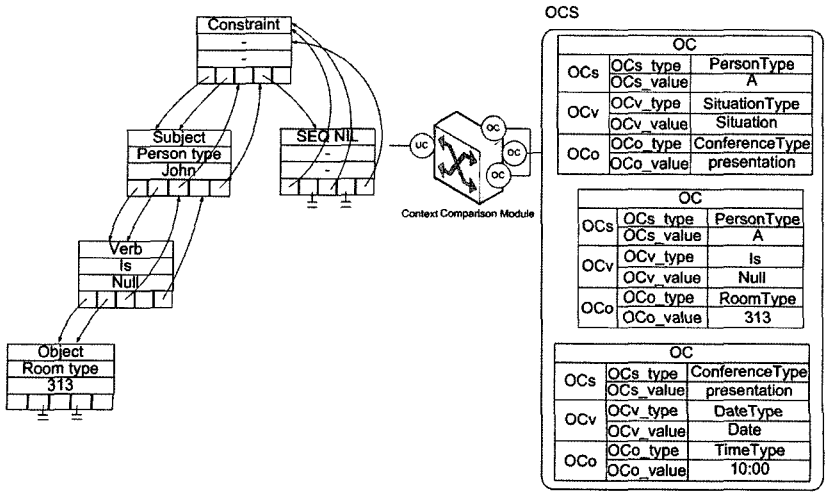



그림 6 시나리오 컨텍스트와 사용자 상황정보에 대한 UC와 OC의 표현

현재 서비스를 위한 i 번째 노드를 부분트리의 루트 노드로 하는 UC에 대해 UContextNode 클래스 유형의 u_node 객체 변수의 값으로 한다. 입력 OC와 UC를 나타내는 각 o_node와 u_node의 값과 유형 비교를 통해 컨텍스트와 일치되는지를 결정한다.

이때, 그림 6과 같이 RDF 트리플릿 구조의 부분트리를 구성하는 하위 node 엘리먼트와의 컨텍스트 비교를 위하여 현재의 u_node와 o_node가 가지는 왼쪽 child를 새로운 u_node와 o_node의 값으로 변경한다. 컨텍스트 비교를 통해 CCM은 UC의 값과 유형에 모두 일치하는 OC를 입력 OCS 중에 발견하면 해당 UC를 서비스 실행 조건으로 가지고 있는 서비스의 실행을 결정한다. 따라서 제안하는 시스템이 제공하는 CCM을 통한 서비스의 실행은 상황인지 워크플로우 서비스를 보장한다.

### 3.5 그래픽 기반 시나리오 편집기

본 논문에서는 개발자에게 워크플로우 시나리오 작성을 위한 그래픽 기반 시나리오 편집기를 제공한다. 그림 7은 개발자나 사용자가 상황인지 워크플로우 응용프로그램을 개발하기 위하여 사용하는 CWeditor이다. 시나리오 편집기는 웹 서비스 표준 기술 문서인 WSDL[18]

표 3 UC A와 OC B의 컨텍스트 비교 알고리즘

```
입력 : OCS와 UC
출력 : 서비스 분기 정보로써의 Boolean 값
BEGIN
   i = 0
   foreach OC in ocListener()
      OContextNode o_node = OC
      UContextNode u_node = getContext_Subtree(i)
   foreach left in node
      if(left[o_node] == NULL)||left[u_node] == NULL)then
         exit
      else if((u_node.type == o_type&&u_node.value == o_value)then
         u_node = left[u_node]
         o_node = left[o_node]
      else
         return False
      end-if
   end-foreach
   end-foreach
         return True
END
```

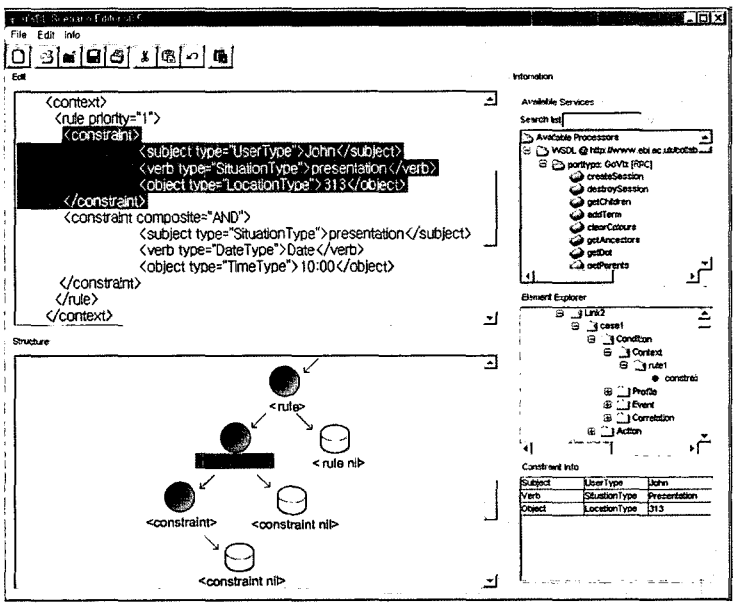



그림 7 그래픽 기반 시나리오 편집기

을 기반으로 현재 상황정보에 가장 적합한 웹 서비스를 검색하여 쉽게 상황인지 워크플로우 서비스 시나리오 작성이 가능하도록 지원한다. CWeditor는 텍스트 기반의 시나리오 편집 창과 트리 형태의 그래픽 기반 시나리오 편집 창을 함께 제공한다. 텍스트 기반 편집 창은 상황인지 워크플로우 개발 언어의 문법에 익숙한 개발자들이 개발 워크플로우의 전체적인 개발 흐름을 파악하며 편집할 수 있는 기회를 제공하며, 그래픽 기반 시나리오 편집 창은 불필요한 문법 구조를 편집 과정에서 제외함으로써 개발 언어 문법에 익숙하지 개발자 및 일반 사용자에게 보다 편리한 시나리오 편집 환경을 제공한다. 또한, 그래픽 기반 시나리오 편집 창은 개발 시나리오의 전체적인 흐름과 문법적 구조를 출력함으로써, 개발자는 실행 과정을 쉽게 파악할 수 있다.

개발 시나리오를 기반으로 상황인지 워크플로우 서비스 엔진이 서비스를 진행하는 가운데 CWparser는 새로운 서비스 요구에 대해 점진적 파싱과 DItree handler를 통해 시나리오를 재구성하고 그 변경 시나리오의 내용 및 구조 정보를 언파싱 규칙을 통해 CWeditor 화면에 출력한다. 개발자는 변경 내용을 확인할 수 있으며, 경우에 따라 CWeditor를 통해 직접 변경이 가능하다. 웹 서비스 목록창은 편집 중인 서비스 컨텍스트에 대한 제약조건(Constraints) 정보를 편집할 수 있는 제약조건 정보 입력창을 가진다. 또한 우측 가운데 보이는 창은 서비스 목록에 대한 트리 구조형태를 보여주는 것이다.

## 4. 실험 및 평가

본 논문에서 제안하는 시스템의 설계 목적은 점진적 파싱기법을 통해 실행 중인 시나리오의 변경을 인식하여 재구성함으로써 워크플로우의 흐름을 중단하지 않고 계속적인 서비스를 제공하는 것이다. 따라서 본 실험에서는 이와 같은 목적을 달성하기 위하여 초기 워크프로우 시나리오를 작성하였으며, 워크플로우 서비스를 받고 있던 사용자가 새로운 서비스를 제공받기 위하여 서비스를 요청하였을 때 이러한 서비스 요구를 초기 워크플로우 시나리오에 동적으로 반영할 수 있는지를 확인하였다.

제안하는 시스템의 구현 환경은 윈도우즈 XP를 탑재한 Pentium 4 3.0Ghz을 이용하였으며, 개발 도구와 플랫폼으로써 JDK 1.4와 eclipse를 사용하였다. 제안하는 시스템의 실험은 uWDL 시나리오에 대해 파싱을 통해 DItree를 출력하고, 새로운 사용자 서비스 요구에 대해 동적으로 DItree를 변경하는 과정과 사용자 상황정보와의 비교를 통해 서비스 분기정보를 생성하는 과정을 보인다. 실험을 위하여 사용자 상황정보에 따라 서비스 분기를 결정하는 회의준비 uWDL[8] 워크플로우 서비스 시나리오를 사용한다.

서비스는 크게 RFID 감지 서비스(RFID Sensing Service), 사용자 검색 서비스(User Search Service), 다운로드 서비스(Download Service)로 나뉜다. RFID 감지 서비스는 RFID 센서로부터 전송되는 컨텍스트 정보(이름, IP 주소 등)를 관리한다. 사용자 검색 서비스는 RFID 감지 서비스로부터 전달받은 컨텍스트 정보를 이용하여 해당 정보가 uWDL에 의해 명시된 규칙(rule)에 적합한 경우 다운로드 서비스를 호출하여 해당 파일을 전송받아 실행시킨다. 이때 개발자는 상황 정보인 컨텍스트 및 프로파일에 따라 어떤 서비스를 선택할 것인지를 결정해야 한다. 이는 uWDL의 <subject>, <verb>, <object> 원소를 이용하여 컨텍스트 및 프로파일의 관계를 기술하게 된다. 표 4는 uWDL 기반의 상황인지 워크플로우 서비스를 위한 시나리오이다.

표 4 동적 사용자 서비스가 요구되는 회의준비 시나리오

1. 회사원 John은 출근 후 자신의 노트북을 이용하여 일정 계획에 아침 10시에 313호 회의실에서 발표와 오후 2시에 316호 상담실에서 고객 S와의 미팅, 그리고 오후 5시에 302호 회의실에서 팀원들과의 결과 회의 계획이 있음을 기록한다.
2. 발표 준비를 하던 John은 9시 40분경에 회의를 위하여 313호 회의실로 이동한다.
3. 313호 회의실 문 위쪽에는 RFID 센서가 설치되어 있어 John의 기본 정보(이름, 노트북의 IP 주소 등)를 서버로 전송한다.
4. 서버에서는 전송된 John의 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 John의 발표 자료를 다운로드하여 해당 프로그램을 실행시킨다.
5. John이 2시에 고객 S와의 고객 상담을 위하여 316호로 이동하는 도중에 315호 상담실에서 3시에 고객 T와의 새로운 상담 일정을 자신이 가진 PDA를 이용하여 입력한다.
6. 고객T와의 새로운 상담 서비스 요구와 그에 대한 서비스 실행 조건을 서버에 전송한다.
7. 서버에서는 초기 시나리오 문서를 기반으로 고객 S와의 고객 상담을 위한 고객 상담 자료를 John의 로컬 컴퓨터로부터 316호 상담실에 있는 컴퓨터로 다운로드한다. 동시에, T와의 새로운 고객 상담 서비스에 대해 초기 시나리오의 DItree에서 변경될 부분을 찾아 점진적 파싱을 통해 새로운 시나리오 DItree를 생성한다.
8. John이 316호 상담실에서 고객 S와의 고객 상담을 마치고 고객 T와의 상담을 위해 오후 3시에 315호 상담실로 이동하면, 서버는 고객 T와의 상담을 위한 고객 상담 자료를 John의 로컬 컴퓨터로부터 315호 상담실의 컴퓨터로 다운로드한다.
9. John이 상담을 마치고 오후 5시에 302호로 이동하면, 서버는 John의 로컬 컴퓨터로부터 업무결과 회의 자료를 302호 회의실에 있는 컴퓨터로 다운로드한다.

표 4에서 사용자 John은 자신이 초기 입력한 업무 일정에 따라 3가지의 서비스를 받게 될 것이다. 초기 작성된 John의 업무 일정은 표 4의 1~4번 그리고 9번에 해당한다. 이 때, John은 업무 일정 서비스를 받고 있는 도중에 자신의 PDA를 통해 새로운 서비스를 요청하여 자신의 상황에 적합한 서비스를 계속해서 받기를 원한다. 이러한 새로운 서비스의 요구는 상황인지 워크플로우 시스템 입장에서는 표 4의 5~8번에 해당하는 새로운 일정 시나리오의 생성을 의미하며, 이러한 변화는 초기 워크플로우 시나리오 문서에 동적으로 적용되어야 한다.

그림 8은 표 4에 기초하여 John의 업무 일정에 따라 초기에 작성된 uWDL 워크플로우 문서와 일정에 해당하는 문서대 대한 DItree를 나타낸다. 그림 8에서 점선 상자로 표시된 부분은 John의 초기 일정에서 두 번째 서비스인 고객 S와 316호에서의 고객 상담을 나타내며, 아래 트리는 시나리오 편집기에 기술된 시나리오 문서의 구조를 나타내는 DItree이다. 상황인지 워크플로우 시스템은 초기에 기술된 워크플로우 시나리오 문서에 기술된 흐름대로 서비스를 제공하는 역할을 담당한다. CWeditor는 이러한 서비스를 용이하게 표현할 수 있고, CWparser는 작성된 시나리오 문서를 파싱하여 DItree를 출력하며, DItree Handler는 CWParser로부터 생성된 DItree를 동적으로 재구성하여 상황인지 워크플로우 시스템이 새롭게 구성된 시나리오를 즉각적으로 서비스 도중에 재적용할 수 있도록 도와준다.

그림 9는 실험 시나리오에 따라 John이 새로 입력한 고객 T와의 고객 상담을 위한 서비스에 대해 새롭게 작성된 uWDL 시나리오 문서의 내용과 그에 대해 동적으

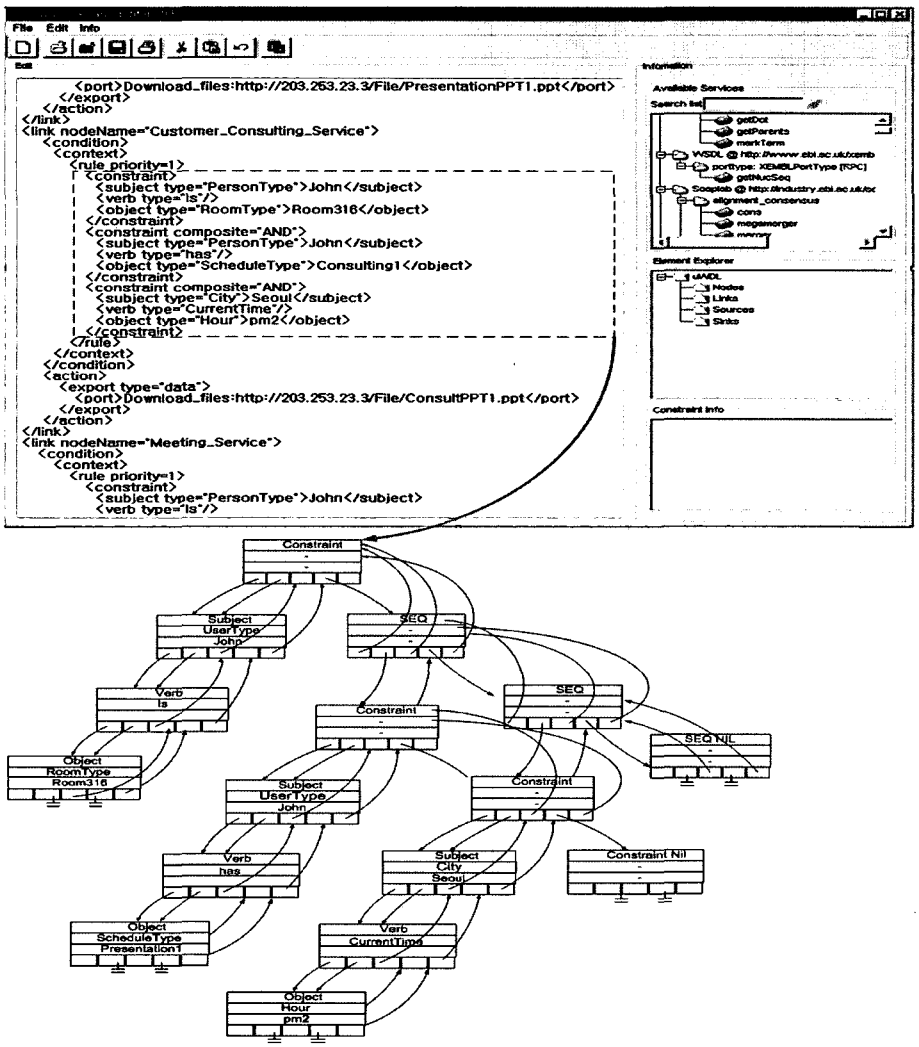



그림 8 초기 실험 uWDL 워크플로우 서비스 시나리오 문서와 DItree의 <constraint> 부분트리

로 재구성된 DItree의 변화를 나타낸다. 그림 9에서 아랫부분의 좌측 DItree는 그림 8에서와 같은 초기 워크플로우 시나리오 문서의 구조를 나타내는 것이며, 우측은 사용자가 새롭게 요구한 서비스에 대한 문서의 구조를 나타낸다. 새롭게 입력된 고객 T와의 고객 상담을 위한 상담자료 다운로드 서비스는 초기 uWDL 문서에 나타난 두 번째 서비스인 고객 S와의 고객 상담 서비스와 세 번째 서비스인 실적 미팅 서비스 사이에 위치하여야 하며, 시나리오 문서에 이러한 내용이 재구성되어 DItree로 새롭게 생성되어야 한다. 이러한 절차는 제안하는 상황인지 워크플로우 시스템의 동적 시나리오 재구성 모듈(DItree Handler)에 의해 수행된다. 또한 그림 9는 워크플로우의 실행을 위한 시나리오 문서의 동적인 재구성이 부분 트리(DItree)의 변경을 통하여 이루어졌음을 보여준다.

시나리오 기반의 기존 상황인지 워크플로우 시스템들[9,10]은 본 논문에서와 같이 워크플로우를 이용하여 사용자에게 서비스를 제공하고 있으나, 현재 서비스를 제공받고 있는 도중에 사용자의 새로운 서비스 요구에 대한 즉각적인 서비스 대응을 위한 적절한 처리 방법을 마련하지는 못 하였다. 그러나 본 실험에서는 시나리오 기반의 상황인지 워크플로우 시스템에 점진적 파싱 기법[18]을 적용하여 초기 시나리오 문서를 동적으로 재구성함으로써 유비쿼터스 환경에서 다양하게 발생하는 사용자의 서비스 요구에 적합한 서비스를 제공할 수 있는 새로운 방법을 제안하였다.

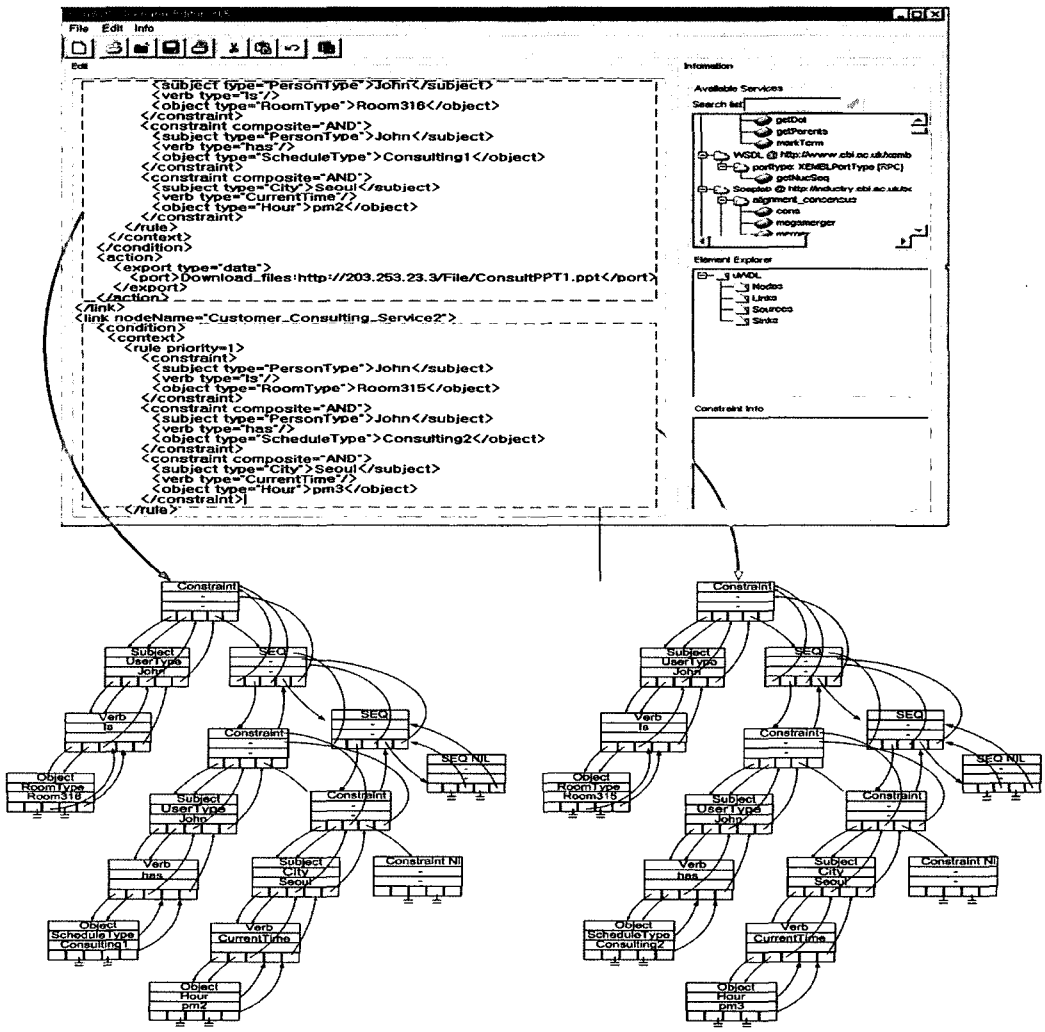



그림 9 실험 시나리오에 따라 사용자가 PDA를 통해 입력하는 새로운 서비스에 대한 초기 uWDL 시나리오 문서의 변경과 DItree에서의 <constraint> 부분트리의 동적인 변화

## 5. 결 론

유비쿼터스 컴퓨팅 환경에서 상황인지 워크플로우 시스템은 다양한 도메인에서 동적으로 발생할 수 있는 상황 정보와 함께 사용자의 새로운 요구조건을 받아, 실행중인 워크플로우 시스템에 적용하여 사용자에게 보다 적합한 서비스를 제공할 수 있어야 한다. 현재까지 시도되었던 시나리오 기반의 워크플로우에 관련된 연구사례[7,9,10]들은 이러한 요구 사항에 대한 적절한 대안을 마련하지 못하고 있다. 따라서 본 논문에서는 이러한 문제를 해결하기 위하여본 논문에서는 사용자가 입력하는 새로운 서비스에 대한 요구사항을 실행 중인 워크플로우 시나리오에 즉각적으로 반영하여 기존에 제공되고 있던 서비스를 중단시키지 않고 새롭게 적용된 서비스를 제공하는 유비쿼터스 환경에서의 상황인지 워크플로우 시스템을 제안하였다.

제안한 시스템은 점진적 파싱 기법[17]을 통해 실행중인 시나리오의 변경 위치를 정확히 인식하고 시나리오를 신속히 재구성함으로써 워크플로우 흐름을 중단하지 않고 계속적인 서비스를 제공할 수 있다. 이를 위하여 "동적 사용자 서비스가 요구되는 시나리오"를 처리할 수 있는 시스템을 위한 "점진적 파싱 알고리즘"을 설계하였으며, 현재 상황정보에 가장 적합한 서비스를 검색하여 쉽게 uWDL 워크플로우 서비스 시나리오를 작성 가능하도록 지원하는 "GUI 기반의 시나리오 편집기"를 개발하였다. 또한 본 논문에서 제안한 시스템은 동적인 사용자 서비스의 요구를 만족시킴으로써, 정적인 시나리오에 대한 사용자 서비스를 제공하였던 기존의 uFlow 시스템[7]보다 동적인 서비스 제공 측면에서 그 성능이 개선되었다. 그리고 본 시스템에서 적용하고 있는 점진적 파싱 기법은 새롭게 떠오르는 기술은 아니지만 기존의 기술을 유비쿼터스 컴퓨팅 환경에서 사용되고 있는 워크플로우 기술에 접목시킴으로써 다양한 서비스 도메인을 가진 유비쿼터스 환경에서 동적으로 발생할 수 있는 사용자의 요구에 대한 서비스를 제공할 수 있는 새로운 기술로서의 효용성을 높일 수 있었다.

앞으로 본 시스템은 다양한 서비스를 제공하기 위하여 여러 가지 서비스 도메인에서의 시나리오에 대한 추가적인 연구가 요구되며, 서비스 참여자들 상호간 협업이 중요시 되는 커뮤니티 컴퓨팅[19] 환경에서 상황인지 정보를 기반으로 상황변화를 스스로 인식하여 사용자에게 동적으로 다양한 워크플로우 서비스를 제공하기 위한 연구가 지속되어야 할 것이다.

## 참 고 문 헌

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

[2] L. Ardissono, A. Di Leva, G. Petrone, M. Segnan, M. Sonnessa, "Adaptive Medical Workflow Management for a Context-Dependent Home Healthcare Assistance Service," Electronic Notes in Theoretical Computer Science, Elsevier, pp. 59-68, 2006.

[3] By Prof. Dr. Frank Leymann, "Web Services Flow Language (WSFL 1.0)," Distinguished Engineer Member IBM Academy of Technology, IBM Software Group, May 2001.

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

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

[6] Anders Møller and Michael I. Schwartzbach, "An Introduction to XMl AND Web Technologies," Addison-Wesley, ISBN: 0321269667, January 2006.

[7] Joo Han, Eun Kim, Yong Cho, Jaey Choi, "A Ubiquitous Workflow Service Framework," LNCS 3983, pp.30-39, 2006.

[8] J Han, Y Cho and J Choi, "Context-aware Workflow Language based on Web Services for Ubiquitous Computing," LNCS 3481 - ICCSA 2005, pp. 1008-1017, Springer, 2005.

[9] Anand Ranganathan, Scott McFaddin, "Using Workflows to Coordinate Web Services in Pervasive Computing Environments," Proceedings of the IEEE International Conference on Web Services, ICWS'04, pp. 189-197, 2004.

[10] Jun Li, Yingyi Bu, Shaxun Chen, Xianping Tao, Jian Lu, "FollowMe: On Research of Pluggable Infrastructure for Context-Awareness," 20th International Conference on Advanced Information Networking and Applications(AINA'06), Volume 1, pp. 199-204, 2006.

[11] D. Hollingsworth, "The Workflow Reference Model," Technical Report TC00-1003, Workflow Management Coalition, 1994.

[12] Amit Sheth and Krys J. Kochut, "Scalable and dynamic work coordination and collaboration systems," In Dogac A., Kalinichenko L., Tamer Ozsu M., and Sheth A., editors, Workflow Applications to Research Agenda, Istanbul, Turkey, August 1997. NATO Advanced Study Institute.

[13] Workflow Management Coalition, The Workflow Handbook 2002, chapter 1. Future Strategies Inc. and Lighthouse Point, FL, USA., 2002.

[14] Pedro Vieira, Antonio Rito-Silva, "Adaptive Workflow Management in WorkSCo," 16th International Workshop on Database and Expert Systems Applications (DEXA'05), pp. 640-645, 2005.

[15] Dragos-Anton Manolescu. Micro-Workflow: A Workflow Architecture Supporting Compositional

Object-Oriented Software Development. PhD thesis, University of Illinois, 2000.
[16] P Dias, Pe Vieira, and Ant Rito-Silva. "Dynamic evolution in workflow management systems," In Proceedings of the 3rd International Workshop on Web Based Collaboration (WBC2003), Prague, Czech Republik, 2003. IEEE.
[17] C. Ghezzi and D. Mandrioli. "Incremental Parsing," ACM Transactions on Programming Languages and Systems, 1(1):58-70, 1979.
[18] WSDL, "Web Services Description Language (WSDL) 1.1", W3C Note, http://www.w3.org/TR/wsdl, 15 March 2001.
[19] Toru Ishida Ed. Community Computing: Collaboration over Global Information Net- works. John Wiley and Sons, 1998.

최 종 선

2000년 숭실대학교 컴퓨터학부(학사). 2002년 숭실대학교 컴퓨터학과(공학석사). 2003년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 유비쿼터스 컴퓨팅, 고성능 컴퓨팅, 시스템 소프트웨어

조 용 윤

1995년 시립 인천대학교 전자계산과 졸업(학사). 1998년 숭실대학교 대학원 전자계산학과 졸업(공학석사). 2006년 숭실대학교 컴퓨터학과(공학박사). 2006년~현재 숭실대학교 정보미디어 연구원. 관심분야는 프로그래밍 언어, 컴파일러, XML, HCI, 유비쿼터스 컴퓨팅

최 재 영

1984년 서울대학교 제어계측공학과(학사) 1986년 미국 남가주대학교 컴퓨터공학과(공학석사). 1991년 미국 코넬대학교 컴퓨터공학과(공학박사). 1992년~1994년 미국 국립 오크리지연구소 연구원. 1994년~1995년 미국 테네시 주립대학교 연구교수. 1995년~현재 숭실대학교 컴퓨터학부 교수. 관심분야는 유비쿼터스 컴퓨팅, 그리드 컴퓨팅, 병렬/분산처리, 클러스터링, 시스템 소프트웨어