DOI: 10.3745/KIPSTA.2010.17A.2.093

복합 워크플로우 서비스를 위한 CAWL 기반 상황인지 워크플로우 시스템

최 종 선[†] · 조 용 윤[††] · 최 재 영[†††]

요 약

유비쿼터스 환경에서는 사용자의 주변 환경에서 발생할 수 있는 상황이 매우 다양하므로, 이에 대응하기 위한 자동화 서비스의 개발이 요구된다. 그러나 기존의 상황인지 기반 워크플로우 시스템들은 단일 워크플로우 서비스만을 제공할 수 있으므로, 다수의 워크플로우 조합을 통해 복합적이고 다양한 서비스를 제공하는데 제약이 있다. 이를 위해 본 논문에서는 다수의 워크플로우에 존재하는 개별적인 서비스 흐름을 하나의 워크플로우로 통합 표현할 수 있는 상황인지 기반의 워크플로우 언어인 CAWL(Context-Aware Workflow Language)를 기반으로 하는 워크플로우 시스템을 제안한다. 제안하는 시스템은 CAWL를 이용하여 작성한 시나리오를 바탕으로 사용자에게 다양한 복합 워크플로우 서비스를 제공할 수 있다. 또한 각각 존재하는 다수의 워크플로우를 복합 워크플로우 서비스를 구성하기 위한 일부로써 재사용할 수 있으므로, 자동화 서비스 개발의 효율성을 증대시킬 수 있다.

키워드 : 유비쿼터스 컴퓨팅, 상황인지 워크플로우 언어(CAWL), 복합-워크플로우 서비스, 워크플로우 시스템

A CAWL-based Context-Aware Workflow System for Composite Workflow Services

Jongsun Choi[†] · Yongyun Cho[††] · Jaeyoung Choi[†††]

ABSTRACT

There are many complicated situations which could be occurred in users' surroundings, so it is required to develop automation services to provide users with appropriate services in ubiquitous computing environments. However, most of the current context-aware workflow systems express context-aware services only with a single workflow. Therefore, they have difficulties in providing users with various and composite services by combining different workflows. In this paper we propose a CAWL-based context-aware workflow system, where CAWL is a context-aware worflow language to express a composite workflow model by describing individual service workflows. The proposed system can provide users with various composite workflow services based on a service scenario, which is described with CAWL. And by reusing a number of single workflows to construct composite workflow services, it is possible to save time and effort to develop context-aware workflows.

Keywords : Ubiquitous Computing, Context-Aware Workflow Langueage(CAWL), Composite Workflow Services, Workflow System

1. 서 론

워크플로우 기술의 중요성은 기계 중심의 분산 컴퓨팅 환경으로부터 인간 중심의 유비쿼터스 컴퓨팅 환경으로 점차 옮겨지고 있다. 이와 같은 환경에서 사용자의 다양한 요구사항을 만족시키기 위해서는, 사용자와 주변 환경에 대한 정보(상황정보)를 바탕으로 사용자에게 적합한 서비스를 자동으로 제공할 수 있어야 한다. 그러나 유비쿼터스 환경에서는 사용자의 주변 환경에서 발생할 수 있는 상황이 매우 복잡 다양하므로, 이에 대응하기 위해서는 복합적인 다양한 상황에 적응할 수 있는 자동화된 서비스의 개발이 이루어져야 한다. 또한 개발된 자동화 서비스들은 제공의 효율성 측면에서 기존에 만들어진 서비스에 대한 재사용성을 반드시 고려해야 한다. 컴퓨팅 환경의 변화에 따른 요구사항에 발

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(NIPA-2009-(C1090-0902-0007)).
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 정 회 원 : 순천대학교 정보통신공학부 조교수
††† 종신회원 : 숭실대학교 정보과학대학 컴퓨터학부 교수(교신저자)
논문접수 : 2009년 12월 10일
수 정 일 : 1차 2010년 2월 16일
심사완료 : 2010년 3월 17일

맞추어, 최근 대표적인 비즈니스 프로세스 자동화 모델인 워크플로우를 유비쿼터스 컴퓨팅 환경에 적용해야 하는 필요성이 대두되고 있다. 이와 더불어 상황인지 기반의 워크플로우 모델을 유비쿼터스 컴퓨팅 환경에 적용시키는 연구들이 진행되고 있다 [1, 2, 4, 7]. 그러나 기존 연구들은 프로토타입 시스템을 구현하는 수준에 있으므로, 복합적이며 동적인 사용자 주변 환경에 대응하여 적합한 서비스를 제공하기에는 시스템의 구현측면에서 아직 개선의 여지가 많다.

상황인지 기반의 워크플로우 시스템은 유비쿼터스 컴퓨팅 환경에서 자동화 서비스를 제공하기 위한 것이며, 사전에 기술한 서비스 시나리오를 바탕으로 주변 환경에서 발생하는 상황정보에 따라 사용자에게 적합한 서비스를 제공할 수 있어야 한다 [7]. 또한 사용자의 주변 환경은 다양한 서비스 도메인으로 분류될 수 있으며, 도메인마다 사용자의 요구에 따른 이질적인 서비스들이 존재할 것이다. 그러므로 이와 같은 환경에서의 상황인지 기반 시스템은 단순 자동화 서비스보다는 여러 서비스로 조합된 형태를 가진 복합적인 자동화 서비스를 제공해야 할 것이다. 그러나 기존의 상황인지 기반 워크플로우 시스템들은 단일 워크플로우 서비스만을 제공할 수 있으므로, 다수의 워크플로우 조합을 통해 복합적이고 다양한 서비스를 제공하는데 제약이 있다 [5, 6, 8]. 이와 같은 문제점을 해결하기 위해, 본 논문에서는 다수의 워크플로우에 존재하는 개별적인 서비스 흐름을 하나의 워크플로우로 통합 표현할 수 있는 상황인지 기반의 워크플로우 언어인 CAWL(Context-Aware Workflow Language)를 기반으로 하는 워크플로우 시스템을 제안한다. 제안하는 시스템은 상황인지 기반의 복합 워크플로우 모델을 지원하는 CAWL을 이용하므로, 이를 바탕으로 기술한 서비스 시나리오 문서를 처리하여 사용자의 다양한 서비스 요구에 적합한 복합 워크플로우 서비스를 제공할 수 있다. 또한 제안하는 시스템을 통해 개발자는 각각 존재하는 다수의 워크플로우를 복합 워크플로우 서비스를 구성하기 위한 일부로써 재사용할 수 있으므로, 자동화 서비스 개발의 효율성을 증대시킬 수 있는 이점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 워크플로우 기술의 동향 및 상황인지 기반 워크플로우 미들웨어에 대하여 살펴본다. 3장에서는 제안하는 시스템의 설계 및 구조에 대하여 기술한다. 4장에서는 제안하는 시스템에 적용된 워크플로우 모델에 대하여 설명한다. 그리고 5장에서는 제안하는 시스템의 효용성을 위해 시나리오를 통한 응용실험에 대하여 기술하고, 마지막으로 6장에서는 결론에 대하여 언급한다.

2. 관련 연구

최근 비즈니스 프로세스와 모델을 위한 워크플로우 기술은 다양한 서비스 도메인에서 활용되고 있으며, 이러한 기술을 유비쿼터스 컴퓨팅 환경에 적용하려는 연구들이 활발하게 이루어지고 있다 [7]. 유비쿼터스 컴퓨팅 환경은 서비스에 대한 사용자의 요구가 매우 광범위하고 다양할 수 있으므로, 유동적인 비즈니스 프로세스 도메인보다 더욱 복잡하고 다양하게 구성되어 있다. 이와 같은 환경으로 적응하기 위한 도전은 상황인지 워크플로우 [2, 3, 6, 8] 또는 스마트 워크플로우 [4] 등과 같은 개념으로 시도되고 있다. 그리고 이를 기반으로 하는 워크플로우 시스템들은 대부분 프로토타입 수준에서 진행되고 있다 [1, 2, 4, 7, 13, 14]. 그러나 사용자의 서비스 요구에 따라 다수의 워크플로우를 통합하여 복합 워크플로우 서비스를 제공하는 상황인지 기반의 워크플로우 시스템에 대한 연구는 시도되지 않고 있다.

상황인지 워크플로우 미들웨어에 대하여 좀 더 상세히 살펴보도록 하자. 상황인지 기반의 워크플로우 엔진을 제안한 연구 [5]에서는 컨텍스트 모델링과 유비쿼터스 환경에서 네트워크에 연결된 디바이스의 제어와 리즈닝(reasoning) 메커니즘에 초점을 맞추고 있다. FollowMe [8]는 편재형 컴퓨팅 환경을 위한 OSGi 프레임워크로써, 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 장점을 가진다. 이 연구의 초점은 디바이스의 제어와 사용자 어플리케이션에 국한되어 있다.

Context4BPEL [3]은 상황인지 기반의 워크플로우 개념을 위해 WS-BPEL [11]을 확장하였으며, 이를 바탕으로 Context4BPEL의 실행환경을 위한 아키텍처를 제안하고 있다. 그러나 WS-BPEL을 확장하였으므로 이를 처리하기 위한 워크플로우 엔진의 수정이 불가피하다. 따라서 WS-BPEL의 개발자가 아닌 경우 이를 구현하는데 어려움이 있다. 그러므로 Context4BPEL의 실행 환경을 구축하기 위한 워크플로우 엔진에 대한 연구가 더 필요하다. 그리고 Context4BPEL을 기반으로 스마트 워크플로우 개념을 제안한 연구 [4]에서는 컨텍스트 제공, 컨텍스트 처리 엔진, 워크플로우 정의로 계층화된 시스템 모델을 제시하고 있다.

uFlow [6]는 uWDL [9]을 바탕으로 구축한 상황인지 워크플로우 서비스 프레임워크이다. uWDL은 기존 웹 서비스 기반 워크플로우 언어들이 서비스 전이 조건으로 상황정보를 표현하지 못하는 점을 해결하기 위해 제안한 유비쿼터스 워크플로우 언어이다. uWDL은 워크플로우가 상황인지를 바탕으로 상태를 전이할 수 있도록 워크플로우의 상태 전이 조건에 컨텍스트 정보를 표현하고 있으며, 이를 위해 uWDL은 RDF 형식의 {주어, 동사, 목적} 형태를 가진 객체화 모델을 나타내는 구조적 컨텍스트 모델 [6]을 기반으로 한다. 그러나 uWDL은 시나리오 문서 내에 상황정보를 정적으로 표현하고 있으며, 순차적인 하나의 워크플로우 만을 표현할 수 있다. 따라서 uWDL은 상황정보의 동적 표현 및 다수의 워크플로우를 표현하는데 한계가 있다.

상황인지 워크플로우의 적응에 관한 연구 [7]에서는 상황인지 워크플로우와 관련된 연구 사례를 통해 다양한 접근방법에 대하여 비교 분석하였다. 이 연구에서는 상황인지 워크플로우의 적응 수준을 추상(워크플로우의 추상적 정의),

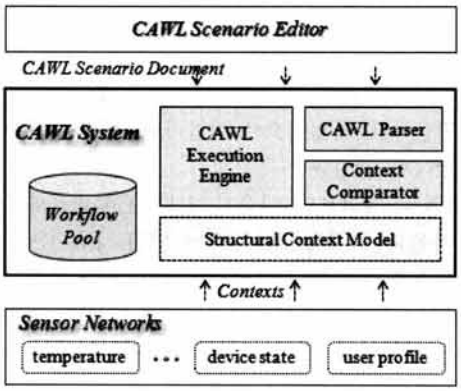
구체적 워크플로우(워크플로우 엔진에 의해 실행 가능한 워크플로우의 정의) 그리고 실행 가능한 워크플로우 인스턴스로 나누어 설명하고 있으며, 각각의 수준에 해당하는 접근 방식을 기존 연구 사례를 통해 설명하고 있다. 예를 들어, ContextBPEL는 추상적인 워크플로우에 해당하며, 본 논문에서 제안하는 시스템을 포함하여 FollowMe, uFlow 등은 구체적 워크플로우에 초점을 두고 있으며, 이를 설명하기 위해 시나리오(추상 정의) 및 워크플로우 실행(워크플로우 인스턴스)을 언급한다.

CAWL [10]은 복합 워크플로우 모델을 바탕으로 여러 사용자에게 다중 워크플로우 서비스를 제공할 수 있는 기능을 지원하는 상황인지 워크플로우 언어이다. CAWL은 uWDL에서 드러난 문제점을 해결하기 위해 동적으로 발생하는 컨텍스트 정보를 표현할 수 있도록 하였으며, 서브-워크플로우(subworkflow)[1] 개념을 추가하여 복합 워크플로우 기능을 지원하였다. 다시 말해서, CAWL을 바탕으로 다수의 워크플로우 서비스를 하나의 시나리오 문서에 표현함으로써 좀 더 큰 규모의 복합 워크플로우 서비스를 지원할 수 있게 되었다. 본 논문에서는 앞서 언급한 CAWL의 개선된 기능을 바탕으로 다양한 서비스 시나리오 문서를 처리하고, 복합 워크플로우 서비스를 제공하기 위한 워크플로우 시스템(CAWL System)을 제안한다.

3. CAWL 시스템

여러 서비스 도메인이 공존하는 유비쿼터스 컴퓨팅 환경에서 사용자에게 상황인지 서비스를 제공하기 위해서는, 다양한 서비스를 표현하고 이를 처리할 수 있는 컴퓨팅 환경이 필요하다. 이에 본 논문에서는 여러 워크플로우의 흐름을 하나의 흐름으로 통합하여, 보다 큰 규모의 복합 워크플로우 서비스를 제공하기 위한 CAWL 기반의 상황인지 워크플로우 시스템(이하 CAWL 시스템)을 제안한다. CAWL 시스템은 기존의 여러 워크플로우 서비스를 조합하여 복합 워크플로우 서비스를 구성할 수 있도록 지원하는 CAWL를 기반으로, 다양한 복합 워크플로우 서비스를 제공할 수 있다.

(그림 1)은 제안하는 시스템의 전체 구조를 나타내는 개념도이다. CAWL 시스템은 센서 네트워크를 통해 얻은 사용자 주변의 상황정보(사용자 프로파일, 장비 상태, 온도, 위치 등)와 시나리오 편집기(CAWL Scenario Editor)를 이용하여 작성한 사용자 서비스 시나리오 문서(CAWL Document)를 입력정보로 활용한다. 또한 이와 같은 입력 정보를 바탕으로 사용자에게 상황인지 기반의 복합 워크플로우 서비스를 제공하는 역할을 수행한다. 시나리오 편집기는 서브, 병렬 워크플로우 패턴과 같은 복합 워크플로우 모델(4절 참고)을 이용하여, 여러 워크플로우를 하나의 흐름으로 통합할 수 있는 상황인지 기반 워크플로우 언어인 CAWL을 이용하여 서비스를 표현한다. 그림에서 CAWL 시스템은 입력된 워크플로우 문서를 파싱하기 위한 상황인지 워크플로우 파서(CAWL parser), RDF 형식으로 작성된 시나리오 문서에서의 상황 정보와 센서에서 들어온 정보를 가공한 상황 모델을 비교하여 사용자가 처한 환경이 시나리오에 작성된 상황 정보와 같은지를 검증하는 상황 비교기(Context Comparator), 그리고 시나리오에 기술된 서비스 실행 조건에 따라 서비스를 실행시키기 위한 실행 엔진(CAWL Execution Engine), 마지막으로 기존에 작성된 시나리오 문서를 파싱하여 얻은 결과, 즉 AST(Abstract Syntax Tree) 객체를 저장하기 위한 풀(Workflow Pool)로 구성된다.

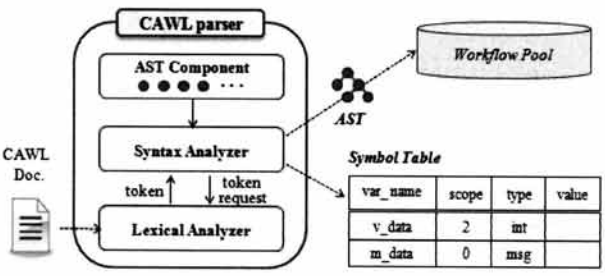


(그림 1) CAWL 시스템 아키텍처

3.1 워크플로우 파서 (CAWL Parser)

CAWL 파서는 복합 워크플로우 모델을 지원하는 상황인지 워크플로우 언어(CAWL)을 이용하여 작성한 시나리오를 파싱하는 과정에서 크게 2가지 기능을 수행한다. 첫 번째 기능은 다수의 워크플로우를 복합적으로 구성하는 과정에서의 재사용성에 관한 것으로, 이를 위해 워크플로우를 저장(Workflow Pool)하는 것이다. 두 번째 기능은 시나리오 문서의 관리적인 측면으로, 다수의 워크플로우 문서에 기술한 변수들을 효율적으로 관리하기 위한 심볼 테이블(Symbol Table)을 구성하는 것이다.

(그림 2)에서 CAWL 파서는 3가지 모듈로 구성된다. 첫째, AST 컴포넌트(AST Component)는 시나리오에 기술되어 있는 요소(elment)를 나타내는 것으로 AST 트리의 각 노드에 해당하는 객체 단위로 구성된다. 둘째, 어휘 분석기(lexical Analyzer)는 AST와 Symbol Table을 생성하기 위



(그림 2) CAWL 파서 구성도

1) 서브 워크플로우는 미리 정의된 다른 하나의 워크플로우에 속한 서비스 노드를 표현하기 위한 개념으로, 구조적 프로그래밍 방법론에서 한 루틴에서 또 다른 루틴(subroutine)을 호출하는 메커니즘과 동일한 개념이다.

해 시나리오 문서를 토큰 단위로 인식한다. 마지막으로 구문 분석기(Syntax Analyzer)는 어휘 분석기에서 인식한 토큰(token)을 이용하여 문서의 구조에 대한 문법적인 이상 유무를 판단하고, 문법적으로 적합하면 AST 컴포넌트를 이용하여 AST를 생성한다.

파싱과정을 살펴보면 다음과 같다. CAWL 파서의 입력 정보로 사용되는 시나리오 문서(CAWL Doc.)는 파서에서 어휘 분석기를 통해 토큰 단위로 읽혀 구문 분석기로 전달된다. 전달된 토큰은 구문분석기에서는 CAWL 스키마에 정의된 문법 구조에 따라 문서의 토큰 규칙을 검사한다. 그리고 구문분석기는 구문분석 과정에서 처리되는 문서의 엘리먼트 단위와 일치하는 AST 컴포넌트의 조합으로 AST를 생성하며, 또한 시나리오에서 사용되는 변수의 이름, 유효범위, 값 등을 저장하는 심볼 테이블을 생성한다.

3.1.1 심볼 테이블(Symbol Table)과 AST 생성

심볼 테이블은 시나리오 문서에 기술된 변수에 대한 접근성을 높여, 각 변수들을 효율적으로 관리하기 위해 생성된다. (그림 2)에서의 구문 분석기는 어휘 분석기로부터 전달받은 토큰을 바탕으로 토큰열(token-sequence)이 CAWL 스키마에 적합한지를 판단한다. 이와 같이 스키마에 적합한 경우 시나리오 문서에서 기술된 변수에 대한 정보를 심볼 테이블에 저장한다. 다시 말해서, 파싱하는 과정에서 시나리오 문서에서 사용되는 내부 변수 또는 외부 서비스(다른 워크플로우 서비스)와의 메시지 전달을 위해 사용되는 메시지 변수에 관한 정보를 심볼 테이블에 저장한다.

(그림 3)은 심볼 테이블의 구성과 AST의 생성과정을 보여준다. 심볼 테이블에 저장되는 변수는 크게 심플 타입(Built-in Simple Type)의 변수(v_data)와 사용자 정의에 의해서 사용되는 컴플렉스 타입(Complex Type)의 메시지 변수(m_data)가 있으며, 변수에 관한 정보는 이름(var_name), 유효범위(scope), 값(value)으로 구성되어 있다. 변수 정보 중에 scope은 일반적인 프로그래밍 언어에서 변수의 유효범위(variable scope)과 같은 개념으로, 최상위 노드는 0, Flow는 1, 마지막으로 Node 수준은 2로 정의하여 사용한다.

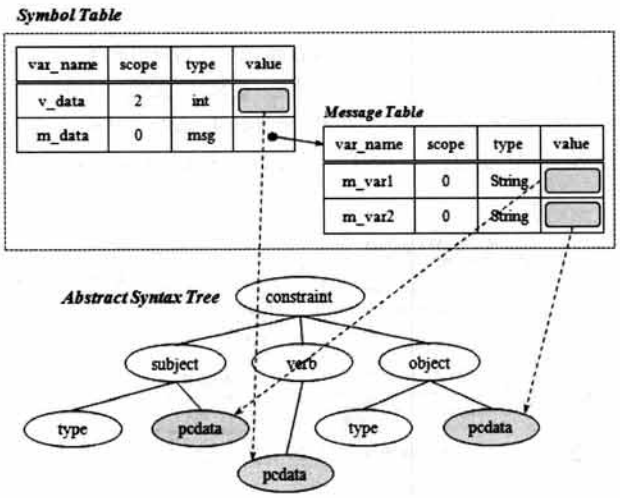
심플 타입의 변수는 해당 값을 테이블에 저장하고, 메시지 변수는 사용자가 정의한 메시지 타입(msg)과 함께 값(value)에 참조할 변수의 정보를 가진다. 예를 들면, (그림 3)은 v_data 변수는 유효범위가 2이고, 정수(int) 타입을 사용하는 변수 값을 갖는다. 그리고 m_data 메시지 변수는 유효 범위가 0이고, 사용자가 정의한 msg 타입을 사용하는 실제 변수에 대한 참조 정보를 갖는다. 다시 말해서, 심볼 테이블에서 msg 타입에 해당하는 실제 값은 메시지 테이블(*Message Table*)에 대한 링크(link) 정보만을 저장하고, 실제 전달되는 메시지의 내용은 메시지 테이블에 저장한다.

그림에서 AST(Abstract Syntax Tree)는 시나리오 문서 전체를 나타내는 시나리오 문서 인스턴스 트리(DITree: Document Instance AST)의 일부분을 나타낸다. AST 구조에서 최상위 노드(constraint)는 사용자의 상황정보를 시나리오에 기술하기 위한 단위로써, RDF를 기반으로 {subject, verb, object} 형태의 엔티티(entity)들에 대한 집합으로 표현된다. 심볼 테이블에서 변수의 값은 AST의 pcdata를 구성하고, 메시지 변수의 경우에는 참조 테이블(*Message Table*)에 저장되어 있는 변수의 값이 AST의 pcdata를 구성하게 된다. 그리고 이들 AST에 저장되는 값들은 컨텍스트 비교기의 입력 값으로 활용된다.

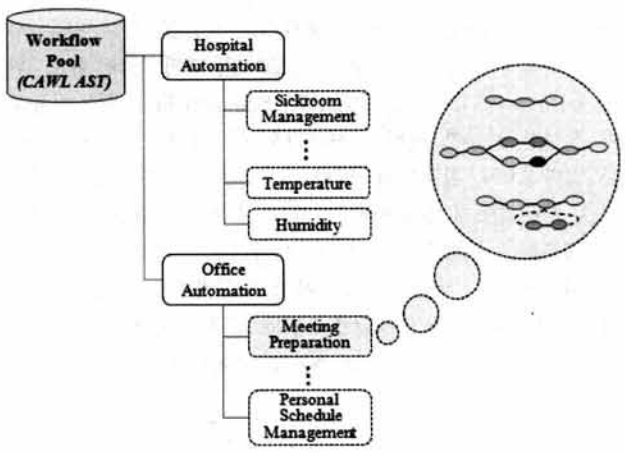
3.1.2 워크플로우 풀 (Workflow Pool)

기존에 작성된 워크플로우 시나리오에 대한 효율성을 높이기 위해서는 워크플로우의 분류 및 검색이 가능한 워크플로우 풀이 요구된다. 이를 위해 CAWL 시스템에서는 워크플로우 시나리오를 AST 형태로 저장할 수 있는 워크플로우 풀을 제공한다. 워크플로우 풀은 시스템이 새로운 워크플로우 서비스를 제공할 때 워크플로우 엔진에서 시나리오 문서에 대한 파싱과정을 거치지 않고, 이미 파싱된 결과 값인 워크플로우의 AST를 객체 형태로 저장하고 있다가, 나중에 이를 재사용하기 위한 것이다.

(그림 4)는 워크플로우 풀의 구성 방법을 나타낸다. 워크



(그림 3) 심볼 테이블 구성 및 AST 생성 과정



(그림 4) 워크플로우 풀 구성

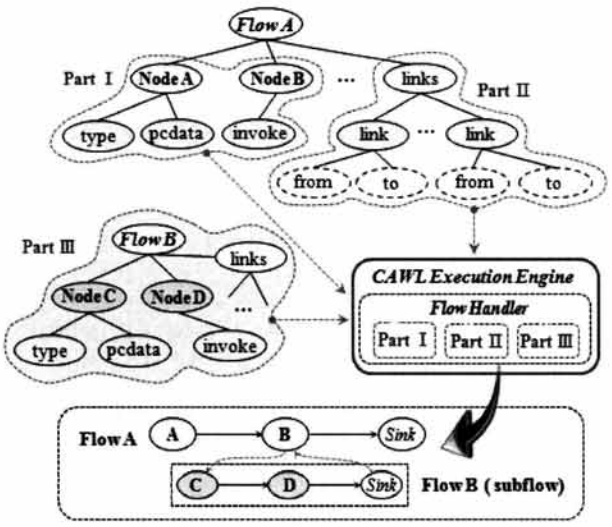
플로우 풀은 병원, 사무실과 같이 서비스 도메인에 따라 네이밍(naming)을 통하여 세부적으로 서비스를 분류하고, 이에 해당하는 실제 워크플로우를 일련의 서비스 집합인 Flow 단위의 AST를 저장한다. 그리고 저장된 각각의 워크플로우 시나리오는 재사용을 위해 각 카테고리에 해당하는 워크플오우 서비스의 위치를 시나리오 테이블에 저장한다. 저장된 각각의 시나리오는 카테고리에 따라 분류되어 시스템에서 검색을 통하여 이용될 수 있다.

3.2 워크플로우 흐름 처리기 (Flow Handler)

워크플로우는 일련의 연산 혹은 작업들을 기술한 것으로 일련의 연산 혹은 작업들을 순서대로 처리할 수 있는 방법이 요구된다. 이에 CAWL 시스템은 워크플로우 실행 엔진을 제공한다. 이는 상황인지 개발 환경에서 CAWL을 이용하여 작성한 서비스 시나리오를 처리하여 사용자가 처한 상황에 알맞은 웹서비스를 제공한다.

(그림 5)는 워크플로우 서비스의 처리 과정을 나타낸다. 워크플로우 엔진의 주요 모듈은 워크플로우의 실행 및 흐름을 제어하는 워크플로우 흐름 처리기(Flow Handler, 이하 F_Handler)이다. F_handler는 파싱과정에서 생성된 AST를 순회(traverse)하며 각 노드의 링크 정보를 참조하여 시나리오에 기술된 워크플로우 흐름을 처리한다. 워크플로우 실행 엔진의 입력 정보는 Part Ⅰ, Part Ⅱ, 그리고 Part Ⅲ로 구성된다. 좀 더 구체적으로 살펴보면 다음과 같다. Part Ⅰ은 Flow A의 서비스 노드 A, B의 정보를 나타낸다. Part Ⅱ는 Flow A의 노드 간 연결 정보를 나타내며, Part Ⅲ는 Flow A의 서브 워크플로우 정보인 Flow B의 노드와 링크 정보를 나타낸다.

F_Handler는 Part Ⅰ, Part Ⅱ, 그리고 Part Ⅲ를 바탕으로 복합 워크플로우 서비스 제공을 위해 Flow A와 Flow B를 통합하여 워크프로우를 처리한다. F_Handler는 파싱과정에서 생성된 AST에서 <links> 요소를 참조하여 <link> 요소의 속성인 from과 to의 정보를 이용하여 각 서비스 노드 간의 연결 처리를 한다. <invoke> 요소는 같은 시나리오 문서 혹은 다른 문서에 존재하는 Flow 참조하는데 사용한다. 다른 문서에 있는 Flow를 참조하는 경우, 이를 서브플로우(subflow)로 지칭한다. 따라서 전체 워크플로우는 서브플로우를 <invoke> 요소를 이용하여 연결 및 통합되고, 이는 복합 워크플로우로 구성된다. 이와 같은 통합 과정은 일반적인 프로그래밍 기법에서 루틴(routine)이 서브루틴(subroutine)을 호출하는 개념과 동일하다. (그림 5)의 하단 부분은 서브플로우(Flow B)가 호출되어 Flow A와 통합된 모습을 보여주는 복합 워크플로우를 나타낸 것이다.



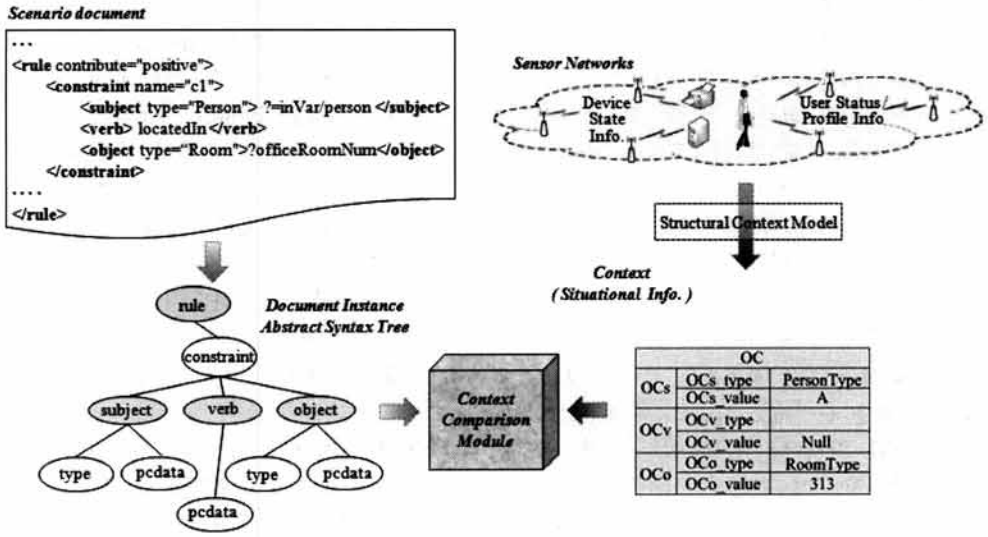
(그림 5) 워크플로우 처리 과정

3.3 컨텍스트 비교기(Context Comparator)

유비쿼터스 컴퓨팅 환경에서 사용자의 상황정보에 적합한 서비스를 제공하기 위해서는 센서에서 발생하는 상황 정보와 시나리오 문서에 기술한 상황정보를 비교 및 판단할 필요가 있다. 이를 위해 CAWL 시스템은 컨텍스트 비교기를 제공한다.

컨텍스트 비교기의 역할은 AST의 상황정보, 즉 시나리오 문서에 RDF 형식으로 기술한 상황정보와 센서로부터 들어온 컨텍스트 정보를 구조적 컨텍스트 모델(structual context model) [6]을 기반으로 가공한 상황정보가 일치하는지 검증하는 것이다. 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 기반의 구조적 컨텍스트 모델을 통해 {주어, 동사, 목적어}로 구성된 엔티티의 집합으로 객체화된다.

(그림 6)은 상황정보의 비교과정을 나타낸다. 컨텍스트 비교 모듈의 입력 정보는 좌측의 시나리오 정보와 그림 우측의 객체화된 컨텍스트 (OC: Objectified Context) 정보이다. (그림 6)에서 시나리오 내용은 "Person에 해당하는 사람이 OfficeRoomNum에 위치하는가?"를 판단하는 내용을 기술한 것이다. 이때 Person과 OfficeRoomNum의 정보는 문서에 수치로 값이 정해져 있는 정적인 정보가 아니고, 센서로부터 전달받은 값, 즉 사람과 사무실에 대한 정보를 판단한 후 입력되는 정보이다. 이렇게 입력된 정보는 (그림 2)에서 설명하였던 것과 같이 워크플로우 파서를 통해 문서 인스턴스 AST를 생성하며, 이 정보를 컨텍스트 비교 모듈의 입력 값이 된다. 그리고 그림에서 우측의 입력 값은 실제 센싱된 사람의 정보와 사무실 방 번호에 대한 컨텍스트 정보를 구조적 컨텍스트 모델을 통해 객체화시킨 정보이다. 따라서 컨텍스트 비교 모듈은 이 두 입력 값의 비교를 통해 사용자의 주변상황을 판단한다. 기존의 연구 uFlow [6]에서는 시나리오 문서에 기술할 때 정적으로 미리 정해진 값을 입력 정보로 사용하였다. 그러나 CAWL 기반의 시나리오 문서에서 기술하는 상황정보는 센서로부터 전달되는 값을 전달받아 상황정보를 기술하는 방법을 적용하여, 기존에 정적으로 상황정보를 기술하였던 측면을 개선하였다.

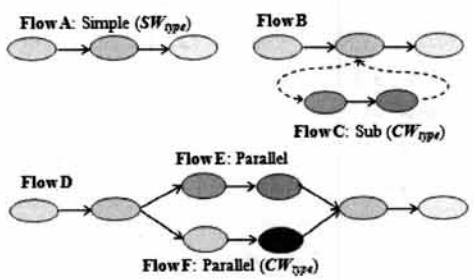


(그림 6) 상황정보와 시나리오 문서 정보의 비교 과정

4. 복합 워크플로우 모델(Composite Workflow Model: CWM)

상황인지 시스템이 다양한 복합 워크플로우 서비스를 제공하기 위해서는 이를 표현하기 위한 시나리오가 있어야 하며, 또한 시나리오를 작성하기 위한 워크플로우 언어가 필요하다. 따라서 본 시스템은 복합 워크플로우를 지원하는 CAWL을 기반으로 하고 있다. CAWL은 다수의 워크플로우에 존재하는 개별적인 서비스 흐름을 하나의 워크플로우로 통합 표현할 수 있는 상황인지 기반의 워크플로우 언어로, 사용자가 원하는 서비스를 제공하기 위해 결합 가능한 다수의 워크플로우를 자연스럽게 연결하여 다양한 형태의 상황인지 워크플로우 서비스를 표현할 수 있다. 또한 복합 워크플로우 모델(CWM)을 포함한다.

(그림 7)은 워크플로우 모델을 보여준다. CWM은 기본 워크플로우 패턴(basic workflow patterns) [12]을 적용한 단순 워크플로우 타입 (SWtype: Simple Workfow Type)과 이들의 조합을 통해 작성된 복합 워크플로우 타입(CWtype: Composite Workflow Type)을 포함한다. CWtype은 서브-워크플로우, 병렬-워크플로우와 같은 모델을 지원한다. 그림에서 Flow B는 또 다른 워크플로우(Flow C)를 자신의 일부로 삼고 있으며, 이 때 Flow C를 서브-워크플로우라 지칭한다. 병렬 워크플로우 타입은 동시에 두 개 이상의 서비스를 동시에 제공하기 위한 것이다. 이는 결정적인 흐름과 그렇지 않은 흐름을 모두 가지고 있는 형태로, 이를 통해 보다 큰 규모의 워크플로우를 표현할 수 있다. 예를 들어, 그림에서 Flow D는 서브 워크플로우 Flow E와 F를 포함하고 있으며, 이들 서브-워크플로우는 Flow D의 두 번째 노드가 종료된 이후에 Flow E와 Flow F로 동시에 진행되는 프로세스를 갖는다.



(그림 7) 워크플로우 모델의 유형

5. 실험 및 평가

본 논문에서 제안하는 CAWL 시스템의 설계 목적은 복합 워크플로우 모델을 지원하는 CAWL을 기반으로 여러 흐름의 워크플로우를 통합하는 복합 워크플로우 서비스를 제공하는 것이다. 이를 위해 본 실험에서는 CAWL을 이용하여 시나리오를 작성하였으며, 이를 이용하여 사용자 일정에 따라 적합한 서비스를 제공하는지 시뮬레이션 실험으로 확인하였다. 실험을 위해 사용자 상황정보에 따라 서비스 분기를 결정하는 John의 하루 일정에 관련된 워크플로우 서비스 시나리오를 사용한다. 시나리오의 내용은 (그림 8)과 같다. 시나리오의 내용은 일반적으로 회사원이면 누구나 경험할 수 있는 상황을 가정하여 작성하였다.

> 회사원 John은 출근 후 자신의 노트북으로 일정에 오전 11시에 301호에서 발표가 있음을 기록한다. 10시 30분이 되면 회의 준비 시스템은 노트북에 일정을 보여주고, 회의실의 조명, 마이크, 프로젝터 등의 상태를 점검한다. 기기들의 상태가 양호하면, John의 발표자료를 301호 PC에 다운로드하여 해당 프로그램을 실행시킨다. 미팅 시간이 되면 PC를 제외한 각각의 기기들의 전원을 켜서 회의할 수 있는 환경을 제공한다.

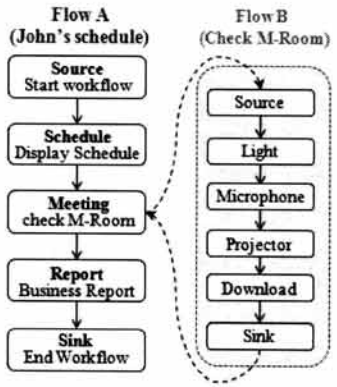
(그림 8) 복합 워크플로우 서비스가 요구되는 서비스 시나리오

```
<CAWL name="OfficeManageSC" version="2.0.1" targetNamespace="test">
  … (중략) …
   <activator name="JohnScheduleActivator">
        <condition expression="c1 and c2">
            <context>
                <rule contribute="positive">
                    <constraint name="c1">
                        <subject type="militaryTime">?Current</subject>
                        <verb>is</verb>
                        <object type="militaryTime">?reservedTime</object>
                    </constraint>
                    <constraint name="c2">
  … (중략) …
   </activator>
  … (중략) …
   <flow name="JohnScheduleFlow">
        <source>
            <input name="inVarJohn" type="officeSchedule_in"/>
        </source>
        <!-- John's SchedDisplay : Start -->
        <node name="SchedDisplay">
            <variable name="schedDisply" type="schedDisplayOn">
                <initialize part="pcId">
                    <from expression="?=inVarJohn/pcId"/>
                </initialize>
  … (중략) …
            <condition expression="c1 and c2 and c3 and c4">
                <context>
                    <rule contribute="positive">
                        <constraint name="c1">
                            <subject type="dateType">?Today</subject>
                            <verb>is </verb>
                            <object type="dateType">?whatDay</object>
                        </constraint>
  … (중략) …
                    </rule>
                </context>
            </condition>
            <invoke serviceProvider="Company" portType="schedDisplayPT"
                operation="schedDisplay" input="?schedDisply"/>
        </node>
  … (중략) …
   <flow name="CheckMeetingRoom">
        <variable name="roomCheck" type="checkRoom">
            <initialize part="roomNum">
                                    <from expression="501"/>
            </initialize>
  ……
```

(그림 9) 개발자가 작성한 시나리오 문서

(그림 9)는 (그림 8)에 기술한 시나리오를 바탕으로 작성한 시나리오 문서이다.

(그림 9)의 시나리오 문서는 CAWL 파서의 입력 정보로 전달되며, 이를 바탕으로 문서 인스턴스 AST(DIAST), 변수 테이블 등이 생성된다. (그림 9)에는 시나리오 문서의 내용이 많아 실험의 주요 부분을 위한 내용만을 나타내었다. 복합 워크플로우 서비스를 제공하기 위한 시나리오 문서에는 다수의 플로우(flow)를 기술할 수 있어야 한다. 이를 위해 본 실험에서는 상황의 복잡한 정도를 최소 수준으로 가정하여 2개의 워크플로우로 표현하였다. 시나리오 문서에 대한 워크플로우의 내용은 (그림 10)에 표현하였다. (그림 9)에서 첫 번째 기술한 플로우(JohnScheduleFlow)는 (그림 10)의



(그림 10) 시나리오에 대한 워크플로우

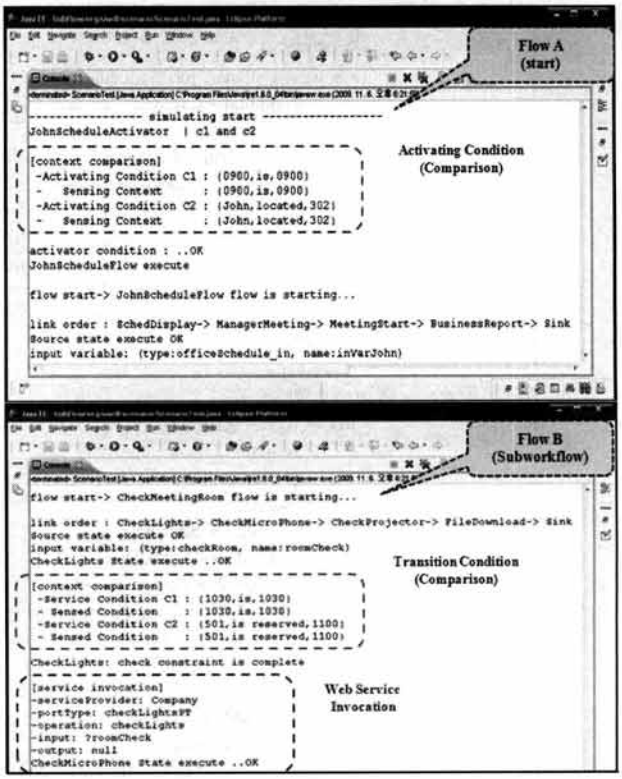
Flow A를, 두 번째 나타나는 플로우(CheckMeetingRoom)는 서브-워크플로우로 결합되는 (그림 10)의 Flow B를 나타낸다. 그리고 <activator>…</activator>에 해당하는 부분은 시나리오 문서에 단 한번만 나타난다. 이것의 역할을 "사용자에게 서비스를 제공해야 할 상황이 맞는가?"를 판단하는 것으로, 상황이 적합하다고 판단되면 워크플로우를 시작하게 하는 활성자이다. Flow A는 John의 하루 일정에 관련된 시나리오 문서의 내용을, Flow B는 서브-워크플로우로 결합되는 Check MeetingRoom 플로우에 해당하는 문서의 내용을 워크플로우로 표현한 것이다.

본 실험에서는 실제 사용자 주변 환경으로부터 발생되는 컨텍스트 정보를 얻는 것과 동일한 환경을 조성하기 위해 난수 발생기를 활용하였다. 난수 발생기로부터 제공되는 정보의 단위는 constraint를 나타내는 {subject, verb, object}의 엔티티 집합이다. 집합을 이루는 각각의 엔티티(entity)는 실제 센서로부터 들어오는 값들이며, 이들은 구조적 컨텍스트 모델을 거쳐 constraint 단위로 구성된다. 이렇게 가공된 컨텍스트 정보를 저장해 두었다가 난수 발생기를 통해 {주어, 동사, 목적어}의 집합으로 구성된 임의의 컨텍스트 정보를 지속적으로 발생시켜, 실제 상황에서 센서를 통해 발생하는 상황과 동일하도록 구현하였다. 이 같은 가공된 컨텍스트 정보의 생성과정은 (그림 6)의 우측에 표현하고 있으며, 내용은 센서 네트워크로부터 제공되는 저 수준의 컨텍스트 정보를 구조적 컨텍스트 모델을 통해 {주어, 동사, 목적어}로 구성된 엔티티의 집합으로 객체화(OC: Objectfied Context)하는 것이다.

(그림 11)은 이클립스 통해 실행된 결과를 화면으로 나타낸 것이며, 복합 워크플로우 서비스를 제공하는 과정을 보여준다. 그림에서 상단의 붉은 색 점선 박스는 Flow A를 활성화시키기 위해 활성 조건(activating condition)을 비교하는 모습을 보여준다. 비교 과정에서 C1, C2는 상황을 기술하기 위한 최소 단위인 Constraint을 나타내는 것으로, 시나리오 문서로부터 얻은 상황정보의 값을 나타낸다. 그리고 Sensing context에 해당하는 정보는 난수 발생기로부터 얻은 값 중 일치하는 값이 나타난 경우 해당 값을 나타내는 것이다. 이와 같이 "시나리오 문서에 기술한 상황정보와 센서로부터 받은 상황정보가 동일하다"라고 판단되면 워크플로우가 활성화되며, 이와 동시에 JohnScheduleFlow가 시작된다. 그 다음은 워크플로우의 실행 순서를 나타내는 링크 정보를 보여준다.

(그림 11)에서 하단은 서브-워크플로우로 호출되는 Flow B의 실행과정을 보여주고 있다. 붉은색 박스는 전이 조건을 위한 상황정보를 비교하는 과정이다. 이는 활성 조건을 나타내는 것이 아니라 다음 서비스로의 전이를 위한 전이 조건(transition condition)을 판단하는 것이다. 그리고 비교 절차가 끝난 후 실행하게 될 서비스(Check Light)를 호출하는 과정을 보여주고 있다.

시나리오 기반의 기존 상황인지 워크플로우 시스템들 [2, 3, 6, 8]은 본 논문에서와 같이 워크플로우를 이용한 서비스



(그림 11) 복합 워크플로우 서비스 실행 과정

를 제공하고 있으나, 여러 워크플로우를 결합한 복합 워크플로우 서비스를 제공하는데 한계가 있었다. 이를 위해 본 실험에서는 복수개의 워크플로우를 통합할 수 있는 CAWL을 기반으로, 여러 워크플로우를 처리할 수 있는 기법들(3장 참고)을 적용하여 복합 워크플로우 서비스를 제공할 수 있음을 보여주었다.

6. 결 론

유비쿼터스 컴퓨팅 환경에서 상황인지 워크플로우 시스템은 다양한 도메인에서 발생할 수 있는 상황을 사용자의 요구조건에 따라 여러 종류의 복합적인 상황을 표현할 수 있는 자동화 서비스를 제공할 수 있어야 한다. 따라서 본 논문에서는 이러한 문제를 해결하기 위하여 상황인지 기반의 복합 워크플로우 모델을 지원하는 CAWL을 기반으로 복합 워크플로우 서비스를 제공하는 워크플로우 시스템을 제안하였다. 제안하는 CAWL 시스템은 상황정보를 기반으로 여러 사용자에게 단일 워크플로우를 조합한 보다 큰 규모의 복합 워크플로우 서비스를 제공할 수 있으며, 동일한 서비스에 대해서는 기존에 작성한 워크플로우를 재사용함으로써 개발의 효율성을 증대시킬 수 있다. 이를 위해 CAWL 시스템은 CAWL 파서, 워크플로우 풀(pool), 컨텍스트 비교기 등과 같은 주요 기능을 제공하는 모듈을 포함하고 있다.

CAWL 시스템은 기존 상황인지 기반 시스템 [6]에 비해 다수의 워크플로우를 통합하여 복합 워크플로우를 구성할

수 있다는 측면에서 기능이 개선되었다. 이를 위해 기존의 워크플로우를 재사용하기 위해 워크플로우 풀을 구성하였다. 그리고 여러 시나리오 문서에서 발생하는 변수 및 메시지들에 대한 접근성을 높이기 위해, CAWL 파서가 AST를 생성하는 것 이외에도 심볼 테이블(symbol table)을 구성할 수 있는 기능을 가지도록 하였다. 마지막으로 컨텍스트 비교기의 경우, uFlow에서는 상황정보를 시나리오에 정적으로 기술하였던 반면에, CAWL 기반의 시나리오에서는 상황정보를 받을 때마다 변수로 처리하도록 기술하였다. 다시 말해서, uFlow에서는 컨텍스트 정보를 {주어, 동사, 목적어} 엔티티 집합 형태로 사전에 기술하여 한꺼번에 비교하였던 것에 비해, CAWL 시스템에서는 각각의 엔티티를 각각 비교하여 {주어, 동사, 목적}의 셋을 확인하는 절차를 거친다.

향후 본 연구는 CAWL 시스템의 효용성을 다각도로 검증하기 위해서 이질적 서비스 도메인에서 발생할 수 있는 다양한 상황에 대하여 풍부한 시나리오를 작성해야 할 것이다.

참 고 문 헌

[1] Chen, S., Bu, Y., Li, J., Tao, X., Lu, J., "Toward context-awareness: a workflow embedded middleware," Proceedings of IFIP 2006 International Conference on Ubiquitous and Intelligent Computing, Vol.4159 of LNCS, pp.766-775. 2006.

[2] Abu Zafar, A. and A. S. Zubair, "A Conceptual Framework for Smart Workflow Management," Proceedings of the 2009 International Conference on Information Management and Engineering, IEEE Computer Society: pp.574-578. 2009.

[3] M Wieland, O. K., D Nicklas, F Leymann, "Towards context-aware workflows," CAiSE07 Proc. of the Workshops and Doctoral Consortium, 2007.

[4] Wieland M. Kaczmarczyk P., Nicklas D., "Context Integration for Smart Workflows," Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), IEEE Computer Society: pp.239-242. 2008.

[5] Y. C. Ngeow, D. C., A. K. Mustapha, E. Goh, H. K. Low, "A Context-Aware Architecture for Smart Space Environment," 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), 26-28 April, 2007, Seoul, Korea, IEEE Computer Science. pp.908-913. 2007.

[6] Joohyun Han, Yongyun Cho, Eunhoe Kim, Jaeyoung Choi, "A Ubiquitous Workflow Service Framework," ICCSA06: pp.30-39, 2006.

[7] Sucha S., L. Sea, et al., "A survey on context-aware workflow adaptations," Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia(MoMM08),, ACM: pp.414-417. 2008.

[8] 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 (AINA06), Vol.1, pp.199-204, 2006.

[9] 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.

[10] 최종선, 조용윤, 최재영, "다중-워크플로우를 지원하는 상황인지 워크플로우 언어의 설계", 한국인터넷정보학회 논문지, 제10권 제6호, 2009.12.

[11] IBM, SAP AG: WS-BPEL Extension for People. http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people

[12] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow Patterns," Distributed and Parallel Databases, 14(3), pp.5-51, 2003.

[13] 이용주, "시맨틱과 워크플로우 혼합기법에 의한 자동화된 웹 서비스 조합시스", 정보처리학회논문지D, 제13권 제4호, pp.265-272, 2007.4

[14] 한주현, 조용윤, 최재영, "웹 서비스 기반의 유비쿼터스 워크플로우 언어", 정보처리학회논문지A, 제12권 제6호, pp.485-492, 2005.12

최 종 선

e-mail : jschoi@ss.ssu.ac.kr

2000년 숭실대학교 컴퓨터학부 (학사)

2002년 숭실대학교 컴퓨터학과 (공학석사)

2003년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 시스템 소프트웨어, 병렬/분산 처리, 유비쿼터스 컴퓨팅

조 용 윤

e-mail : yycho@sunchon.ac.kr

1995년 인천대학교 전산학과(학사)

1998년 숭실대학교 컴퓨터학과(석사)

2006년 숭실대학교 컴퓨터학과(박사)

2009년~현 재 순천대학교 정보통신공학부 조교수

관심분야 : 시스템 소프트웨어, 임베디드 소프트웨어, 유비쿼터스 컴퓨팅

최 재 영

e-mail : choi@ssu.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1986년 미국 남가주대학교 컴퓨터공학(석사)

1991년 미국 코넬대학교 컴퓨터공학(박사)

1992년~1994년 미국 국립 오크리지연구소 연구원

1994년~1995년 미국 테네시 주립대학교 연구교수

2001년~2002년 미국 수퍼컴퓨팅응용센터(NCSA) 초빙연구원

1995년~현 재 숭실대학교 정보과학대학 컴퓨터학부 교수

관심분야 : 병렬/분산처리, 고성능컴퓨팅(HPC), 유비쿼터스컴퓨팅