퍼베이시브 시스템을 위한 상황 지식 모델링

Context Knowledge Modeling for Pervasive Systems

조 준 면[1] · 김 현[1] · 한 순 흥[2]

Joonmyun Cho[1] · Hyun Kim[1] · Soonhung Han[2]

Abstract: For the pervasive computing in ubiquitous environment, it is very important to manage the context model to provide pertinent context knowledge to context-aware applications. The context model should be able to support efficiently the context knowledge reusing and sharing as well as reasoning. Previous works focus mainly on the context knowledge *representation* scheme for reasoning. This paper proposes a context knowledge *modeling* scheme especially for reusing and sharing. This scheme provides well-established principles and guides for 1) context knowledge modularization and hierarchization, and for 2) context knowledge identification and organization. Once the context models are built according to the scheme, the structure of the context model and the meanings of the context knowledge elements become clear and consistent, so that context-aware applications can share and reuse the context knowledge in easy and error-reduced manner. This paper also discusses the implementation of a context model and an application for *Presentation Helper* scenario running on a software middleware system (CAMUS) for ubiquitous service robots which is being developed by ETRI Korea.

Keywords: Context Knowledge Modeling, Context-Aware Application, Pervasive Computing, Ubiquitous Computing, Service Robot

1. 서 론

퍼베이시브 컴퓨팅은 센서 디바이스, 서비스 디바이스, 소프트웨어 에이전트 등이 구분 없이 통합되고 서로 협동하여, 사용자의 요구를 스스로 알아 적절한 서비스를 언제 어디서나 적절한 형태로 제공하는 비전을 제시한다[1,2,3,4]. 이러한 비전을 실현하기 위해서는 유비쿼터스 객체(소프트웨어 에이전트, 센서 디바이스, 서비스 디바이스 등)들이 상황을 인식할 수 있는 기반구조가 마련되어야 한다. 유비쿼터스 객체들은 상황의 변화에 맞추어 자신을 적응시킬 수 있어야 하기 때문이다.

'상황'은 주위환경 (예를 들어, 방의 온도, 소음 수준, 조명 수준 등), 위치 등을 포함한, 사용자와 디바이스, 소프트웨어 에이전트들의 상태에 관한 지식을 의미한다. 또한, 상황 지식은 사용자 또는 디바이스, 소프트웨어 에이전트들이 관여하는 행위, 역할, 의도까지 포함할 수 있다[1,5].

퍼베이시브 컴퓨팅에서 상황 인식을 지원하는 기반 구조의 핵심 기능은, 상황 모델을 관리하고 이를 바탕으로 적절한 상황 지식을 제공하는 데 있다. 구체적으로 상황 모델은 1) 상황 지식 추론, 2) 상황 지식 재사용, 3) 상황 지식 공유를 효과적으로 지원할 수 있어야 한다[5,6].

기존의 상황 기반 컴퓨팅 연구들은 주로 상황 지식 추론에 초점을 맞추었다. 능동적 (proactive)이고 지능적인 서비스를 제공하기 위해, 상황 모델에 명시적으로 표현하지 않은 암묵적인 지식을 유도하거나, 센서로부터 취득된 환경의 상황 데이터를 상위 수준의 상황 지식으로 해석하는데 추론이 요구되기 때문이다. 그러나 상황 지식의 공유와 재사용에 대해서는 상대적으로 관심을 두지 않았다.

※ 본 연구는 정보통신부 정보통신 선도 기반기술 개발 사업 – 능동형 서비스를 위한 URC 서버 프레임워크 개발 과제 지원으로 수행되었음.

[1] 한국전자통신연구원 소프트웨어로봇연구팀
Software Robot Research Team, Electronics and Telecommunications Research Institute(ETRI)

[2] 한국과학기술원 기계공학과
Dept. of ME, Korea Advanced Institute of Science and Technology (KAIST)

본 논문은 퍼베이시브 컴퓨팅을 지원하기 위한 미들웨어 시스템 개발 경험을 바탕으로, 상황 지식의 공유와 재사용이 이론적인 측면에서뿐만 아니라 실제 상황 기반 응용의 구현 측면에서도 중요한 요소임을 제시한다. 그리고 상황 지식 공유와 재사용을 위해 상황 지식을 어떻게 *표현*할 것인가 보다는, 상황 지식을 어떻게 *마련*할 것인가에 초점을 맞추어, 새로운 *상황 지식 모델링 방법*을 제안한다. 잘 정비된 상황 지식 모델링 원리와 지침 없이 상황 모델을 개발하게 되면, 상황 지식의 공유와 재사용은 귀찮은 작업이 된다. 예를 들어, 상황 기반 응용 또는 상황 기반 서비스가 필요한 지식을, 상황 모델의 어디에서 어떻게 찾아야 하는지, 변화된 상황을 어디에 어떻게 반영해야 하는지를 직관적으로 판단할 수 없기 때문에, 상황 기반 응용 개발자가 상황 모델을 자세히 숙지하고 있어야 하고, 개발된 응용을 디버깅하는 데에도 더 많은 시간이 요구된다.

본 논문은 다음과 같이 구성된다. 2절에서 상황 기반 퍼베이시브 응용에 관한 기존 연구들을 분석한다. 3절에서는 한국전자통신연구원 (ETRI)에서 개발중인 퍼베이시브 컴퓨팅 환경을 위한 소프트웨어 기반구조 (CAMUS)를 소개하고, 간단한 응용 시나리오와 상황 지식 공유 및 재사용의 필요성을 살펴본다. 4절에서 구체적으로 상황 지식 모델링 방법을 설명한다. 5절에서는 4절의 결과를 적용하여 실험한 결과를 설명하고, 제안된 상황 지식 모델링 방법의 유용성을 검증한다.

2. 관련 연구

기존 연구들에서 제안한 상황 모델은 지식 표현 방법 (scheme)에 따라 엄밀 (formal)한 것과 엄밀하지 않은 (informal) 것으로 구분할 수 있다[6]. 엄밀하지 않은 상황 모델은 보통 고유한 지식 표현 방법에 기반한다. Context Toolkit [1]의 경우 속성-값 튜플로 상황 지식을 표현하였고 Cooltown [2]의 경우 웹 기반 모델을 이용하여 객체들에 각자의 웹 명세를 부여하여 상황을 표현하였다. Henricksen의 연구 그룹 [3]은 ER과 UML 언어를 사용하여 상황 지식을 표현하였다. 엄밀하지 않은 상황 모델은 상황 지식을 추론하는데 어려움이 있다.

반면, 엄밀한 상황 모델은 엄밀한 지식 표현 체계를 사용하여 상황 지식을 표현하기 때문에, 일정 수준의 상황 지식 추론을 지원한다. Ranganathan의 연구 그룹 [4]은 Gaia 시스템에서 DAML+OIL [7]로 표현된 일차 술어식을 사용하였다. Wang의 연구 그룹 [6], Chen의 연구 그룹 [5], 그리고 김학래의 연구 그룹 [8]은 OWL 웹 온톨로지 언어를 기반으로 상황 모델을 구축하였다. 특히 이 연구들은 온톨로지 지식을 명시적으로 표현하여 활용할 것을 제안하였다.

그러나, 기존 연구들은 상황 지식의 공유와 재사용에 대해서는 상대적으로 상세히 다루지 않고 있다. Wang은 OWL 웹 온톨로지 언어를 이용한 다른 연구와는 달리, 온톨로지를 상위 온톨로지 (upper ontology)와 도메인 온톨로지 (domain ontology)로 구분함으로써 상위 온톨로지의 재사용을 강조하였지만, 구체적으로 어떤 기준으로 상위 온톨로지와 도메인 온토로지를 구분할지, 이들을 어떻게 계층화 할지에 대해서는 설명하지 않고 있다.

더군다나 상황 지식의 공유를 위해 상황 지식을 어떻게 모델링 해야 하는지에 대해서는 구체적인 연구가 부족한 실정이다. 김병만의 연구 그룹 [9]은 주로 센서로부터 얻어지는 환경 정보를 표현하는 '환경 모델'과, 주로 응용 인터페이스를 통해 사용자와 상호작용함으로써 얻어지는 사용자의 선호사항 및 행동 정보를 표현하는 '사용자 모델'을 구분할 것을 주장하였다. 이러한 구분은 지식의 획득 방법의 차이 및 지식의 활용 형태의 차이를 반영하고 있어, 상황 지식 모델을 개발하는 사람이 대상 지식을 해석하고 개념화하는데 나름대로 유용한 모델링 관점을 제공한다. 그러나 구체적으로 어떤 지식을 사용자 모델에 표현해야 하는지, 어떤 지식을 환경 모델에 표현해야 하는지, 지식을 어떻게 구조화해야 하는지에 대해서는 논의하지 않고 있다. 상황 지식 공유를 위해서는 상황 모델을 동일한 관점에서 모델링하고 여러 상황 기반 응용들이 동일한 상황 모델을 사용하는 것만으로는 부족하다. 효과적인 지식의 공유를 위해서는 상황 모델에 어떤 상황 지식 요소 (클래스, 속성, 인스턴스)를 식별해야 하고, 이들을 어떻게 구조화해야 하는지에 대한 일관된 원리가 필요하다[10,11].

[표 1]은 구현된 상황 지식 관리 시스템과 채택하고 있는 지식 표현 방법의 특징을 고려하여 기존 연구들의 상황 지식 관리 기능을 비교 정리한 것이다.

표 1. 유사 연구의 상황 지식 관리 기능 비교

		상황 지식 추론	상황지식 재사용		상황지식 공유	
			상황지식 모듈화	상황지식 계층화	ONT 모델링 원리	IB 관리 가이드
Informal KR	Context Toolkit	X	X	X	X	X
	Cooltown	X	X	X	X	X
	Henricksen, et al.	△	△	△	X	X
Formal KR	Gaia	○	△	△	X	X
	Chen et al.	○	△	△	△	X
	Wang et al.	○	○	○	△	△
본 논문		○	○	○	○	○

○: 지원, △: 일부지원, X: 부족함
KR: Knowledge Representation, KB: Knowledge Base,
ONT: Ontology, IB: Instance Base, KB = ONTs + IBs

3. CAMUS (Context-Awareness Middleware for URC Systems)

일반적으로 로봇은 외부 환경을 센싱하고, 이를 바탕으로 판단하고, 이 판단에 따라 행동하는 세가지 기능적 요소를 갖는다. 한국전자통신연구원에서 수행하고 있는 URC (Ubiquitous Robotic Companion) 과제는, 로봇 자체에서 처리되던 이 세가지 필수 기능을 외부 유비쿼터스 환경에 분산시켜, 싼 가격에 고 기능의 로봇 서비스를 구현하는 것이 목적이다. 즉, 로봇 자체의 센싱 기능을 늘려가기 보다는 외부 환경에 내재된 센서 기능을 활용할 수 있게 하고, 기존 로봇의 프로세싱 파워를 높이기 보다는, 원격지의 고기능 서버를 다수의 로봇이 공동으로 활용할 수 있게 하자는 것이다.

CAMUS는 URC를 위한 미들웨어 시스템으로써, 유비쿼터스 컴퓨팅 환경에서 수행되는 상황 기반 응용을 개발하고 실행하는 소프트웨어 기반구조를 제공한다.

3.1 시스템 구조 및 상황 지식 관리 도구

[그림 1]은 CAMUS의 시스템 구조도를 보여준다. CAMUS 설계의 기본 개념은, 상황 기반 응용 및 센서와 액츄에이터와 같은 서비스 디바이스를 상황 지식 관리와 분리 (decouple)하는 것이다. CAMUS 시스템은 다양한 유비쿼터스 컴퓨팅 환경을 지원해야 하고, 한 환경에서도 상황이 동적으로 변화하며, 상황 기반 응용과 서비스는

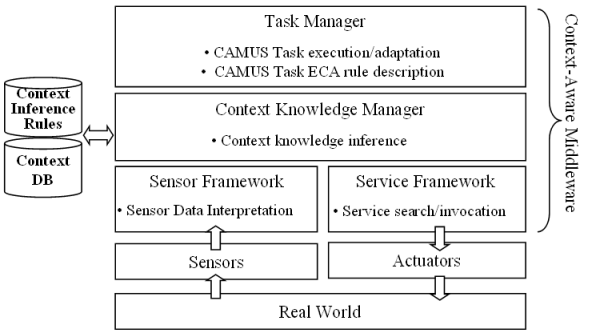

그림 1. CAMUS의 개념적 구조

이러한 변화에 맞추어 적절하게 적응해야 한다. 이러한 특성을 지원하기 위해 소프트웨어 기반구조 중 상황 지식 관리 계층은 다른 계층과 분리되어야 하기 때문이다 [3,4,5].

태스크 관리자는 개별 CAMUS 태스크를 기동시키고, 수행중인 CAMUS 태스크의 프로세스를 관리 또는 제어하는 역할을 한다. CAMUS 태스크가 바로 사용자에게 실제 서비스를 제공하는 상황 기반 응용에 해당된다.

CAMUS 태스크는 다수의 태스크 규칙 (Task Rule)으로 구성된다. 각 태스크 규칙은 ECA (Event-Condition-Action)의 표현으로, 이벤트 (event)와 조건 (condition)을 통해 적절한 상황과 사용자 또는 응용의 요구를 기술하고, 해당 상황이 발생될 때 수행할 서비스를 행동(action)으로 기술한다. ECA 태스크 규칙이 수행될 때 특히 조건 부분에서 필요한 상황 지식을 상황 지식 관리자 (Context Knowledge Manager)을 통해 참조한다.

센서 프레임워크 (Sensor Framework)는 물리 공간의 센서를 가상 공간으로 매핑하고, 이들 센서 정보로부터 상황 데이터를 추출하여 상황 지식 관리자에 제공함으로써, 상황 기반 응용이 능동적으로 서비스를 제공할 수 있도록 지원한다. 사용자로부터 발생하는 음성정보, 영상정보, 온도/습도 정보, 사용자 일정 정보 등이 모두 센서 정보가 될 수 있다. 이 계층에서는 상황 데이터의 추출뿐만 아니라, 이들을 가공 (interpretation)하는 역할도 수행한다. 예를 들어, 센서 정보를 필터링하고 조합하기도 한다.

상황 지식 관리자는 센서 프레임워크로부터 전달된 상황 정보와, 이 정보를 바탕으로 암묵적인 지식을 추론하여 별도의 저장소에 관리한다. 이 계층에서 관리하는 상황 지식은, 추후 CAMUS 태스크 즉, 상황 기반 응용이 실행될 때 참조된다. 따라서, 상황 지식 관리자는 상황 지식을 표현을 위한 상황 모델과 함께, 센서 프레임워크가 상황 정보를 추가하거나 수정하는

기능, 상황 지식 검색 기능, 그리고 암묵적인 지식을 추론하는 기능을 제공한다.

서비스 프레임워크 (Service Framework)는 조명 장치, 디스플레이 장치, 가전기기 등의 제어, 그리고 음성 처리, 일정 관리와 같이, CAMUS의 태스크가 이용할 서비스 (컴퓨터 프로그램 모듈)들에 대한 인터페이스와 이들의 구현 코드를 관리한다. 이 계층은 태스크 관리자에 의해 요구되는 서비스를 탐색하고 실제 구현 모듈을 호출하는 역할도 한다.

CAMUS에서 CAMUS 태스크와 소프트웨어 에이전트 즉, 서비스들이 상황 지식을 공유하고 추론하는데 온톨로지가 중요한 역할을 한다. CAMUS의 상황 모델 즉, 지식 베이스가 온톨로지를 기반으로 구축되기 때문이다. CAMUS는 온톨로지 기반 상황 모델을 관리하기 위해 웹 온톨로지 언어인 OWL [13]과 Jena [14] 개발 도구를 사용한다.

OWL은 다음과 같은 장점을 가진다.

- 정교하게 지식을 표현할 수 있다. 예를 들어, 클래스의 속성 정의 시 부여할 수 있는 다중값 특성 (cardinality)은, 인스턴스가 가질 수 있는 속성 값의 개수를 표현할 수 있다. 또한, 객체 (즉, 인스턴스) 간 관계 (relation)에 이행적 (transitive), 대칭적 (symmetric), 함수적 (functional)인 성질을 부여할 수 있다.
- 분산 컴퓨팅 환경을 지원하는 미리 정의된 기능을 가지고 있다. 예를 들어, OWL 온톨로지는 사전에 독립적으로 작성된 다른 온톨로지를 수입 (import)하여, 쉽게 새로운 온톨로지를 작성하거나 확장할 수 있다.
- 이론적 측면에서, OWL은 디스크립션 로직 (Description Logic) [15]과 동등하다. 따라서, OWL로 표현된 지식에 DL의 추론 규칙을 적용할 수 있다. 예를 들어, 클래스 포함관계, 인스턴스 타입과 같은 지식을 자동화된 추론을 통해 얻을 수 있다.
- 구현 측면에서, 국제 표준으로 제정되었기 때문에 다양한 개발 및 응용 도구를 쉽게 이용할 수 있다. 예를 들어, FaCT [16], RACER[17]와 같은 DL 기반 추론 엔진과, Protégé-OWL [18]과 같은 저작 도구, 그리고 Jena와 같은 응용 개발 도구들이 이미 시장에 출시되어 있다.

한편, Jena는 다음과 같은 장점을 제공한다.

- 지식을 생성, 수정하고 탐색할 수 있는 다양한 API를 제공한다.
- SQL (structured query language)과 유사한 구조화된 검색 언어 SPARQL (SPARQL Protocol and RDF Query Language)를 제공한다.
- 파일 시스템 또는 RDB를 하부 (back-end) 시스템으로 사용하는 영속적 저장 (persistent storage) 기능을 제공한다.
- 다른 기능과 잘 통합된 다양한 추론엔진을 제공한다. 또한 제공되는 추론 엔진은 필요에 따라 여러 개를 함께 사용할 수 있다.
- 사용자 정의 추론 룰 (user-defined reasoning rule)을 정의하고 적용할 수 있다.

[그림 2]는 OWL과 Jena 응용 개발 도구를 기반하여 구현된 CAMUS의 상황 지식 관리자 (Context Knowledge Manager) 내부에서 상황 기반 응용 즉, CAMUS 태스크가 적절한 상황 지식을 이용할 수 있도록 상황 모델이 관리되는 구조를 보여준다. 그림에서는 개발 시점과 동작 시점을 구분하였다.

우선, 개발 시점에 Protégé-OWL과 같은 OWL 기반 지식 베이스 (knowledge base) 저작도구를 이용하여 초기 상황 모델을 작성하고, Jena가 제공하는 추론 규칙 정의 언어를 이용하여 응용에 특화된 추론 규칙을 작성한다. 동작 시점에 초기 상황 모델은 [그림 1]의 상황 지식 관리자가 관리하는 영속적인 저장소에 로딩된다. 이렇게 로딩된 상황 모델은 외부에는 RDF (resource description framework) 모델 즉, Triple 형태의 표현식을 통해 제공된다. 이후 상황 지식 관리자는 OWL 추론 엔진을 적용하여 '추론된 모델'을 생성한다. 추론된 모델도 외부에는 RDF 모델을 통해 제공되지만, 상황 모델에 명시적으로 표현되지 않은 암묵적인 지식들이 유도되어 포함된다는 것이 차이점이다. OWL 추론 엔진이 적용될 때, 개발자에 의해 작성된 응용에 특화된 추론 규칙이, DL의 추론 규칙과 함께 적용된다. 상황 기반 응용들은 SPARQL을 지원하는 검색 엔진을 통해 또는 Jena 모델 API를 이용하여 상황 모델에 접근하여 필요한 상황 지식을 검색하거나 상황 지식을 생성/변경한다.

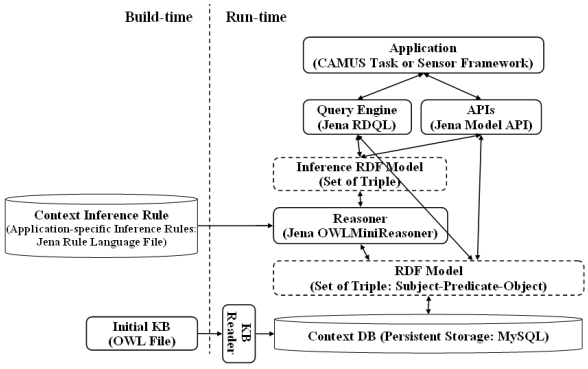

그림 2. CAMUS의 상황 기반 응용 실행 구조

3.2 적용 환경 및 발표 도우미 시나리오

CAMUS는 아파트 또는 사무실과 같은 주거환경에 적용되리라 예상된다. 고성능의 중앙 서버에 CAMUS의 핵심 모듈 (태스크 관리자, 상황 지식 관리자 등)과 함께, 음성 인식, 화상 인식과 같이 큰 계산 능력이 요구되는 프로그램 모듈이 탑재된다. 각 가정이나 사무실에는 센서와 이들을 관리하는 CAMUS의 센서 프레임워크, 그리고 다양한 홈 네트워크 디바이스와 이들을 관리하는 CAMUS의 서비스 프레임워크가 설치된다. 중앙 서버는 다수의 유비쿼터스 환경 (가정, 사무실)에 의해 공유된다. 각 유비쿼터스 환경은 다시 작은 단위의 하위 환경을 포함할 수 있다. 예를 들어, 가정 환경의 경우 안방, 거실, 주방과 같은 하위 환경을 포함한다.

각 환경의 상황 데이터는 센서에 의해 감지되고, 일차적으로 센서 프레임워크에 의해 가공되어, 중앙 서버의 CAMUS 상황 지식 관리자로 전달된다. 이렇게 모아진 상황 지식은 중앙 서버의 CAMUS 태스크 관리자에서 실행되는 상황 기반 응용 즉, CAMUS 태스크에 의해 참조되어 적절한 서비스를 능동적으로 제공하게 된다.

본 논문에서 제안하는 상황 지식 모델링 방법을 발표 도우미 시나리오 구현에 적용하여 상황 모델을 구축하고 발표 도우미 CAMUS 태스크를 개발하여 상황 지식의 공유와 재사용의 용이성을 검증하였다. 발표 도우미 CAMUS 태스크는 RFID (Radio Frequency Identification) 센서, 음성 센서 (마이크), 빔 프로젝터가 설치된 사무실 환경에서 실행된다. RFID 센서는 전체 사무실을 담당하는 것과 발표 장소를 담당하는 것 두개가 설치된다.

사람이 사무실에 입장하면 사무실 RFID 센서가 사람과 위치 정보를 센싱하여 CAMUS의 상황 지식 관리자에 전달한다. 좀더 자세히 설명하면, RFID 태그를 가진 사람이 RFID 센서가 감지하는 공간 범위 내로 진입하면, RFID 센서가 해당 태그의 일련번호와 자신의 ID 정보를 CAMUS의 센서 프레임워크로 전달한다. 센서 프레임워크는 상황 모델에 미리 등록된 태그의 인련번호를 비교하여, 해당 태그를 가진 사람이 누구인지를 판단하고, 태그를 인식한 RFID 센서의 위치 정보를 이용하여 해당 사람의 위치를 해석하여 RFID 태그를 가지고 있는 사람이 누구인지와 그 사람의 위치가 어디인지를 상황 지식 관리자에게 전달한다.

상황 지식 관리자는 사무실에서 현재 시간에 예정된 회의 정보와 사람의 위치 정보를 비교하여, 사무실에 입장한 사람을 회의의 참석자로 해석하여 상황 모델에 반영한다. 회의 주관자가 음성으로 회의 시작을 명령한 후 사전에 상황 모델에 등록된 발표자가 발표 장소로 이동하면, 발표 장소 RFID 센서가 이를 센싱하여, 해당 발표자의 발표자료를 빔 프로젝터을 이용하여 스크린에 보여 주고, 발표 상태를 진행중으로 수정하여 상황 모델에 반영한다. 발표자가 음성으로 발표 종료를 명령하면, 스크린 프로젝션을 마치고 발표 상태를 종료로 수정하여 상황 모델에 반영한다. 회의 주관자가 음성으로 회의 종료를 명령하기 전까지 위의 과정을 계속 반복한다.

회의 시작, 회의 종료, 발표 종료를 위한 음성 명령과 그 밖의 상황 변화는 항상 상황 모델에 저장된 지식과 추론 규칙에 의해 해석된다. 예를 들어, 발표 주관자가 아닌 사람이 회의 시작이나 회의 종료를 음성 명령한 경우, 아무런 동작을 수행하지 않는다. 마찬가지로, 발표자가 아닌 사람이 발표 종료를 명령한 경우도 아무런 동작을 수행하지 않는다. 또한, 이미 발표를 마친 사람이 발표 장소로 이동하는 경우나, 현재 발표가 진행중인데 다른 발표자가 발표 장소로 이동한 경우도, 상황 모델에 저장된 발표 상태 지식을 이용하여, 원하지 않는 동작이 수행되지 않도록 한다.

3.3 상황 지식 공유 및 재사용의 필요성

CAMUS의 상황 기반 응용은 각 환경을 기반으로 실행된다. 예를 들어, 발표 도우미 응용은 사무실에 설치된 RFID 센서와 빔 프로젝터를 기반으로 실행된다. 또한, 하나의 환경에 여러 개의 상황 기반 응용이 실행될 수 있다. 예를 들어 한 사무실 (사무실 A)에서 온도 조절 응용과 발표 도우미 응용이 함께 실행될 수 있다. 이런 경우, 한 상황 기반 응용의 실행에 의해 환경 즉, 상황이 변하게 되면 (예를 들어, 온도 조절 응용에 의해 사무실의 온도가 내려가면) 다른 응용에 그 영향이 미친다. 여러 응용이 하나의 환경을 공유하기 때문이다. 따라서, 하나의 상황 모델을 여러 개의 상황 기반 응용이 공유해야 한다. 한편, 여러 환경에서 동일한 상황 기반 응용이 실행될 수 있다. 예를 들어, 사무실 B에서도 온도 조절 응용과 발표 도우미 응용이 실행될 수 있다. 따라서, 사무실 A의 상황 기반 응용을 위해 개발된 기존의 상황 모델을 재 사용할 수 있어야 한다.

CAMUS를 실제 환경에 적용하는데 있어 또 한가지 주목할 것은 다양한 조직 또는 사람이 역할 분담을 통해 참여한다는 것이다. 상황 기반 응용의 개발로 국한하여 생각해 보더라도, 온도 조절 응용, 발표 도우미 응용과 같은 모든 상황 기반 응용을 하나의 조직 또는 한

사람이 개발하는 것이 아니라, 독립적인 조직과 사람들이 서로 다른 시점에 개발하고 구축하게 된다. 따라서, 더더욱 상황 지식의 효율적인 공유와 재사용이 요구된다.

4. 상황 지식 모델링 방법

상황 지식의 재사용을 위해서는 상황 모델이 모듈화되고 계층화 되어야 한다. 어떤 지식은 여러 응용에서 공유되고 재사용 되지만 어떤 지식은 특정 응용에 대해서만 의미를 가지고 따라서 공유되거나 재사용되지 않기 때문이다. 따라서 이런 특성을 잘 고려해서 공유와 재사용이 빈번히 일어나는 지식과 그렇지 않은 지식을 명확한 기준에 의해 분리하여 모듈화 할 필요가 있다. 더 나아가, 상황 기반 응용이 필요한 상황 지식 모듈을 쉽게 사용할 수 있도록 해야 한다. 이를 위해 상황 지식 모듈들을 계층적인 구조로 만들어 하나의 상황 지식 모듈을 선택하였을 때 포함되는 모듈과 포함되지 않는 모듈이 자연스럽고 명확하게 구분되도록 할 필요가 있다. 일반적으로 공유와 재사용이 빈번히 요구되는 지식은 그렇지 않은 지식보다 상위 계층을 구성한다.

상황 지식이 적절하게 모듈화되고 계층화된다면, 기존에 작성된 상황 모델과 유사한 상황 모델을 재작성하거나 필요하지 않은 지식을 어쩔 수 없이 상황 모델에 포함시켜 상황 기반 응용을 실행해야 하는 경우를 줄일 수 있다. 더 나아가, 잘 모듈화되고 계층화된 상황 지식은 나중에 필요에 따라 수정하거나 확장하기 수월하다.

상황 지식의 공유를 위해서는 각 상황 모델이 동일한 개념화 결과를 가져야 한다. 상황 지식의 공유는 여러 상황 기반 응용과 유비쿼터스 객체들이 공통의 상황 모델을 사용함으로써, 동일한 상황 지식 요소 즉, 동일한 클래스 (class), 속성 (property), 객체 (individual), 관계 (relation)를 사용하는 것 만으로는 부족하다. 상황 모델에 표현된 상황 지식 요소들이 명확하고 일관된 의미를 가져야 한다. 예를 들어, *발표자* 라는 클래스는 독자적으로 존재 가능한 개념인지 즉, 자신의 인스턴스들을 시간과 상황에 관계없이 식별/재식별할 수 있는 식별조건을 가지는지, 아니면 반드시 *사람*이라는 클래스의 하위 클래스로서만 존재 가능한지 즉, 상위의 클래스로부터 식별조건을 물려 받아야 하는지와 같은 의미가 명확하고 일관되어야 한다.

상황 지식 요소들이 명확하고 일관된 의미를 가질 때, 상황 기반 응용이 필요한 지식을 어디에서 어떻게 찾아야 하는지, 변화된 상황을 어디에 어떻게 반영해야 하는지를 쉽고 직관적으로 판단할 수 있어, 여러 유비쿼터스 객체들이 오류 없이 상황 지식 요소를 공유할 수 있기 때문이다. 더 나아가, 상황 정보의 생성 또는 변경이 다른 상황 정보에 어떤 영향을 주는지 쉽게 파악할 수 있어, 응용 개발 시점에 상황 지식의 불일치와 같은 오류를 쉽게 찾아내고 방지할 수 있다.

본 논문에서 새롭게 제안하는 상황 지식 모델링 방법은 1) 어떤 기준으로 상황 지식을 모듈화하고 계층화해야 하는지, 2) 상황 모델에 어떤 상황 지식 요소를 식별하고 이들을 어떻게 구조화해야 하는지에 대한 가이드를 제공한다.

4.1 상황 지식 모듈화 및 계층화

상황 지식은 가정, 사무실과 같은 상황 기반 응용이 실행되는 환경 즉, 응용 도메인에 따라 모듈화될 수 있다. 실제로 상황 기반 응용을 실행되는 환경에 따라 그룹핑되고 각 응용 그룹은 함께 관리되는 것이 일반적이다.

한편, 상황 지식 모듈은 추상화 수준에 따라 계층적으로 구조화 될 수 있다. 일반적으로 추상화 수준이 높은 지식은 여러 상황 기반 응용에 의해 빈번히 재사용되고 공유된다. 예를 들어 장소, 사람, 장치와 같은 지식은 상황 지식의 뼈대를 구성하고 다른 관련된 지식의 인덱스 역할을 하기 때문에 다양한 상황 기반 응용에서 사용되며 지식의 계층 구조상에서 상위에 위치한다.

논문에서 제안하는 상황 지식 모델링 방법에서는 상황 지식의 계층구조를 3계층으로 구분한다. 최상위에는 *공유 온톨로지*가 위치한다. 공유 온톨로지는 일반적으로 CAMUS 시스템이 구축되는 시점에 배포된다. 여러 환경 (또는 도메인)에서 수행되는 여러 응용 또는 서비스에 공통적인 온톨로지 개념 (클래스, 속성)을 정의한다. 상황 기반 응용과 서비스에는 최상위 온톨로지 지식을 제공하고, 하위의 *도메인 온톨로지*들에는 유사한 집성화 (aggregation) 수준과 유사한 정도 (granularity)의 온톨로지 개념들이 정의되도록 가이드하는 역할을 한다.

도메인 온톨로지는 일반적으로 상황 기반 응용 또는 서비스가 개발되는 시점에 작성된다. 공유 온톨로지에 정의된 상위 클래스와 속성을 상속받아, 해당 도메인과 개발되는 응용에 특화된 좀더 상세화된 클래스와 속성을 정의한다. 공유 온톨로지에 정의된 지식 요소만으로는 특정 도메인에서 수행되는 응용에 필요한 상황 지식을

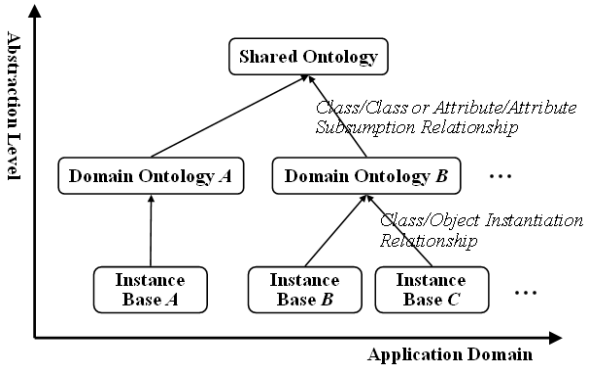

그림 3. 상황 지식의 구성 – 상황 지식의 모듈화 및 계층화

표현하기 어렵기 때문이다. 실제적으로는 상위 *공유 온톨로지*를 수입 (import)하여 작성된다. 상황 기반 응용과 서비스에는 도메인 온톨로지 지식을 제공하고, 하위의 *인스턴스 베이스*에는 실제 객체와 객체 간의 관계를 생성하고 수정하는 스키마의 역할을 한다.

가장 하위에는 실제 객체들에 대한 지식을 표현한 *인스턴스 베이스*가 위치한다. 인스턴스들은 일반적으로 상황 기반 응용이 실행되는 시점에 생성되고 계속적으로 수정된다. 상황 기반 응용과 서비스에 실 세계의 상황 데이터를 제공하는 역할을 한다.

[그림 3]이 지금까지의 논의를 바탕으로 상황 모델의 구성을 도식적으로 보여준다. 이렇게 상황 모델이 상위 수준의 지식을 포함하는 것과, 특정 응용 환경에 특화된 하위 수준의 지식을 포함하는 것으로 계층화되고, 적용 도메인에 따라 모듈화되면 상황 기반 응용이 실행되는 환경과 요구되는 상황 지식의 정도에 따라, 적절한 상황 모델을 선택하여 이용할 수 있다.

4.2 상황 지식의 식별 및 조직화

상황 지식 요소들이 일관된 의미를 갖는 문제는, 어떤 개념을 클래스로 식별해야 하는지, 클래스들의 Taxonomy를 어떻게 구성해야 하는지, 클래스에 어떤 속성을 정의해야 하는지, 속성 중 어떤 것을 클래스의 멤버쉽 조건으로 정의해야 하는지, 속성을 클래스 Taxonomy의 어느 레벨에서 부여해야 하는지, 어느 클래스에서 인스턴스를 생성해야 하는지와 같은 질문으로 나타난다[10,11,19,20].

Guarino의 상위 온톨로지 이론[21]은 *식별성* (*identity*), *영속성* (*rigidity*), *의존성* (*dependence*)과 같은 존재론적 본성 (ontological nature)을 이용하여, 실세계 개념을 *TYPE*, *QUASI-TYPE*, *MATERIAL ROLE*, *PHASED SORTAL*, *CATEGORY*, *ATTRIBUTION* 등의 종류 (ontological distinction)로 구분하고, 개념 종류에 존재론적 본성을 바탕으로 명확한 특성과 논리적 제약 조건을 부여한다. [표 2]에 존재론적 개념의 종류와 각각의 존재론적 본성을 정리하였다.

표 2. Guarino 온톨로지 이론의 존재론적 구분과 이들의 존재론적 본성 [21]

개념 종류 (Ontological Distinctions)	예	존재론적 본성 (Ontological Natures)			
		식별성 (Identity)		영속성 (Rigidity) (R)	의존성 (Dependence) (D)
		공급 (O)	전달 (I)		
CATEGORY	*실개체, 추상개체*	–	–	+	+ –
TYPE	*사람, 고양이*	+	+	+	+ –
PHASED SORTAL	*애벌레, 나비*	–	+	~	–
MATERIAL ROLE	*학생, 식량*	–	+	~	+

*식별성*은 개념이 식별조건 (identity condition: IC)을 제공하는지에 관련된다. 식별조건은 한 객체를 특정 개념의 인스턴스로 식별하고, 이들을 개별적으로 구분하며 상황 및 시간에 상관없이 재 식별할 수 있는 조건을 말한다. 예를 들어, *사람* 개념은 *지문*이라는 조건을 제공하는데, 이 조건을 이용하여 특정 객체를 언제나 어떤 상태에서도 구분하고 재 식별할 수 있다. 그런데 식별조건을 스스로 공급 (supply)하는 개념 ([표 2]에서는 +*O*로 표기됨)과 다른 개념으로부터 상속받아 전달 (carry)하는 개념 (+*I*로 표기됨)을 구분할 필요가 있다. *학생*과 같은 개념은 자신만의 새로운 식별조건 (예: *지문*)을 공급하지 못하고 *사람* 개념으로부터 물려 받아 전달한다. 한편, *학생* 개념도 *학번*과 같은 조건은 공급할 수 있지만, 이러한 조건은 특정 상황 또는 특정 시점에서만 유효하다. 따라서, *학번*과 같은 조건을 *지문*과 같은 조건과 구분할 필요가 있다. 전자를 *로컬* 식별조건 후자를 *글로벌* 식별조건이라 한다.

*영속성*은 개념이 해당 개념의 모든 인스턴스에 본질적인지에 관련된다. 본질적이라는 일반 개념은 필수적이라는 일반 개념과 연결된다. 또한, 시간 또는 양태의 변화에도 관련된다. 예를 들어, *사람* 개념은 일반적으로 영속적 (+*R*로 표기됨)이라 판단할 수 있다. 어떤 객체가 사람의 인스턴스이면 모든 가능태 (possible world)에서 반드시 사람의 인스턴스임이 자명하기 때문이다. 반대로 *학생* 개념의 경우 일반적으로

영속적이지 않다고 판단할 수 있다. 동일한 객체임에도 불구하고 특정 시간 또는 특정 상황에서는 학생임이 참이지만, 다른 시간 또는 상황에서는 거짓일 수 있기 때문이다. 영속적이지 않은 개념은 다시 반영속적 (anti-rigid, ~*R*로 표기됨)이거나 약영속적 (semi-rigid, –*R*로 표기됨)일 수 있다. 반영속적인 개념은 *학생*, *식당*과 같은 개념으로, 적어도 하나의 가능태에서는 모든 인스턴스가 참임을 보장할 수 있는 경우이다 (그러나, 다른 가능태에서는 거짓일 수 있다). 약영속적인 개념은 하나의 가능태에서 조차 참일 수도 있고 거짓일 수도 있는 것들을 말한다. 예를 들어, *딱딱함*이라는 개념은 *스폰지*라는 하나의 가능태에서도 스폰지가 물에 젖은 경우는 거짓이고 마른 경우는 참일 수 있다.

*의존성*은 한 개념의 인스턴스가 참이기 위해 다른 개념의 인스턴스가 존재해야 하는지에 관련된다. 예를 들어, *학생* 인스턴스는 *학교* 인스턴스가 존재할 때만 참일 수 있다.

본 논문은 위에서 간단히 소개한 Guarino의 상위 온톨로지 이론 적용하여, 상황 모델을 다음과 같이 모델링하고 관리할 것을 제안한다. 온톨로지 (공유 온톨로지, 도메인 온톨로지)에는 Guarino 이론의 *CATEGORY*, *TYPE*, *PHASED SORTAL*, *MATERIAL ROLE*에 해당되는 개념을 클래스로 정의한다. *CATEGORY* 클래스는 최상위에 정의한다. *CATEGORY* 클래스는 영속적이지만 식별조건을 공급하지도 못하고 전달하지도 못하며, 따라서, 명확하게 한정된 멤버쉽 조건을 가질 수 없기 때문이다. 이 클래스는 대상 도메인의 상황 지식을 유용한 구획으로 나누어 구분하는 역할을 한다. *TYPE* 클래스는 최상위에 정의하거나, 다른 *TYPE* 클래스 또는 *CATEGORY* 클래스의 하위 클래스로 정의한다. *TYPE* 클래스는 영속적이며 글로벌 식별조건을 스스로 공급한다. *TYPE* 클래스가 다른 *TYPE* 클래스를 상세화하는 경우, 상위 클래스의 글로벌 식별조건을 상속받고, 추가적으로 자신만의 식별조건을 공급한다. *PHASED SORTAL* 클래스는 *TYPE* 클래스의 하위 클래스로 정의한다. 글로벌 식별조건을 상속받아야 하기 때문이다. *PHASED SORTAL* 클래스는 반영속적이고 비의존적이며, 새로운 글로벌 식별조건을 공급하지는 못하지만 로컬 식별조건을 공급한다. 즉, 시간이나 상황에 따라 변하는 식별조건을 가진다. 예를 들어, *에벌레*와 *나비*는 동일한 객체가 변태한 것이다. 이러한 경우 이 객체의 식별 조건 중 일부 (로컬 식별조건)가 시간이나 상황에 따라 변할 수 있다. *MATERIAL ROLE* 클래스는 *TYPE* 클래스 하위나 *PHASED SORTAL* 클래스 하위에 정의한다. 이 클래스는 반영속적이고 어떤 상황에서도 의존적이다. 보통 객체 간의 관계로 나타나는 특정 이벤트에 있어 해당 객체에 의해 수행되는 역할에 해당된다. 역할을 수행하는 실제 객체의 식별조건을 물려받아야 한다.

식별조건 (글로벌 식별조건, 로컬 식별조건)은 해당 클래스에 속성으로 부여된다. 클래스의 여러 속성 중 식별조건에 해당되는 속성은 적어도 반영속적이고 의존적인 것이어야 한다. 우선, 시간과 상황에 상관없이 언제나 인스턴스를 구분하고 개별적으로 식별할 수 있기 위해서, 적어도 해당 클래스에는 본질적이어야 하기 때문이다. 또한, 해당 클래스에 의존적이어야 한다. 만약, 의존적이지 않고 따라서 인스턴스가 해당 속성으로 어떠한 값도 가질 수 있다면, 인스턴스를 개별적으로 식별하는데 아무런 도움을 줄 수 없기 때문이다. 본 논문은 OWL을 사용하여 상황 모델을 작성한다. 따라서 실제로는 식별조건에 해당되는 속성은 상황 모델에서는 해당 클래스의 필요조건으로 표현된다. 그런데, 식별조건을 가지는 개념 중 *MATERIAL ROLE*에 해당되는 개념은 의존적이기 때문에 모든 식별조건을 다른 개념으로부터 물려 받는다. 따라서, *MATERIAL ROLE* 개념을 표현한 클래스에서 식별조건은 필요충분 (necessary and sufficient)한 멤버쉽 조건으로 정의된다. 한편, 클래스의 속성 중 식별조건에 해당되지 않는 것들은 필요하지도 않고 충분하지도 않는 속성으로 정의된다. OWL을 사용하는 상황 모델에서는 속성의 도메인 (domain)으로 표현된다.

인스턴스 베이스에는 *TYPE* 클래스나 *PHASED SORTAL* 클래스의 인스턴스만 생성하고 관리한다. Guarino의 상위 온톨로지 이론이 제안한 개념 종류의 존재론적 의미에 따르면, *CATEGORY* 클래스나 *MATERIAL ROLE* 클래스는 직접적인 인스턴스를 가질 수 없다. *CATEGORY* 클래스는 명확한 멤버쉽 조건을 가지지 못하고, *MATERIAL ROLE* 클래스에 속하는 객체는 실제로는 *TYPE* 클래스나 *PHASED SORTAL* 클래스의 인스턴스이기 때문이다. 또한, 인스턴스 베이스에 객체를 생성할 때 멤버쉽 조건으로 정의된 속성 즉, 식별조건에 해당되는 속성은 반드시 함께 생성한다. 한편, 멤버쉽 조건으로 정의되지 않은 속성은 상황 기반 응용이나 서비스가 실행될 때 필요에 따라 생성한다.

Guarino의 상위 온톨로지 이론이 제안한 *PHASED SORTAL* 클래스의 존재론적 의미를 따르면, 이 클래스의 인스턴스는 상위 *TYPE* 클래스 인스턴스가 특정 상황에 다른 상태로 나타난 것이다. 이러한 의미를 일관되고

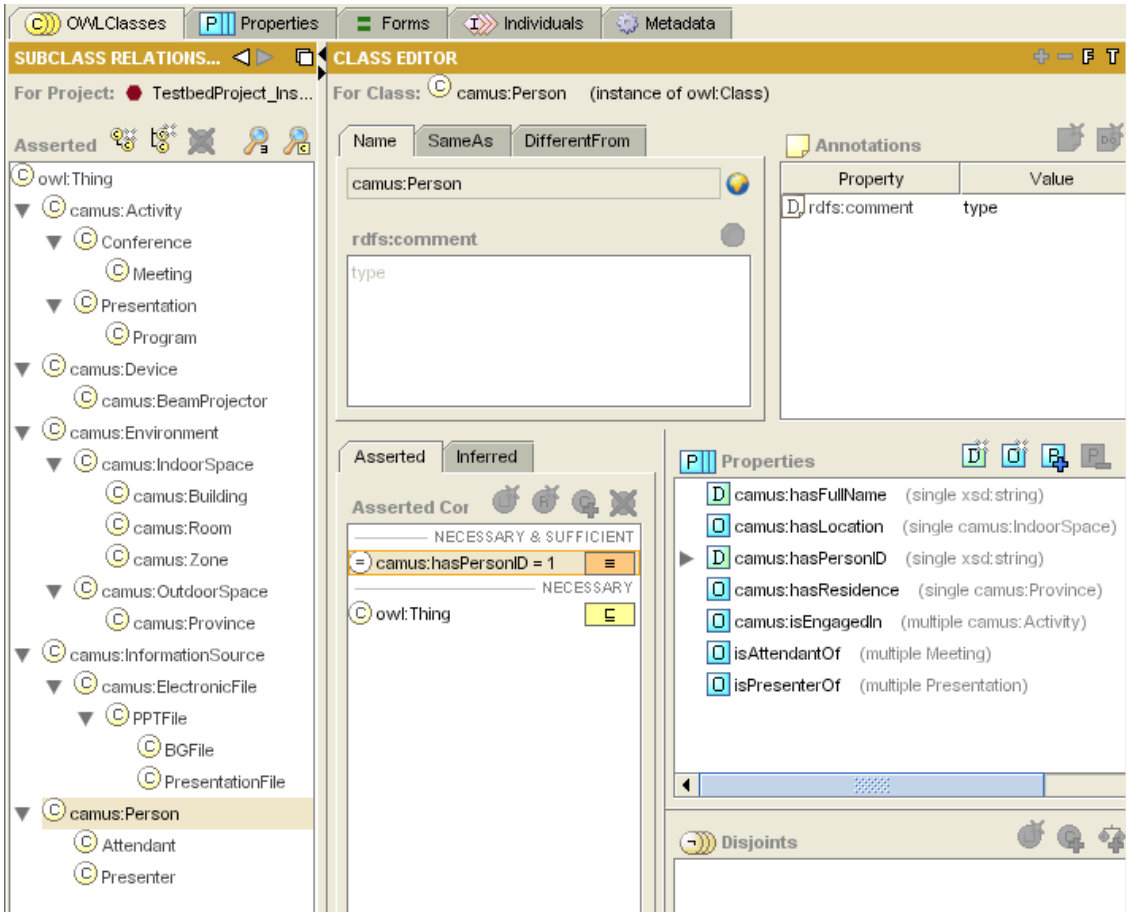

그림 4. 발표 도우미 응용을 위한 상황 모델

논리적으로 표현하기 위해, *PHASED SORTAL* 클래스의 인스턴스와 이 클래스에 대응하는 상위 *TYPE* 클래스의 인스턴스가 같은 것임을 명시적으로 표현한다 (OWL의 "sameAs" axiom으로 표현한다.

5. 프로토타입 구현과 실험

발표 도우미 시나리오를 위한 상황 모델을, 4절에서 설명한 상황 지식 모델링 방법을 적용하여 작성하고, 발표 도우미 응용 (즉, CAMUS 태스크)를 구현하여 실험하였다. [그림 4]는 작성된 상황 모델을 보여주는 Protégé-OWL OWL 지식 베이스 저작도구의 화면이다.

상황 모델은 공유 온톨로지, 도메인 온톨로지, 그리고 인스턴스 베이스로 구성된다. 그림에서 "camus"라는 네임스페이스 접두사 (namespace prefix)가 표시된 상황 지식 요소들이 공유 온톨로지에 정의된 것들이다. 공유 온톨로지에는 *Activity*, *Device*, *Environment*, *Information Source*, *Person*과 같이, 여러 상황 기반 응용과 도메인에서 공통적으로 사용할 수 있는 추상화 수준이 높은 클래스와 속성들을 정의하였다. 도메인 온톨로지에는 *Conference*, *Meeting*, *PPTFile*, *Attendant*, *Presenter*와 같이, 특정 응용 (발표 도우미 응용)에 특화된 추상화 수준이 낮은 클래스와 속성을 정의하였다. 그리고, 도메인 온톨로지는 공유 온톨로지를 수입하여 재사용하였다.

클래스들은 4.2절에서 설명한 온톨로지 개념 식별과 조직화를 위한 원리와 가이드에 따라 정의되었다. 예를 들어, *Person* 클래스는 영속적이고, 인스턴스를 개별적으로 구분할 수 있는 식별조건을 스스로 공급하기 때문에 *TYPE* 종류의 클래스로 모델링 하였다. *hasPersonID*가 *Person* 클래스의 식별조건으로 모델링 되었다. 따라서, *hasPersonID*는 *Person* 클래스의

속성으로 정의되었고 필요조건으로 표현되었다.

Person 클래스에는 *hasPersonID*와 같은 식별조건에 해당되는 속성뿐만 아니라 *isAttendantOf*와 같은 속성도 함께 정의 되었다. *isAttendantOf* 속성은 *Attendant* 클래스의 속성이고 따라서, *Attendant* 클래스에 정의되어야 한다고 생각할 수 있다. 그러나, *isAttendantOf* 속성을 소유하는 실제 사람 객체는 *Person* 클래스의 인스턴스이다. *Person* 클래스와 같은 *TYPE* 종류의 클래스만이 자신의 직접적인 인스턴스를 가질 수 있기 때문이다. 한편 *Attendant* 클래스와 같이 영속적이지 않으며 항상 의존적인 클래스는 자신의 직접적인 인스턴스를 가질 수 없고, 상위 부모 클래스로 정의된 영속적인 클래스의 인스턴스가 특정 상황 또는 특정 조건을 만족할 때 간접적인 인스턴스로 분류된다. 따라서 *isAttendantOf* 속성은 *Person* 클래스에 정의되고 *Attendant* 클래스에 상속되며, *Attendant* 클래스의 필요충분한 멤버쉽 조건 정의에 사용된다.

Attendant 클래스는 *MATERIAL ROLE* 개념 종류에 해당된다. 위에서 설명한 것처럼, 자신만의 새로운 식별조건을 공급하지 못하고 *Meeting* 클래스 인스턴스에 항상 의존적이기 때문이다. 또한, 식별조건을 *Person* 클래스 인스턴스로부터 물려 받아야 하기 때문에 *Person* 클래스의 자식 클래스로 정의되었다.

인스턴스 베이스에는 오직 *Person*과 같은 *TYPE* 종류의 클래스와 *Meeting*과 같은 *PHASED SORTAL* 종류의 클래스의 인스턴들만이 생성되고 저장된다. *Attendant*와 같은 *MATERIAL ROLE* 종류의 클래스 또는 *Activity*와 같은 *CATEGORY* 종류의 클래스는, Guarino의 상위 온톨로지 이론에 따르면, 자신의 직접적인 인스턴스를 가질 수 없기 때문이다. *Person* 클래스 인스턴스로 표현되는 사람의 위치가 *Meeting* 클래스 인스턴스로 표현되는 회의의 장소와 같을 때, *Person* 클래스 인스턴스와 *Meeting* 클래스 인스턴스는 *isAttendantOf* 속성을 통해 연결된다. 두 인스턴스의 연결을 위한 *isAttendantOf* 속성의 생성은 응용에 특화된 추론 규칙의 적용에 의해 자동적으로 수행된다. *Person* 클래스 인스턴스가 *isAttendantOf* 속성을 가지게 되면 비로서 이 인스턴스는 *Attendant* 클래스의 간접적인 이스턴스로 분류되는데, 이는 *Attendant* 클래스의 필요충분한 멤버쉽 조건이 *isAttendantOf* 속성으로 정의되었기 때문이다. 이러한 인스턴스 분류는 DL 추론에 의해 자동적으로 수행된다.

위에서 설명한 원리로부터 *isAttendantOf* 속성과 같이 비 식별조건에 해당되는 속성은 상황 변화에 따라 동적으로 생성, 변경, 삭제됨을 알 수 있다. 반면, *hasPersonID* 속성과 같이 식별조건에 해당되는 속성은 인스턴스가 생성될 때 함께 생성되고 인스턴스가 삭제될 때까지 동일한 값을 가진다는 것을 알 수 있다.

상황 기반 응용이 상황 모델을 사용하는 일반적이고 빈번한 형태는 필요한 지식을 상황 모델에서 검색하고 변화된 상황을 반영하기 위해 상황 모델 내에 인스턴스와 속성을 생성하거나 변경하는 것이다. 따라서, 논문에서 제안하는 상황 지식 모델링 방법의 효용성은 상황 모델이 이러한 작업을 얼마나 효과적으로 지원하는지를 검증함으로써 확인할 수 있다.

상황 모델이 제안된 상황 지식 모델링 방법에 의해 작성되면, 클래스, 속성 등 상황 지식 요소들의 존재론적 의미가 일관되고 명확해 지기 때문에, 상황 지식을 어디에서 어떻게 찾아야 하는지, 상황 지식을 어디에 어떻게 생성하거나 변경해야 하는지가 명확해 진다. 예를 들어, 발표 도우미 응용이 회의 참석자를 표현하는 객체를 생성할 필요가 있는 경우, 이 객체를 *Attendant* 클래스가 아닌 *Person* 클래스의 인스턴스로 생성해야 함이 명확해 진다. *Attendant* 클래스는 *MATERIAL ROLE* 종류이고, *MATERIAL ROLE* 종류의 개념은 자신의 직접적인 인스턴스를 가질 수 없고, 반면, *Person* 클래스와 같은 상위 *TYPE* 종류의 클래스의 인스턴스가 특정 상황이나 조건에 의해 간접적인 인스턴스로 분류되기 때문이다. 상황 기반 응용이 특정 위치에 있는 사람을 검색하고자 하는 경우 *Attendant* 클래스와 *Presenter* 클래스 모두에서 해당 위치 조건을 만족하는 인스턴스를 검색해야 하는 것이 아니라 *Person* 클래스에서만 검색과 속성 비교를 수행하면 되는 것이 명확해 진다. 사람 객체는 어떤 역할을 하든 상황에 따라 어떤 클래스로 분류되든 항상 *Person* 클래스의 인스턴스임이 자명하기 때문이다.

다음 코드는 어떤 위치의 사람 객체를 찾는 SPARQL 검색문의 예이다. '?' 접두사가 붙은 심볼은 변수를 의미한다. 변수는 실행 시점에 특정 값으로 바인딩된다. 아래 예를 통해 알 수 있듯이 검색문이 매우 간단해진다. 만약 지식 요소의 의미가 불명확하여 한 응용 (예, 발표 도우미 응용)이 사람을 표현하는 객체를 *Attendant* 클래스의 인스턴스로 생성했다고 가정하면, 상황 모델을 공유하는 또 다른 응용 (예, 온도 조절 응용)이 해당 사람 객체와 관련 정보를 검색하기 위해서는 여러 클래스에 걸쳐 그때그때 다른 방법으로 검색해야 할 것이다.

```
SELECT ?person
WHERE  (?user, <rdf:type>,  camus:Person),
       (?user, <camus:hasLocation>, ?loc),
USING  camus FOR <http://etri.re.kr/URC/CAMUS#>
```

상황 모델에 명시적으로 표현되지 않은 암묵적인 지식을 추론하기 위해서는 OWL 추론 규칙 (즉, DL 추론 규칙)뿐만 아니라 응용에 특화된 사용자 정의 추론 규칙을 적용해야 한다. 예를 들어, 회의가 진행 중인 사무실에 사람이 들어오는 경우, 응용에 그 사람이 해당 회의의 참석자라는 지식을 제공하기 위해, 사람을 표현하는 인스턴스와 회의를 표현하는 인스턴스 간에 관계 (예, *isAttendantOf* 속성)를 생성해 주어야 한다. 그런데, '회의 참석자'와 같이 인스턴스 간의 관계를 바탕으로 추론해야 하는 지식은 OWL 언어를 이용해서는 명시적으로 표현할 수 없기 때문이다. 따라서, 논문에서 제안하는 상황 지식 모델링 방법의 효용성은 상황 모델이 지식 추론 규칙을 작성하고, 작성된 추론 규칙이 적용될 때의 효과를 예측하여 원하지 않는 동작이 일어나지 않는지 등을 쉽고 효과적으로 확인 할 수 있는지를 검증함으로써 확인할 수 있다.

상황 모델이 제안된 상황 지식 모델링 방법에 의해 작성되면, 위에서 예를 통해 설명한 것처럼, 사용자 정의 추론 규칙에 의해 동적으로 생성되거나 수정되어야 하는 관계 (즉, 속성)는 *Attendant* 클래스나 *Presenter* 클래스의 필요충분한 멤버쉽 조건으로 정의된 *isAttendantOf* 속성이나 *isPresenterOf* 속성 같이 식별조건에 해당되지 않는 속성으로 국한된다. 또한, 이러한 속성이 실제로는 *Person* 클래스와 같은 *TYPE* 종류의 클래스에 정의된다. 결과적으로, 동적으로 관리해야 하는 속성이 여기저기 흩어져 정의되지 않고 하나의 클래스에 모아져 정의된다. 즉, 동적인 속성을 생성할 때 해당 속성을 생성해 주어야 하는 곳과 방법이, 동적인 속성을 참조하여 추론할 때에 해당 속성을 찾아야 하는 곳과 방법이 명확해 진다.

다음 코드는 위에서 언급한 예를 위해 Jena 추론 규칙 정의 언어를 이용하여 작성된 사용자 정의 추론 규칙이다. 아래 예를 통해 알 수 있듯이 추론 규칙이 매우 간단해진다. 만약 동적으로 관리되어야 하는 속성이 여러 클래스 (예: *Attendant* 클래스, *Presenter* 클래스 등)에 흩어져 있다면 추론 규칙이 각 클래스 별로 생성된 인스턴스를 일일이 찾고 각각의 인스턴스 별로 해당 속성을 생성해 주어야 할 것이다. 이러한 경우 추론 규칙에서 처리해 주어야 하는 인스턴스와 속성 중 일부를 누락하여 결과적으로 상황 지식 베이스가 모순 (inconsistent) 상태에 빠질 위험성이 높아질 것이다.

```
@prefix  camus: <http://etri.re.kr/URC/CAMUS#>
@prefix  oe: <http://etri.re.kr/URC/OEProject#>

@include
<file:D:/ContextManager/RULE/oep_rule.ckr>

[meeting_attendant:
  (?meeting   rdf:type           oe:Meeting)
  (?meeting   oe:hasVenue        ?room)
  (?room      rdf:type           camus:Room)
  (?person    camus:hasLocation  ?room)
  (?person    rdf:type          camus:Person)
 ->
  (?person    oe:isAttendantOf   ?meeting)]
```

5. 결 론

유비쿼터스 서비스 로봇을 위한 미들웨어 시스템 (CAMUS) 개발 경험으로부터 상황 지식의 공유와 재사용이 이론적인 측면에서 뿐만 아니라 상황 기반 응용을 개발하는 것과 같은 실제적인 측면에서도 매우 중요함을 알 수 있었다. 논문에서는 상황 지식의 공유와 재사용을 위해 Guarino의 상위 온톨로지 이론에 기반하여 새로운 상황 지식 모델링 방법을 제안하였다. 제안된 방법은 1) 어떤 기준으로 상황 지식을 모듈화하고 계층화할지, 2) 상황 지식 모듈에 어떤 상황 지식 요소 (클래스, 속성, 인스턴스)를 식별하고, 이들을 어떻게 구조화할지에 대한 원리와 가이드를 제공한다.

프로토타입 구현을 통해 상황 모델이 제안된 상황 지식 모델링 방법에 따라 작성되면 상황 모델의 구조와 표현된 상황 지식 요소의 의미가 일관되고 명확해 져 상황 기반 응용이 상황 지식을 쉽고 오류 없이 공유하고 재사용할 수 있음을 확인하였다. 프로토타입 구현을 통한 검증은 실제 검색문과 사용자 정의 추론 규칙의 적용 예를 통해 필요한 상황 지식을 어디에서 어떻게 찾아야 하는지, 변화된 상황을 어디에 어떻게 반영해야 하는지가 명확해 지는 것을 보임으로써 이루어 졌다. 제안된 상황 지식 모델링 방법을 따르면 또한, 상황 지식 요소의 생성 또는 변경이 어떤 영향을 미치는 지를 쉽게 파악할 수 있어 상황 지식 베이스의 모순 상태를 쉽게 회피할 수 있으며, 검색문 또는 사용자 정의 추론 규칙의 디버깅이 쉬워진다.

참고문헌

[1] A.K. Dey, Gregory D. Abowd, and Daniel Salber. "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications." Human-Computer Interaction, 16(2), 2001.

[2] T. Kindberg, and J. Barton. “A Web-based nomadic computing system.” Computer Networks (Amsterdam, Netherlands: 1999), 35(4), 2001.

[3] K. Henricksen, J. Indulska, and A. Rakotonirainy. “Modeling context information in pervasive computing systems.” Proceedings of the First International Conference on Pervasive Computing, volume 2414 of Lecture Notes in Computer Science, Zurich, August 2002.

[4] Anand Ranganathan, and Roy H. Campbell. “A middleware for context-aware agents in ubiquitous computing environments.” ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, 2003.

[5] Herry Chen, Tim Finin, and Anupam Joshi. “An ontology for context-aware pervasive computing environments.” The Knowledge Engineering Review, 18(3), 2003.

[6] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. “Ontology based context modeling and reasoning using OWL.” Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.

[7] DAML language specification web site: http://www.daml.org/2001/03/daml+oil-index.html

[8] 김학래, 김흥기. “유비쿼터스 서비스(Ubiquitous Services)를 위한 시맨틱 웹 기술(Semantic Web Technology).” 한국경영정보학회 추계학술대회 논문집, 2003.

[9] 김병만, 이경, 박인용, 김시관. “유비쿼터스 컴퓨팅과 사용자 모델링.” 정보처리학회지 제10권 제4호, 2003.7.

[10] Riichiro Mizoguchi. “A step towards ontological engineering.” Proceedings of The 12th National Conference on AI of JSAI, 1998.

[11] Nicola Guarino. “Formal ontology, conceptual analysis and knowledge representation.” International Journal of Human-Computer Studies, 1995, 43(5-6).

[12] Common Object Request Broker Architecture: Core Specification, Object Management Group Inc. March 2004.

[13] Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Sean Bechhofer, “OWL Web Ontology Language Reference.” World Wide Web Consortium (W3C), February 2004.

[14] Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/, Hewlett-Packard Development Company, LP

[15] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. The Description Logic Handbook. Cambridge University Press, 2003.

[16] I. Hrrocks, U. Sattler, and S. Tobies. “Practical reasoning for expressive description logics.” Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR’99), number 1705 in Lecture Notes in Artifical Intelligence, Springer-Verlag, 1999.

[17] R.M. Volker Haarslev. “Description of the racer system and its applications.” Proceedings of International Workshop on Description Logics (DL-2001), 2001.

[18] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. “The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications.” Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, 2004.

[19] 조준면, 한순흥, 김 현, “부품 라이브러리 정보 통합을 위한 온톨로지의 비교 가능성과 균질성 확보”, 한국정보처리학회 논문지, 12-D(3), 2005.

[20] 조준면, 한순흥, 김 현, “상위 온톨로지를 이용한 부품 라이브러리의 정보 통합”, 한국전자거래학회 논문지, 10(3), 2005.

[21] Nicola Guarino, and Christopher Welty. “A formal ontology of properties.” Proceedings of 12th Int. Conf. on Knowledge Engineering and Knowledge Management, Lecture Notes in Computer Science, Springer Verlag, 2000.

조 준 면

1993 한국과학기술원 기계공학과(공학사)
1995 한국과학기술원 기계공학과(공학석사)
2006 한국과학기술원 기계공학과(공학박사)

현재 한국전자통신연구원선임 연구원
관심분야 : Ontological Engineering, Intelligent System, Context-Awareness

김 현

1984 한양대학교 기계설계학과(공학사)
1987 한양대학교 기계설계학과(공학석사)
1997 한양대학교 기계설계학과(공학박사)

1998 – 1999 한양대학교 산업공학과 겸임교수
현재 한국전자통신연구원지능형로봇서버연구팀장 , 책임연구원
관심분야 : Intelligent System, Distributed Computing, Context-Awareness, Engineering Knowledge Management

한 순 흥

1990 미국 Michigan대(공학박사)
1979 – 1992 해사기술연구소 (현 해양연구원)
2003 전자거래학회(www.calsec.or.kr) 회장

2001 – 2003 한국스텝센터(www.kstep.or.kr) 회장
2001 – 2003 International Journal of CAD/CAM (www.ijcc.org) 편집장
1993 – 현재 한국과학기술원 기계공학과 교수
관심분야 : CAD모델표준, STEP, Intelligent CAD, VR응용