다중-워크플로우를 지원하는 상황인지 워크플로우 언어의 설계☆

The Design of a Context-Aware Workflow Language for Supporting Multiple Workflows

최 종 선* 조 용 윤** 최 재 영***
Jongsun Choi Yongyun Cho Jaeyoung Choi

요 약

최근 유비쿼터스 컴퓨팅 환경에서의 서비스 자동화를 위해 워크플로우 기술을 적용하려는 연구가 활발히 진행되고 있다. 그러나 대부분의 기존 상황인지 워크플로우 언어들은 단일 워크플로우 처리만을 고려하기 때문에, 다수의 워크플로우 조합을 통한 복합적이고 다양한 상황인지 서비스 지원에 제약이 있다. 본 논문은 다수의 워크플로우에 존재하는 개별적인 서비스 흐름을 하나의 워크플로우로 통합 표현할 수 있는 상황인지 기반의 워크플로우 언어인 CAWL (Context-Aware Workflow Language)을 소개한다. CAWL은 사용자가 원하는 서비스를 제공하기 위해 결합 가능한 다수의 워크플로우를 자연스럽게 연결하여 다양한 형태의 상황인지 워크플로우 서비스를 표현할 수 있다. 또한 개발자는 새로운 워크플로우 개발을 위해 기존에 존재하는 다수의 워크플로우를 다시 사용할 수 있기 때문에, 상황인지 워크플로우의 개발 노력을 줄이고 워크플로우의 재사용성을 높일 수 있다. 따라서 CAWL은 다수의 사용자 워크플로우가 공존하는 유비쿼터스 컴퓨팅 환경에서 상황인지 서비스 자동화와 관련된 응용 개발에 큰 도움이 될 것으로 기대된다.

ABSTRACT

In recent years, there have been several researches applying workflow technologies for service automation on ubiquitous computing environments. However, most context-aware workflow languages have difficulties in supporting composite workflows composed of single workflows, because these languages are still in its early stage and they only provide single workflow services to their users. This paper introduces CAWL, which is a context-aware workflow language. CAWL is for describing individual service workflows to make integrated service workflows. By using CAWL, service developers are able to reuse existing workflows to develop new context-aware workflow services. Therefore, development efforts and time can be saved and workflow resusability also increased. CAWL is expected to make it easy to develop applications related to context-aware workflow services on ubiquitous computing environments.

☞ KeyWords : ubiquitous computing, context-aware, workflow, multiple workflows, CAWL, 유비쿼터스 컴퓨팅, 상황인지, 워크플로우, 다중-워크플로우, CAWL

1. 서 론

비즈니스 프로세스의 표준화를 기반으로 신뢰성 높은 IT 서비스를 제공하기 위해 워크플로우 기술을 컴퓨팅 환경에 잘 적용시켜 상용화에 이르렀다. 그러나 최근 새로운 IT 패러다임인 유비쿼터스 컴퓨팅이 화두로 등장하면서, 워크플로우 기술의 초점은 기계 중심의 분산 컴퓨팅 환경에서 인간 중심의 유비쿼터스 컴퓨팅 환경으로 옮

* 정 회 원 : 숭실대학교 컴퓨터학과 박사과정
jschoi@ss.ssu.ac.kr
** 정 회 원 : 국립순천대학교 정보통신공학부 조교수
sslabyycho@hotmail.com
*** 정 회 원 : 숭실대학교 정보과학대학 컴퓨터학부 교수
choi@ssu.ac.kr(교신저자)
[2009/03/13 투고 - 2009/03/21 심사 - 2009/05/28 심사완료]
☆본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2009-(C1090-0902-0007))

겨지고 있다. 이와 같은 환경에서 사용자의 요구사항을 만족시키기 위해서는 사용자와 주변 환경에 대한 정보(상황정보)를 바탕으로 사용자에게 적합한 서비스를 자동으로 제공할 수 있어야 한다. 그러므로 상황정보를 기반으로 사용자에게 자동화된 서비스를 제공하기 위해서는 대표적인 비즈니스 프로세스 자동화 모델인 워크플로우를 유비쿼터스 컴퓨팅 환경에 적용해야 할 필요성이 있다. 이에 따라 최근에는 유비쿼터스 컴퓨팅 환경에서의 상황인지 개념을 워크플로우 모델에 적용시키려는 연구들이 진행되고 있다 [1, 2, 3].

유비쿼터스 컴퓨팅 환경에서의 워크플로우 시스템은 사용자의 시나리오를 바탕으로 사용자의 주변 환경에서 발생하는 상황정보에 따라서 사용자에게 적합한 상황인지 서비스를 제공해야 한다 [4]. 이에 따라 워크플로우 언어는 사용자의 상황정보를 표현할 수 있을 뿐만 아니라, 복합적이고 다양한 상황인지 서비스까지 표현할 수 있어야 한다. 또한 워크플로우 시스템이 다수의 사용자에게 적합한 다중 워크플로우 서비스를 제공하기 위한 언어적 기반을 제공해야 한다. 이러한 다중-워크플로우의 중요성은 “다수의 사용자에게 각각의 서비스를 동시에 제공하는 것”이라고 시스템 관점에서 정의한 국제 학술 연구에서 강조된 바 있다 [5]. 또한 다수의 워크플로우의 조합된 형태를 표현하는 워크플로우 패턴(언어적 측면) 및 구현(시스템 측면)에 대한 지침까지 기술하고 있는 P4PAIS* 프로젝트의 워크플로우 패턴에 관한 연구에서도 나타나 있다 [6].

그러나 기존 상황인지 기반 워크플로우 언어들은 아직 단일 워크플로우 처리만을 고려하므로, 다수의 워크플로우 조합을 통해 복합적인 상황인지 서비스를 언어로 표현하는데 제약이 있다. 본 논문에서는 다수의 워크플로우에 존재하는 개별적인 서비스 흐름을 하나의 워크플로우로 통합 표현할 수 있는 상황인지 기반 워크플로우 언어(CAWL)를 제안한다. CAWL은 사용자에게 적합한 서비스를 제공하기 위해 결합 가능한 다수의 워크플로우를 자연스럽게 연결하여 다양한 형태의 상황인지 워크플로우 서비스를 표현할 수 있다. 또한, 개발자는 새로운 워크플로우 개발을 위해 기존에 존재하는 다수의 워크플로우를 사용할 수 있기 때문에, 개발 노력을 줄이고 워크플로우의 재사용성을 높일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 워크플로우 언어들의 고찰을 통해 상황인지 워크플로우 언어의 요구사항에 대하여 알아본다. 3장에서는 CAWL에 대한 설계 및 구조에 대한 설명을 요구사항을 중심으로 기술한다. 4장에서는 CAWL의 효용성을 입증하기 위해 시나리오를 작성 후 비교 및 검토하고, 기존 워크플로우 언어와 CAWL을 비교 분석한다. 마지막으로 5장에서는 향후 CAWL에 추가적으로 필요한 연구에 대하여 언급하도록 한다.

2. 관련 연구

2.1 워크플로우 기술

현재 WfMC는 워크플로우를 “전체 혹은 부분적인 비즈니스 프로세스의 자동화 또는 컴퓨터로 처리되는 간이화”로 정의하고 있다 [7]. 워크플로우에서는 하나의 큰 작업이 완료될 때까지 일어나는 하위 작업들의 흐름을 XML 기반의 언어를 이용하여 표준화된 방법으로 표현한다. 워크플로우의 하위 작업들 간에는 의존성, 수행 순서, 병렬 수행 가능 여부 등과 같은 다양한 관계가 나타난다. 이와 같이 대규모의 작업을 연관된 흐름을 가진 소규모의 작업들로 나누어 표현할 경우, 큰 단위의 작업에 소요되는 자원을 각각의 작은 작업에 효율적으로 분배하므로 전체 작업 수행의 효율성을 향상시킬 수 있다 [8].

워크플로우 기술은 XLANG, WSFL로부터

* Patterns for Process-Aware Information Systems. NWO (Netherlands Organisation for Scientific Research)의 지원을 받아 진행 중인 프로젝트로써 미국의 BPMN, IBM, Oracle BPEL을 포함하여 유럽의 COSA, FLOWer 이외에도 다수의 세계적인 기업들이 벤더로 참여하고 있다.

WS-BPEL에 이르기까지 Microsoft, IBM, Bea, Oracle 등과 같은 세계적인 컴퓨터 기업의 지원에 힘입어 지속적인 발전이 이루어지고 있다. 이들 워크플로우 언어의 특징을 살펴보면 다음과 같다. BPEL4WS [9]은 초기 워크플로우 언어인 방향성 그래프 기반의 XLANG [10]과 블록 구조 형태인 WSFL [8]을 수렴(convergence)하여 IBM에서 설계한 초기 BPEL을 발전시킨 웹 서비스 기반의 워크플로우 언어이다. 이들 워크플로우 언어는 서비스 분기와 선택을 위한 전이조건으로 이전 서비스 결과로부터 XPath, XLink, XPoint와 같은 XML 링크 기능 [11]의 조건-관계 연산식을 통해 얻어진 값을 이용할 수 있다. 그러나 유비쿼터스 환경에서 발생하는 상황정보는 이와 같은 연산식의 단순 결과 값이 아니라 사용자의 위치 정보 및 시간 정보 등과 같은 복합적인 주변 정보들의 집합으로 구성된다. 따라서 유비쿼터스 컴퓨팅 환경에서 기존 워크플로우 언어의 XML 링크와 같은 기능으로는 워크플로우 서비스 간 분기 또는 다음 제공할 서비스를 선택하기 위한 전이 조건을 충분히 표현할 수 없다.

2.2 상황인지 워크플로우

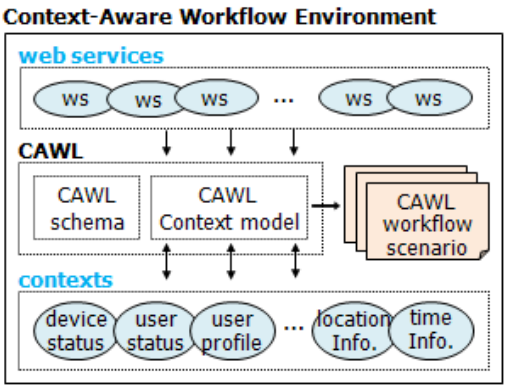
본 절에서는 상황인지 기반의 유비쿼터스 컴퓨팅 환경에서 워크플로우 모델에 상황인지 개념을 적용한 사례 및 이들에 대한 요구사항에 대하여 살펴보기로 한다. FollowMe는 편재형 컴퓨팅 환경에서 존재하는 다양한 영역에 적용할 수 있는 통일된 형태의 상황인지 워크플로우 기반구조를 제공하는 미들웨어이다 [12]. FollowMe는 다른 영역의 상황정보들과 응용 프로그램들을 접속시켜 다양한 영역에 최적화될 수 있고, 사용자의 다양한 서비스 요구를 만족시키기 위하여 시나리오 기반의 워크플로우 모델을 사용하고 있다. 또한 상황인지 환경에서 FollowMe의 효율 및 적응성을 높이기 위하여 XPDL (XML Process Definition Language)을 수정하여 작성한 CPDL (Compact Process Definition Language)을 소개하고 있다. 그러나 FollowMe는 워크플로우 기반의 어플리케이션 모델에 초점을 맞추고 있어, 미들웨어 플랫폼에 적용되고 있는 워크플로우 언어(CPDL)에 대한 구체적 설계 및 정의에 대한 보완이 필요하다. uWDL은 상황정보를 서비스의 전이 조건으로 표현함으로써 기존 웹 서비스 기반의 워크플로우 언어와는 달리, 서비스 영역을 유비쿼터스 컴퓨팅 환경으로 이전시킨 상황인지 기반의 워크플로우 언어이다 [13]. uWDL은 유비쿼터스 컴퓨팅 환경을 위해 기존 웹 서비스 기반의 워크플로우 언어들이 포함하고 있는 기능을 대폭 축소하여 적용하였다. 개발자는 uWDL을 이용하여 워크플로우 서비스의 분기 조건으로써 사용자의 상황정보를 기술하는 시나리오를 작성하여, 이를 기반으로 유비쿼터스 컴퓨팅 환경에서 사용자에게 상황인지 서비스를 제공할 수 있다.

그러나 uWDL은 개별적으로 존재하는 다수의 워크플로우를 조합할 수 있는 기능이 결여되어 있어, 다수의 워크플로우를 조합한 형태의 워크플로우 패턴을 지원하는데 한계가 있다. 유비쿼터스 컴퓨팅 환경은 여러 사용자와 다수의 서비스가 공존하는 공간이다. 그러므로 서비스 개발자는 여러 사용자에게 다양한 형태의 서비스를 제공하기 위해 개별적으로 존재하는 다수의 워크플로우 서비스를 조합한 형태의 다중 워크플로우 서비스를 제공할 수 있어야 하며, 또한 이러한 서비스를 상황인지 워크플로우 언어를 통해 표현할 수 있어야 한다. 따라서 서비스 개발자가 다중 워크플로우 서비스를 상황인지 워크플로우 언어로 표현하기 위해서는 다음과 같은 기능들이 요구된다.

- 복합적인 서비스의 표현을 통한 워크플로우의 다양성 및 재사용성
- 유비쿼터스 컴퓨팅 환경에서 실시간으로 발생하는 동적인 상황정보의 표현
- 서비스 노드 또는 워크플로우 간 데이터 흐름과 제어 흐름에 대한 처리
- 여러 사용자에게 다중 워크플로우 서비스 제공을 위한 언어적 지원

3. CAWL의 설계 및 구성

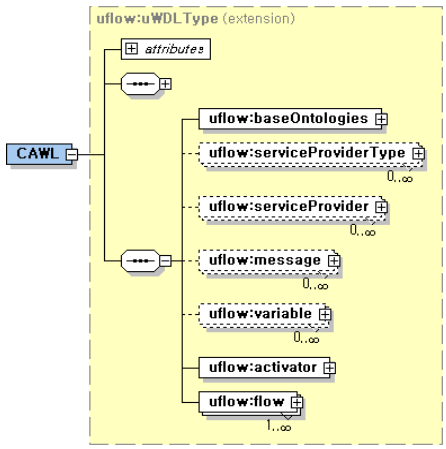
CAWL는 그림 1에서와 같이 웹 서비스를 기반으로 사용자 및 디바이스에 대한 프로파일, 위치, 시간 등과 같은 상황정보를 가지고 사용자에게 적합한 서비스를 선택할 수 있는 기능을 제공할 수 있으며, 다수의 사용자를 위해 워크플로우 서비스의 흐름을 제어하고 동적인 상황정보를 표현할 수 있는 상황인지 기반의 워크플로우 언어이다. 서비스 개발자는 여러 사용자를 위한 다중-워크플로우 서비스를 지원하기 위해 다수의 CAWL을 작성할 수 있으며, 상황정보는 컨텍스트 모델을 기반으로 기술한다.



(그림 1) 다중 워크플로우 지원을 위한 CAWL의 개념도

유비쿼터스 컴퓨팅 환경에서는 다양한 서비스 영역이 존재할 수 있으며, 이들 영역 내에는 여러 사용자와 서비스가 존재한다. 따라서 서비스 개발자는 다수의 CAWL 문서를 기술함으로써 여러 사용자에게 적합한 다중의 워크플로우 서비스를 제공할 수 있다. 각각의 CAWL 문서에는 하나의 사용자에 대한 워크플로우 서비스를 기술한다. 해당 문서에는 하나 또는 그 이상의 워크플로우를 기술함으로써 한 명의 사용자에게 다수의 단일 워크플로우의 조합으로 이루어진 복합 워크플로우 서비스를 표현할 수 있다.

CAWL 문서의 최상위 원소는 그림 2에서 볼 수 있듯이 항상 <CAWL>이다. <CAWL>은 워크플로우의 시작점을 알려주는 활성자(activator-3.1절 참조)와 단일 워크플로우 단위의 서비스를 기술하는 플로우(flow) 이외에도 온톨로지(baseOntologies), 서비스 제공자(serviceProvider), 메시지(message)와 변수(variable) 등에 대한 정의를 포함한다. 웹 서비스를 호출하거나 내부에서 필요로 하는 데이터를 유지(hold)하기 하며, 서비스 간 메시지 전달 및 동적인 상황정보를 표현하기 위한 원소인 <message>, <variable>는 3.2절에서 상세히 설명하도록 한다. CAWL 언어에서 논리적 구조를 포함하는 원소는 <documentation>와 같은 하위 원소를 가질 수 있다. 이는 HTML과 같은 마크-업 언어에서 사용하는 복잡한 방법("<!--")을 사용하지 않고, 해당 원소에 대한 설명을 CDATA를 이용하여 기술할 수 있도록 해준다. 또한 향후 XML 스키마에서 사용하는 annotation과 유사한 기능으로의 확장을 고려할 수 있다.

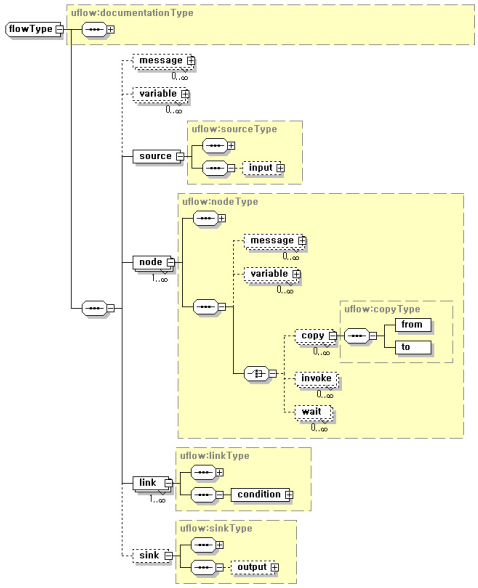


(그림 2) 제안하는 CAWL의 전체 스키마

서비스 개발자는 온톨로지 기반의 상황정보를 사용자에게 제공할 다음 서비스에 대한 전이조건으로 사용해야 한다. 그러므로 <baseOntologies> 원소를 통해 참조할 온톨로지 문서들을 명시할 수 있다. 이 원소는 하나 이상의 <ontology> 하위

원소를 가질 수 있으며, location 속성을 통하여 온톨로지의 기술 방법에 대한 표준인 OWL 문서에 대한 URL을 기술할 수 있다. 또한 namespace 속성을 통하여 워크플로우 문서에서 해당 온톨로지에 대하여 일종의 별명(alias)으로 사용할 네임스페이스를 기술할 수 있다.

서비스 제공자 <serviceProvider> 원소는 하나의 포트 타입(PortType) 또는 여러 포트 타입들의 묶음으로 표현될 수 있는 외부 인터페이스를 정의하며, WSDL과 호환되는 하나 또는 그 이상의 포트 타입들을 포함할 수 있다. 각 포트 타입은 직접 <operation> 하위 원소를 이용해 정의될 수도 있으며, 외부의 WSDL 문서로부터 가져와 포함될(import) 수도 있다. 실제 서비스 제공자는 로케이터(locator)에 기반하여 결정되는데, 로케이터는 정적인 주소로 표현하는 방법과 UDDI를 통해 동적으로 찾는 방법 중 한가지로 정의된다.



〈그림 3〉 개별적으로 존재하는 워크플로우를 표현하기 위한 〈flow〉 원소의 스키마

CAWL에서 각각의 워크플로우는 <flow> 원소로 표현되며 'name'이라는 속성에 의해 명명된다. 그림 3과 같이 <flow>는 <message>와 <variable>외에 <source>, <node>, <link>, <sink>라는 하위 원소들을 가질 수 있다. <source>는 각각의 워크플로우 시작점으로써 워크플로우의 입력을 선언한다. <node>는 하나 이상 나타날 수 있으며, 각각은 서비스 제공자 중 하나가 제공하는 웹 서비스의 호출 프로세스로 이루어진다. 호출 프로세스에는 실제 웹 서비스를 호출하기 위한 <invoke>, 응답을 기다리기 위한 <wait>, 그리고 변수와 메시지 사이에서 값을 복사하기 위한 <copy>등과 같은 하위 원소들이 포함될 수 있다. <link>는 <source>와 각각의 <node> 그리고 <sink>와 같은 각각의 서비스 노드에 대한 연결을 정의하며, <condition> 하위 원소를 이용하여 서비스의 전이를 위한 전이 조건(transition condition)을 기술할 수 있다. 워크플로우의 종료점을 나타내는 <sink> 원소는 워크플로우의 출력을 선언한다. 워크플로우의 출력은 현재의 워크플로우를 호출한 내·외부의 다른 워크플로우 또는 활성자에 의해 처리될 수 있다.

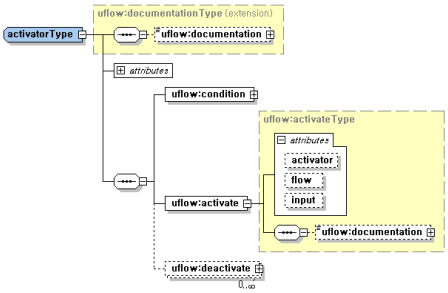
CAWL은 앞서 설명한 <flow> 원소를 이용하여 표현된 다수의 워크플로우를 조합하여, 이를 기반으로 복합 서비스의 표현 및 여러 사용자에게 적합한 다중 워크플로우 서비스를 지원하기 위해 다음과 같은 기능에 초점을 두고 설계되었다.

- 여러 사용자에게 다중 워크플로우 서비스를 제공하기 위한 언어적 표현 기능
- 상황인지 워크플로우 서비스의 흐름을 제어하기 위한 메시징 기능
- 사용자 주변 환경으로부터 발생하는 동적인 상황정보의 표현 기능
- 워크플로우의 다양성 및 재사용성을 증대시키기 위한 복합 워크플로우 서비스의 표현 기능

3.1 다중-워크플로우 서비스의 지원

워크플로우는 여러 서비스가 조합된 것으로써 스스로 동작할 수 있다. 그러나 유비쿼터스 환경

에서 필요한 수많은 워크플로우가 스스로 활성 조건을 가지고 항상 대기 상태에 있다면 자원의 낭비를 초래할 수 있다. 그림 4와 같이 활성자(activator)는 이러한 자원의 낭비를 최소화하기 위하여 워크플로우와는 별개로 존재하며, 다른 워크플로우를 활성화 또는 비활성화시킬 수 있는 경량 트리거(lightweight trigger)이다.



〈그림 4〉 〈activator〉 스키마

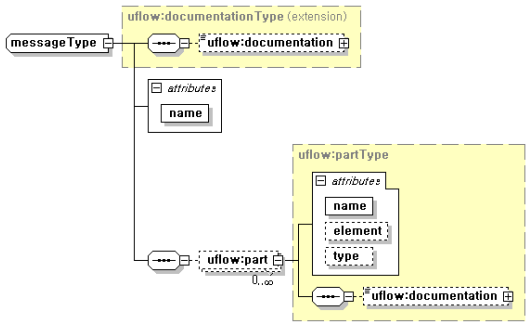
각각의 사용자에 대하여 하나의 CAWL 문서를 기술하므로 각 CAWL 문서에는 <activator> 원소가 한 개만 존재한다. 따라서 하나의 <activator> 원소는 하나 이상의 워크플로우 서비스를 활성화하기 위한 활성자를 정의하는데 사용되며, 다수의 워크플로우들 중에서 처음으로 활성화되는 워크플로우 정보를 가진다. 그리고 여러 사용자 대한 다중 워크플로우 서비스를 지원하기 위해서는 둘 이상의 <CAWL> 문서를 작성하도록 한다

활성자를 나타내는 <activator> 원소는 고유의 이름을 가질 수 있으며, 하위 원소 <condition>을 통해 트리거 조건을 명시할 수 있다. 해당 조건이 만족되는 경우 <activate> 또는 <deactivate> 원소를 통하여 지정한 워크플로우를 제어할 수 있다. 센서를 통해 들어온 상황정보와 워크플로우 시나리오에서 <condition> 원소에 기술되어 있는 정보가 일치하면 <activate>에 기술된 플로우(flow)가 실행된다. <deactivate> 원소는 자신을 포함한 어떤 활성자 또는 플로우를 비활성화하며, <activator> 원소에 속성이 지정되지 않는 경우에는 자신을 비활성화한다.

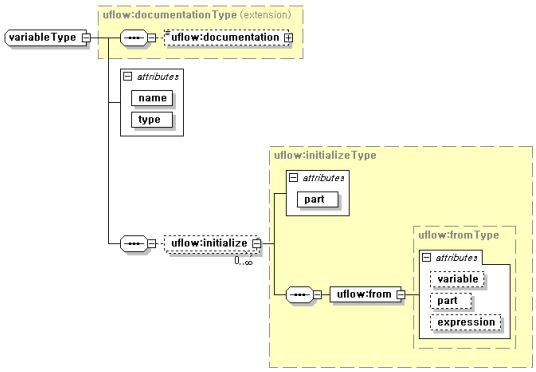
3.2 메시징을 통한 흐름 제어

워크플로우에서의 흐름은 제어 흐름(control flow)과 데이터 흐름(data flow)으로 나뉜다. CAWL는 이 두 가지의 흐름에 대한 제어 기능을 모두 지원한다. BPEL4WS와 같은 워크플로우 언어는 모두를 지원하고 있지만, 상황인지 기반의 워크플로우 언어들은 이러한 기능을 지원하지 않는다. 또한 CAWL은 또한 동적인 상황정보를 기술하기 위한 수단을 제공한다. 이에 대해서는 3.3절에서 <context>와 함께 다루기로 한다.

CAWL은 주변 환경으로부터 발생하는 메시지의 전달을 위해 <message>와 <variable> 원소를 포함한다. 그리고 CAWL은 기본적으로 웹 서비스들을 조합하기 위한 언어이다. 따라서 웹 서비스를 호출하고 결과를 확인하기 위하여 WSDL과 호환되는 메시지를 정의하는 기능을 제공한다. 그림 5의 (a)에서 보는 바와 같이 각각의 메시지는 <message> 원소에 의해 정의되며 'name' 속성으로 이름을 결정한다. 하나의 <message> 원소에는 하나 이상의 <part> 하위 원소가 올 수 있으며, 각 <part>는 'name' 속성에 의해 이름이 주어지고 'element' 또는 'type' 속성에 의해 타입이 결정된다. 다시 말해서 참조하고자 하는 타입 정보가 XML 스키마의 <element> 원소인지 <type> 원소인지에 따라서 결정된다.



(a) 〈message〉 스키마



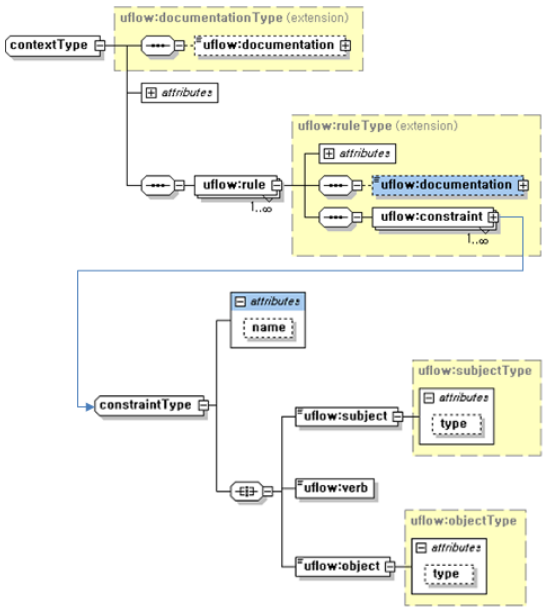
(b) 〈variable〉 스키마

(그림 5) 메시지 전달 및 동적 상황정보 표현

CAWL은 웹 서비스를 호출하거나 서비스를 제공하기 위해 필요한 데이터를 유지하기 위해 변수를 선언할 수 있다. 그림 5의 (b)와 같이 변수는 <variable> 원소를 이용하여 선언할 수 있으며 'name' 속성으로 이름을 정의한다. 그리고 'type', 'element', 'message' 중의 한 가지 속성을 이용하여 타입을 정의할 수 있다. variableType은 변수를 의미하며 XML Schema에 의해 제공되는 타입과 더불어 정의된 message를 type으로 가질 수 있다. <variable>의 하위 원소인 <initialize>는 변수의 초기화를 위해 사용되며, message의 type이 variable일 경우 각각의 part를 초기화할 수 있다. <initialize> 원소는 초기화하고자 하는 특정 부분(part)의 이름을 결정하고, 그 하위 원소인 <from>을 이용하여 특정 값이나 변수들의 수식으로부터 값을 계산하여 초기화한다.

3.3 동적인 상황정보의 표현

기존의 상황인지 워크플로우어 언어 [13]에서는 상황정보를 정적으로 기술하였다. 그러나 CAWL는 센서로부터 실시간으로 전달되는 동적 상황정보를 <context>와 더불어 <message>, <variable> 원소를 이용하여 표현할 수 있다. 동적인 상황정보를 표현하는 구체적인 예는 4.1절에서 그림 10의 (b)를 통해 설명하기로 한다.



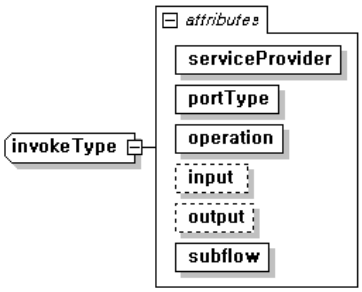
(그림 6) 상황정보 표현을 위한 〈context〉 스키마

유비쿼터스 환경에서 발생하는 사용자 상황정보와 시나리오에서의 컨텍스트는 RDF 기반의 {주어, 동사, 목적어} 형태의 엔티티(entity)들에 대한 집합으로 표현된다. 그림 6은 이러한 상황정보의 표현을 위한 <context> 원소의 스키마이다. <context> 원소에는 하나 이상의 <rule> 하위 원소가 포함되며, 이 원소의 'priority' 속성을 통하여 각 규칙(rule)의 우선순위를 지정할 수 있다. 하나의 규칙은 하나 이상의 단위 컨텍스트, 즉 <constraint> 원소들의 논리 연산식이 결합된 복합 컨텍스트로 표현된다. 각 <constraint> 원소는 <subject>, <verb>, <object>라는 세 개의 하위 원소를 가지며, <subject>와 <object>는 해당 엔티티의 타입(type)과 값(value)을, <verb>는 컨텍스트 모델상에서 <subject>의 속성 이름을 나타낸다.

3.4 통합된 복합 워크플로우 서비스의 지원

CAWL는 <invoke> 원소를 이용하여 단일 워크플로우 단위의 서비스를 호출하여 다양하고 복합적인 워크플로우 서비스를 표현할 수 있다. 그리

고 CAWL에서는 호출되는 단일 워크플로우 서비스를 현재 사용자에게 제공되고 있는 워크플로우 서비스와 상대적인 개념인 서브플로우(subflow)로 명명하였다. 이와 같은 기능을 통해 워크플로우의 서비스 노드에서 <invoke> 원소를 이용하여 현재 CAWL 문서상의 시나리오에 있는 다른 종류의 워크플로우를 서브플로우로 호출할 수 있다. 서브플로우는 하나의 커다란 시나리오의 작은 부분에 해당하는 소규모 시나리오가 될 수 있으며, 이러한 다수의 소규모 시나리오 문서들을 하나로 묶어 대규모의 시나리오 흐름을 구성하여 복합 서비스를 제공할 수 있다. 구체적 예는 4장의 그림 8과 4.1절의 그림 11에서 기술하도록 한다.



(그림 7) 〈invoke〉 스키마

그림 7의 <invoke> 원소는 그림 3에서 볼수 있듯이 <flow>의 하위 원소인 <node>의 하위 원소이다. <invoke>는 웹 서비스 또는 외부의 워크플로우의 서비스를 호출하는 기능을 수행하며, 속성으로 호출하고자 하는 서비스를 포함하는 서비스 제공자(serviceProvider)와 포트타입(portType), 호출하고자 하는 서비스의 연산 이름(operation), 전달하고자 하는 입력 변수의 이름(input), 전달 받고자 하는 출력 변수의 이름(oupput), 다른 워크플로우를 호출하는 속성(subflow)을 포함할 수 있다.

4. 실험 및 평가

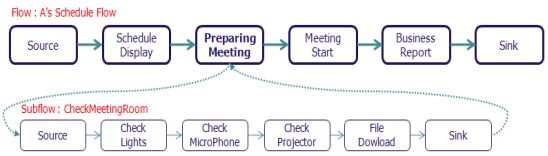
본 장에서는 제안하는 언어의 효용성을 보이기 위해서 CAWL 기반의 서비스 시나리오 예제를 보여준다. 다중의 워크플로우 즉 개별적으로 존재하는 다수의 워크플로우가 존재하고, 이를 서브플로우(sub-flow)라는 개념을 통해 서비스를 호출하면 복합서비스를 제공할 수 있다. 이 같은 서비스는 다수의 문서를 작성하면 다수의 사용자에게 제공될 수 있다. 그러나 이러한 서비스의 제공적인 측면은 시스템 측면에서 고려해야 하는 부분이므로 다중 워크플로우 서비스의 제공을 지원하는 기능적 측면에 초점을 둔다. 또한 기존 워크플로우 언어들과의 기능적 비교 분석을 통해 CAWL이 가진 특징을 알아본다. 예제는 3장에서 언급한 언어 설계 기능을 중심으로 기술한다.

4.1 서비스 시나리오

시나리오는 사용자의 일정에 따라 CAWL을 이용해 표현할 수 있다. 서비스 시나리오를 기술하는 CAWL 문서는 각각의 사용자마다 작성할 수 있다. 또한 여러 사용자에 대해서는 다수의 CAWL 문서를 작성하여 언어적으로 다중의 워크플로루 서비스를 지원할 수 있으므로, 본 절에서는 한명의 사용자 A에 대한 시나리오에 한하여 살펴보도록 한다. 내용은 다음과 같다.

1. 사용자 A는 오전 9시에 출근하여 자신의 사무실 302호에 들어간다.
2. 자신의 일정이 입력된 PDA를 PC와 연결하면 모니터에서 오늘의 일정을 확인한다.
3. 오전 11시의 관리자 미팅 준비를 위해 오전 10시 30분에 501호 회의실에 간다.
4. 오전 11시에 회의 참가자들의 입장이 완료되면 미팅을 시작한다.
5. 오후 4시에는 컴퓨터를 통해 비즈니스 보고서를 제출한다.

위의 시나리오는 회사의 사무실에서 발생할 수 있는 상황에 맞추어 작성하였으며, 이를 워크플로우로 표현하면 그림 8과 같다.



(그림 8) 사용자 A의 시나리오를 표현한 워크플로우

워크플로우 서비스 중 PreparingMeeting 서비스는 사전에 따로 정의되어 있는 단일 워크플로우이다. 따라서 이 워크플로우(subflow)를 호출함으로써 기존의 개별 워크플로우와 새로 정의된 서비스 노드들과 결합하여 새로운 복합 워크플로우 서비스를 제공할 수 있다. 구체적인 서브플로우 호출 예제는 그림 11에 표현되어 있다.

시나리오에서 1번은 사용자 A의 일정 시작을 나타낸다. CAWL는 워크플로우의 시작을 시스템에게 알려주는 <activator>를 이용해 하나의 워크플로우를 활성화할 수 있다. 이를 CAWL 언어를 이용해 기술하면 그림 9과 같다.

```
<activator name="UserActivator">
<!-- … 중략 -->
<!-- activator가 활성화시킬 플로우를 명시 -->
<activate flow="UserFlow"/>
<!-- … 중략 -->
</activator>
<!-- 활성화되는 플로우 -->
<flow name="UserFlow">
<!-- … 중략 -->
</flow>
```

(그림 9) 워크플로우 시작을 나타내는 활성자의 사용 예

<activator>는 하나의 워크플로우를 활성화하는 시작점이자 조건이 된다. uWDL에서는 활성화 기능을 지원하지 않아 하나의 uWDL 문서에는 단일 워크플로우만을 기술할 수 있었으며, 또한 내부에서 선언되어 사용되었던 또 다른 자원(워크플로우 단위의 서비스)들을 재사용할 수 없었다. 이를 위해 CAWL은 <activator> 원소를 이용해 활성화 기능을 제공함으로써 이미 선언된 자원들에 대한 재사용성을 증가시켰다.

그림 10는 동일한 상황에서 uWDL과 CAWL의 상황정보에 대한 기술방법의 차이를 보여준다. 그림 10의 (a)는 기존 uWDL의 정적인 상황정보의 기술 방식을 나타내고 있으며, (b)는 동적인 상황정보의 표현을 위한 예를 나타낸다. CAWL에서의 메시지와 변수는 웹 서비스를 호출할 때 전달되는 입출력 값에 대하여 WSDL과의 호환성 문제때문에 C 언어에서의 구조체 변수와 동일한 개념을 적용하였다.

```
<!-- 문자열과 자료형으로 정적인 상황정보 기술
-->
<context>
    <subject type="person">UserA</subject>
    <verb>located</verb>
    <subject type="officeRoom">301</subject>
</context>
```

(a) uWDL - 상황정보의 정적 기술

```
<!--실행될 플로우에 사용할 메시지와 변수 선언
부분 -->
<message name="OfficeSchdule_In">
    <part name="person" type="Person"/>
    <part name="officeRoom" type="OfficeRoom"/>
</message>
<variable name="inVar" type="OfficeSchdule_In">
    <initialize part="person">
        <from expression="UserA"/>
    </initialize>
    <initialize part="officeRoom">
        <from expression="301"/>
    </initialize>
</variable>
<!-- … 중략 -->
<!-- 실행 플로우 부분 중 상황정보의 표현 부분
-->
<context>
<!-- … 중략 -->
    <subject type="person">?=inVar/person</subject>
    <verb>located</subject>
    <subject type="OfficeRoom">?=inVar/officeRoom</subject>
<!-- … 중략 -->
</context>
```

(b) CAWL - 상황정보의 동적 기술

(그림 10) uWDL과 CAWL의 상황정보에 대한 기술 방법의 차이

그림 10의 (b)에서 메시지(message)는 센서로부터 발생하는 상황정보를 표현하기 위한 것이고, 변수(variable)는 메시지를 전달받아 값을 지정하기 위한 것이다. 이러한 메시지와 변수는 <constraint> 원소를 통하여 동적인 상황정보를 표현하기 위한 수단으로 사용된다. 예를 들면 "UserA가 301호에 위치한다"는 상황정보를 표현하기 위해서는 센서로부터 전달받은 사람과 사무실의 방 번호에 대한 정보를 <variable>의 <part>를 통해 person과 officeRoom를 초기화할 수 있다.

서비스 개발자는 이렇게 초기화된 상황정보를 inVar 변수를 통해 값을 전달할 수 있다. 이렇게 표현된 상황정보를 바탕으로, 워크플로우 시스템은 실제 사용자에 대한 상황정보와 시나리오 문서에 기술된 상황정보를 비교하여 워크플로우 서비스의 실행조건을 판단할 수 있게 된다.

4.1절의 시나리오에서 2번과 5번은 센서에 의해 수집되어 전송된 상황정보가 시나리오에 기술되어 있는 상황정보의 조건과 부합되면 해당 서비스가 작동되어야 한다. 예를 들어 "PDA가 컴퓨터와 연결되고, 사용자 A가 사무실에 존재한다"라는 상황정보가 입력되면 시나리오 문서에 기술된 상황정보의 조건과 비교해야 한다. uWDL에서는 시나리오 문서에 조건을 기술할 때, 단순히 문자열과 자료형을 이용하여 기술하였다. 그러나 이러한 방법은 동일 데이터를 반복해서 표현해야 하는 경우 자원의 낭비를 초래하게 된다. 이를 위해 CAWL은 동적인 상황정보를 처리하기 위한 방법으로 C 프로그래밍 언어에서 사용하는 변수의 개념을 적용하였다.

```
<flow name="UserA_Flow">
  <node name="prepareMeeting">
    <!-- … 중략 -->
  <!-- 속성으로 호출할 플로우 이름을 기술
-->
    <invoke subflow="MeetingRoomService"/>
    <!-- … 중략 -->
  </node>
</flow>
    <!-- … 중략 -->
<!-- 호출되는 플로우 -->
<flow name="MeetingRoomService">
    <!-- … 중략 -->
</flow>
```

(그림 11) 복합 워크플로우 서비스 지원을 위한 서브플로우 호출 예

그리고 시나리오의 3, 4번은 하나의 서비스 노드에서 여러 서비스들이 실행되어야 한다. 관리자 미팅을 위해 미팅 룸의 마이크, 컴퓨터, 프로젝터, 스피커와 같은 장비들의 상태를 검사해야 하고, 발표 자료를 전송받아 발표 준비를 마쳐야 한다. 이러한 준비가 다 끝나게 되면 회의 참가자들에게 회의 참석을 요청하는 메시지를 발송해야 한다. 이러한 상황의 워크플로우는 충분히 반복적으로 일어날 수 있는 상황이다. uWDL에서는 이와 같이 중복되는 상황을 기술하기 위해 기존에 동일한 내용의 워크플로우 문서를 반복해서 작성해야만 했다. 그래서 CAWL은 이와 같이 반복적인 서비스를 호출하기 위해 <invoke>태그의 속성으로 subflow를 추가하여, 이미 작성된 워크플로우 서비스에 대한 재사용성을 증대시켰다. 그리고 이를 통해 다수의 워크플로우를 조합한 형태인 복합 워크플로우 서비스를 표현할 수 있도록 하였다. 그림 11은 이러한 서브플로우의 호출 과정을 나타낸다.

4.2 기존 워크플로우 언어들과의 비교 분석

표 1은 비즈니스 프로세스를 위한 워크플로우 언어 및 이전 상황인지 기반의 워크플로우 언어(uWDL)와의 비교 분석 결과를 정성적으로 나타낸 것이다. BPEL4WS는 WSFL과 XLANG을 기능적, 구조적으로 취합한 형태이므로 표 1에서 분류한 대부분의 항목을 지원한다. 대부분의 워크플로우 언어들과 같이 CAWL 역시 웹 서비스의 프로

(표 1) 워크플로우 언어들의 특징 및 기능적 비교 분석

분류 항목(기능) \ 워크플로우 언어		WSFL	XLANG	BPEL4WS	uWDL	CAWL
언어구조	방향성 그래프(direct graph)	○	×	○	○	○
	블록-구조화(block-structured)	×	○	○	×	○
워크플로우 제어	통제흐름(control flow)	○	○	○	×	○
	데이터흐름(data flow)	○	×	○	○	○
워크플로우 형태	단일(single-flow)	○	○	○	○	○
	다중(multiple-flows)	○	○	○	×	○
서비스 제공	웹서비스 인터페이스(wsdl)	○	○	○	○	○
프로세스 모델	협업(collaboration)기반 모델	○	○	○	×	○
상황정보 표현	정적(static)	×	×	×	○	○
	동적(dynamic)	×	×	×	×	○

토콜을 표준방식으로 기술하고 게시하기 위한 웹 서비스 기술언어인 WSDL을 서비스 인터페이스로 사용하고 있다.

최근에는 BPEL를 이용한 시멘틱 웹 서비스의 지원도 가능하지만 직접적으로 상황정보를 기술할 수 있는 수단은 제공하지 않는다. 그리고 상황인지 기반의 언어인 uWDL는 사용자를 위한 정적인 상황정보 기반의 단일 워크플로우 서비스만을 표현할 수 있다. 이에 비해 CAWL은 메시지 및 변수의 활용이 가능해져 실시간으로 발생하는 동적 상황정보의 표현이 가능하다. uWDL의 경우 상황인지를 기반으로 단일 플로우만을 지원할 수 있으므로, 프로세스를 웹 서비스 제공자간의 상호작용으로 기술하는 협업기반의 모델로 기술하는데 제약사항이 있다. 이에 비해 CAWL은 상황인지 기반의 다중 워크플로우를 지원할 수 있으로 협업 프로세스 모델링이 가능하게 되었다.

또한 CAWL에서는 워크플로우를 활성화시키는 활성자인 <activator> 원소를 추가하여 여러 개의 활성자가 각각의 워크플로우를 활성화시켜 다중-워크플로우의 지원이 가능하도록 설계하였고, 그림 7에서 볼 수 있듯이 <Invoke> 태그에 서브플로우(subflow) 속성을 추가하여 하나의 워크플로우에서 다른 워크플로우를 호출할 수 있다. 이와 같이 워크플로우를 모듈화함으로써 워크플로우에서 사전에 정의된 동일한 서비스의 제공이 필요할 때 서브플로우를 호출하여 재사용할 수 있도록 하였다. 그리고 uWDL는 링크(Link)의 기능으로 서비스 실행 조건과 전이 제어를 표현하였다. 이에 반해 CAWL에서는 링크에서의 역할을 흐름의 제어에만 국한시켰으며, 서비스 노드에서 서비스의 실행 조건과 전이 제어를 표현하여 사용자에게 제공되는 서비스의 실행 제어를 구현하였다. 이에 따라 CAWL는 유비쿼터스 컴퓨팅 환경에서 확장 또는 새롭게 정의한 기능들을 기반으로 다양한 영역에서의 서비스 제공을 위한 시나리오의 작성이 가능하게 되었다.

5. 결 론

본 논문에서는 독립적으로 존재하는 다수 워크플로우의 연결을 통해 다중 워크플로우 서비스를 표현할 수 있는 새로운 상황인지 워크플로우 언어인 CAWL을 제안하였다. 이를 위해 CAWL은 다음과 같은 네 가지 기능에 초점을 두고 설계되었다. 첫째, 여러 사용자에게 적합한 다중 워크플로우 서비스를 제공한다. 둘째, 제어 흐름 및 데이터 흐름을 포함하는 워크플로우 서비스의 흐름을

제어하기 위한 메시징 기능을 지원한다. 셋째 동적 상황정보에 대한 표현을 가능하게 한다. 마지막으로 서브플로우 개념을 바탕으로, 개별적으로 존재하는 단일 워크플로우 서비스를 여러 가지 형태로 조합하여 다양하면서 복합적인 워크플로우 서비스를 제공할 수 있도록 한다. CAWL은 이와 같은 주요 기능을 지원함으로써, 여러 사용자에게 적합한 다중의 복합 워크플로우 서비스를 제공할 수 있을 뿐만 아니라 상황인지 서비스의 자동화와 관련된 응용 개발에도 큰 도움을 줄 것으로 기대한다. 또한 미리 정의 되어있는 단일 워크플로우에 대한 재사용성을 증대시켜 워크플로우 시스템의 자원 소요 비용을 줄일 수 있는 효과를 얻을 수 있을 것이다.

향후에는 서비스 우선순위와 예외 상황 처리를 위한 워크플로우 패턴의 적용에 대한 추가적 연구를 진행할 것이며, 이질적 서비스 영역에서 발생할 수 있는 다양한 상황에 대한 CAWL의 언어 효율검증 연구와 관련 응용 개발 연구를 진행할 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 UIC'06, LNCS 4159, pp.766-775. 2006.

[2] Tang, F., Guo, M., Dong, M., Li, M., and Guan, H. 'Towards Context-Aware Workflow Management for Ubiquitous Computing', Proceedings of ICESS'08, pp.221-228, 2008.

[3] Y.Y.Cho, K.H. Shin, J.Y.Choi, C.W.Yoo, 'A Context-Aware Smart Home Service Based on uWDL', Proceedings of UIC'06, LNCS 4159, pp.756-765, 2006.

[4] L. Ardissono, A. Di Leva, G. Petrone, M. Segnan, M. Sonnessa, 'Adaptive Medical Workflow Management for a Context-Dependent Home Healthcare Assistance Service', Proceedings of the CWS'05, pp. 59-68, 2005.

[5] Anand Ranganathan, Scott McFaddin, 'Using Workflows to Coordinate Web Services in Pervasive Computing Environments', Proceedings of ICWS'04, pp. 189-197, 2004.

[6] 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.

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

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

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

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

[11] Anders Møller and Michael I. Schwartzbach, 'An Introduction to XML AND Web Technologies', Addison-Wesley, ISBN: 0321269667, 2006.

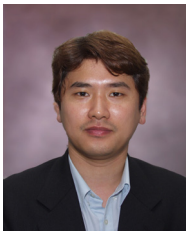
[12] Jun Li, Yingyi Bu, Shaxun Chen, Xianping Tao, Jian Lu, 'FollowMe: On Research of Pluggable Infrastructure for Context-Awareness', Proceedings of AINA'06, Volume 1, pp. 199-204, 2006.

[13] J Han, Y Cho and J Choi, 'Context-aware Workflow Language based on Web Services for Ubiquitous Computing,' Proceedings of ICCSA'05, LNCS 3481, pp. 1008-1017, Springer, 2005.

◐ 저 자 소 개 ◑

최 종 선

2000년 숭실대학교 컴퓨터학부 (학사)
2002년 숭실대학교 컴퓨터학과 (공학석사)
2003년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 시스템 소프트웨어, 병렬/분산 처리, 유비쿼터스 컴퓨팅
E-mail : jschoi@ss.ssu.ac.kr



조 용 윤

1995년 인천대학교 전산학과 졸업(학사)
1998년 숭실대학교 대학원 컴퓨터학과 (석사)
2006년 숭실대학교 대학원 컴퓨터학과 (박사)
2009~현재 국립순천대학교 정보통신공학부 조교수
관심분야 : 시스템 소프트웨어, 임베디드 소프트웨어, 유비쿼터스 컴퓨팅
E-mail : sslabyycho@hotmail.com

최 재 영

1984 서울대학교 제어계측공학과 (학사)
1986 미국 남가주대학교 컴퓨터공학 (석사)
1991 미국 코넬대학교 컴퓨터공학 (박사)
1992 ~ 1994 미국 국립 오크리지연구소 연구원
1994 ~ 1995 미국 테네시 주립대학교 연구교수
2001 ~ 2002 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원
1995 ~ 현재 숭실대학교 정보과학대학 컴퓨터학부 교수
관심분야 : 병렬/분산처리, 고성능컴퓨팅(HPC), 유비쿼터스컴퓨팅
E-mail : choi@ssu.ac.kr