DOI: 10.3745/KIPSTD.2009.16D.6.889

컨테이너 안전 수송을 위한 센서데이터 기반의 상황 모델링과 상황인식 미들웨어 구현

남 태 우[†] · 염 근 혁[††]

요 약

현재 컨테이너 물동량은 지속적으로 증가하고 있는 추세이며 이에 따라 수송 원가를 절감하기 위해 초대형 컨테이너선들이 늘어나서 컨테이너 운송이 대형화되어지고 있다. 그러나 항만의 자원은 한정되어 있어서 효율적이고 안전하게 운송하기 위해서는 컨테이너에서 수집된 센서 데이터를 기반으로 환경과 상황에 따른 정보들을 활용하여 지능적이고 적응적인 서비스를 제공하는 방법이 필요하게 되었다.

본 논문에서는 이러한 문제를 해결하기 위해서 상황인식 기법을 적용한 컨테이너 상황인식 미들웨어를 제시하였다. 먼저 사용자, 사물과 동작, 서비스를 포함하는 모든 것을 통칭 적으로 컨텍스트라는 범주로 포함시키고 이 정보를 바탕으로 서비스의 제공이 필요한 상황마다 활용하기 위해서는 컨테이너 도메인의 상황적 요소를 분석해서 컨테이너가 처할 수 있는 특정 상황들이 어떠한 요소들의 조합으로 이루어지는지 정의하고 그 요소들 사이에는 어떤 관계성이 있는지에 대한 컨테이너 상황 온톨로지 모델링을 하였다. 온톨로지 모델을 바탕으로 컨테이너 및 환경에 대한 컨텍스트를 온톨로지로 구축하고 추론을 위한 규칙을 기반으로 상황인식적인 서비스를 추론하는 미들웨어의 개발 및 우선순위 등 정책적인 상황 관리를 가능하게 하는 기능을 구현하였다.

키워드 : 상황인식 미들웨어, 온톨로지 모델링, 상황 추론

Situation Modeling and Situation Awareness Middleware Development with Sensor Data for the Safety Transportation of Containers

Taewoo Nam[†] · Keunhyuk Yeom[††]

ABSTRACT

Currently, the amount of containers is growing up continuously. However, they still depend on humans checking or validating. In this situation, humans must participate in the systems. Situation-awareness technologies allow human participation to be minimized. The situation-awareness technologies provide data or service appropriate to the object's current state, and the state is recognized automatically by the system. They can automatically execute working processes without human intervention.

We suggest a middleware system based on situation-awareness technologies using an ontology for the safety transportation of containers. The middleware is aware of the current state and sends the data required by an application. It can help build new applications by enhancing agility, and it can also support automated service processing by invoking the service required like fire-fighting, fixing container, and so on. Improving transportation of containers process automation with situation-awareness technologies reduces manpower requirements for checking and validating containers, and so saves logistics costs of container transportation.

Keywords : Situation-Aware Middleware, Ontology Modeling, Situation Inference

1. 서 론

현재 컨테이너 물동량은 전 세계 무역자유화 추세의 가속화 및 중국 경제의 급부상 등으로 인해 지속적으로 증가하고 있는 추세이며 수송 원가를 절감하기 위해 초대형 컨테이너선들이 늘어나서 컨테이너 운송이 대형화되고 있다. 그러나 컨테이너를 위한 항만의 관리 자원은 한정되어 있기 때문에 늘어나는 물동량에 맞춰 컨테이너를 보다 효율적이고 안전하게 운송하기 위해서는 컨테이너에서 수집된 센서 데이터를 기반으로 환경과 상황에 따른 정보들을 활용하여 지능적이고 적응적인 서비스를 제공하는 방법이 필요해졌

※ 이 논문은 2009년 교육과학기술부로부터 지원받아 수행된 연구임(지역거점연구단육성사업/차세대물류IT기술연구사업단).
† 준 회 원 : 부산대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수(교신저자)
논문접수 : 2009년 8월 31일
수 정 일 : 1차 2009년 10월 15일
심사완료 : 2009년 10월 23일

다. 즉, 유비쿼터스 상에서의 상황인식 기법이 필요하게 된 것이다.

이러한 지능적 환경과 서비스의 구현을 위해서는 이동하는 객체로부터 다양한 컨텍스트(context)를 수집하고, 주어진 상황에 맞는 최적의 서비스 제공을 위한 추론 기능이 필요하다. 상황인식 기법을 지원하기 위해서는 컨텍스트에 대한 수집, 관리, 추론, 적용이 가능한 상황인식 기반 지능형 미들웨어 기술이 필수적이다. 또한 이러한 기술의 적용을 위해 컨텍스트에 대한 체계적 관리를 가능하게 하는 온톨로지(ontology) 모델링 기술과 이를 기반으로 맞춤형 서비스의 제공이 가능토록 하는 추론엔진 기술의 활용이 중요 사안으로 대두되고 있다. 보다 확장되고 적합한 서비스의 제공을 위해서는 무엇보다 추론 기술이 명료, 정확해져야 하며, 최적화가 필요하다. 환경 정보를 포함한 컨텍스트를 바탕으로 컨테이너의 위치와 상황을 파악하고 컨테이너에 가장 효율적인 서비스의 제공은 물론, 서비스의 관리가 이루어져야 한다. 그러나 컴퓨터는 사람처럼 생각할 수 있는 존재가 아니기 때문에 이러한 사용자의 요구를 만족시킬 수 없는 것이다. 그래서 컴퓨터에게 사용자의 다양한 정보와 환경 정보를 저장, 관리하게 하며, 이 정보를 이용하여 사용자에게 맞는 서비스를 제공할 수 있도록 구현해야하며 이러한 목적을 위해 컴퓨터에게 제공해주는 컨테이너 및 환경의 정보인 상황 모델과 이 정보들을 가지고 서비스를 도출해 내도록 하는 추론 기술이 필요하다.

따라서 본 논문에서는 상황 모델링 방법을 제시하고 컨테이너 및 환경에 대한 컨텍스트를 온톨로지 모델로 구축하였으며 상황 추론을 위한 룰(rule)을 도출하였다. 이를 기반으로 상황인식적인 서비스를 추론하는 미들웨어를 설계 및 구현하였고 평가하려 한다.

2. 관련 연구

2.1 상황인식 미들웨어 선행 연구

유비쿼터스 미들웨어는 2000년도 중반까지 MIT의 Oxygen[1], CMU의 Aura[2], HP의 Cool Town[3], 마이크로소프트의 Easy Living[4] 프로젝트와 같이 상황인식 기반의 유비쿼터스 사회를 가상한 다양한 연구가 진행되었고 국내에서도 유비쿼터스 컴퓨팅 사업단에서 장기과제[5]로 추진되어 오고 있다. 유비쿼터스 미들웨어의 큰 특징인 컨텍스트, QoS, 이동성에 따라 다양한 제품이 나와 있으며 이중에서 컨텍스트 기반의 상황인식 미들웨어로서 대표적인 것으로는 Gaia, SOCAM, CoBra, 그리고 RCSM을 꼽을 수 있다.

일리노이 대학에서 개발한 Gaia[6]는 인간이 생활하는 물리적인 공간이 인간과 상호작용하고 인간은 그러한 물리적인 공간에 필요한 정보를 요청하고 이용 가능한 다양한 리소스를 통해 원하는 작업을 쉽게 수행할 수 있는 공간인 액티브 공간을 실현하는 것을 목표로 하고 있다.

Gaia application은 component기반의 분산되어지고 이동성이 고려된 환경에서의 서비스에 대한 관리가 가능하도록 되어있다. 상황인식 서비스의 구조로 응용이 다양한 context 정보를 수집하고 이를 기반으로 추론이 가능하도록 구현되었다.

싱가포르 국립대에서는 OWL(Ontology Markup Language) [7] 기반의 SOCAM (Service-Oriented Context-Aware Middleware)을 발표하였다. OWL 기반의 온톨로지를 사용함으로써 의미기반 사용함으로에 대한 표현과 다양한 형태의 사용함으로에 대한 추론, 지식의 공유, 용함으로의 분류기반상호의존성과 관련된 것을 해소하고자 하는 것을 목적으로 한다. 또한 로에 대 내에서 룰을 기술함으로써 1차적 컨텍스트에 대한 추론이 가능하다[8].

애리조나 주립대에서는 분산 환경을 지원하는 RCSM (Reconfigurable Context-Sensitive Middlware)을 개발하였다[9]. 제안된 미들웨어는 장치를 동적으로 발견하고 적응형 객체 컨테이너를 통해서 컨텍스트를 수집, 분석, 그리고 컨텍스트의 변화를 감지한다. 또한 등록된 조건이 만족된 Ad-Hoc 네트워크를 형성하고 맥락을 객체에 전달할 수 있도록 한다.

UMBC(University of Maryland, Baltimore Country)에서 2003년에 개발한 CoBra(Context Broker Architecture)[10]는 에이전트 기반 지능형 공간에서 상황인식을 제공하기 위한 구조를 가지고 있다. CoBra 시스템에서 지능형 공간은 물리적 공간으로 지능형 시스템이 공간에 스며들어 사용자에게 서비스를 제공한다. 지능형 상황 브로커는 에이전트 커뮤니티 사이에서 상황 모델을 공유, 관리 한다. 각 에이전트는 사용자가 휴대하는 모바일 기기에서 동작하는 응용일 수 있고, 각 상태 정보를 웹상에서 제공하는 웹 서비스일 수도 있다. 브로커는 상황지식 베이스, 상황 추론 엔진, 상황 획득 모듈, 보안 관리 모듈로 구성되어 있다.

이들 연구는 각각의 응용 서비스에 따라 서로 다른 의미와 형태를 갖는 상황정보를 사용하는 문제가 있고 사용자 공간과 시간에 따른 정적인 패턴화 된 서비스만 제공이 가능하다.

2.2 CONON (CONtext ONtology)

CONON는 컨텍스트를 모델링하기 위해 고안된 온톨로지 모델링 기법이다[11]. 컨텍스트 정보는 상위 계층의 일반화된(generalized) 온톨로지 정보와 하위 계층의 도메인별 특성에 맞게 상위 온톨로지를 상속하여 별도로 정의되는 하위 도메인 온톨로지 정보로 구성되며, 이들은 웹 온톨로지 언어인 OWL로 표현 된다.

상위 온톨로지 계층은 물리적 혹은 개념적인 객체를 나타내는 추상화된 엔티티(entity)들로 구성된다. 모든 온톨로지는 ContextEntity 클래스(class)에서 상속받아서 만들어지며 최상위 클래스인 ContextEntity 클래스에서 상속받은 Person, Activity, Computational Entity (CompEntity) 그리고 Location 클래스가 존재한다. 이 네 개의 클래스는 추상화된 서브클래스(sub-class)들을 가지며 각 엔티티는 고유의 속성(attribute)을 가지고 다른 엔티티들과 연관(relation) 관계를

맺는다. 도메인에 특화된 온톨로지 계층은 상위 온톨로지 계층의 클래스들로부터 상속을 받아 특정 환경에 맞는 온톨로지 구조를 가지게 된다[12]. 이런 구조를 통해서 CONON 모델은 도메인 온톨로지를 쉽게 구성할 수 있으며, 상황인식 어플리케이션 제작에 시간을 절약할 수 있는 장점이 있다.

2.3 JESS (Java Expert System Shell)

Jess는 산디아 국립 연구소의 Ernest Fiedman-Hill에 의해 개발되었다[13]. 모든 부분이 자바(Java) 언어로 개발되어서 자바언어가 갖는 이점을 모두 가지면서 표현 스타일은 LISP언어를 따르는 형태를 취하고 있다.

Jess는 룰 기반의 추론 시스템으로서 프로그램 자체의 크기가 작고 가벼우며, 규칙(rule)과 사실(fact)의 매칭에는 RETE 알고리즘을 이용하여 빠른 추론 기능을 제공한다[14]. 또한 Jess는 자바의 스크립트적인 요소를 십분 활용하여 자바 오브젝트의 생성이나 메서드 호출을 컴파일 없이 실행시간에 수행할 수 있다. Jess는 제공되는 그대로 콘솔 상에서 명령행(command-line) 기반으로 사용될 수도 있고, 애플릿 환경 또는 GUI 환경에서 사용할 수 있다. 또한 다른 프로그램에 포함 (embedded)된 형태 등 다양한 용도로 사용될 수 있다.

Jess가 가진 단점은 RETE 알고리즘을 사용하게 되면서 발생하는 문제이다. RETE 알고리즘은 계산할 것들을 단 한 번만 수행하여 다시 계산되는 시간의 비용을 줄이며 이미 존재 하는 중간 계산들은 저장되어 필요한 경우 재사용한다. 하지만 사실과 룰의 LHS를 매칭할 때는 중간 크기 정도의 가능성들을 반복적으로 탐색한다[15]. 이것은 추론기능의 속도를 높이는 것에 반하여 실행 시 메모리를 많이 사용하게 되는 문제를 발생한다. 하지만 대부분의 경우 적절한 메모리 사이즈 안에서 수행되게 되므로 큰 무리 없이 사용할 수 있고 필요한 경우 자바의 가상 머신의 힙(heap) 사이즈를 늘려서 수행하게 되면 문제를 해결할 수 있다.

3. 컨테이너 상황 모델

컨텍스트 데이터를 기계가 처리할 수 있는 형태로 정의하고 저장하기 위해서는 컨텍스트 모델이 필요하다. 컨텍스트 정보의 표현과 교환방식에 따른 데이터 구조를 기반으로 하여 Strang과 Linnhoff-Popien에 의해 정리된 컨텍스트 모델링의 기법들로는 key-value 기반 모델, 마크 업 기반 모델, 그래픽 기반 모델, 객체지향 기반 모델, 로직 기반 모델, 온톨로지 기반의 모델 방법이 있다[16]. 이들 중 온톨로지 기반 모델링 방식은 단순성, 유연성, 확장성, 일반성, 표현력에 있어 우월함을 나타낸다. 따라서 본 논문에서는 정보를 구조화 하는데 매우 호의적이며 상호 관계성 및 부분적인 상황의 정보를 쉽게 표현할 수 있는 온톨로지 기반의 모델 방법을 사용하여 상황을 모델링한다. 온톨로지는 서술논리 기반의 온톨로지 표현 언어인 OWL을 사용하며 룰은 규칙 표현 언어인 SWRL[17](Semantic Web Rule Language)을 사용하여 구현하였다.

(그림 1) 상황 모델링 과정

상황에 대한 모델링과 이를 바탕으로 한 추론 룰을 정의하기 위해서 (그림 1)과 같은 과정을 제시한다.

3.1 상황인식 미들웨어에서 상황 정의

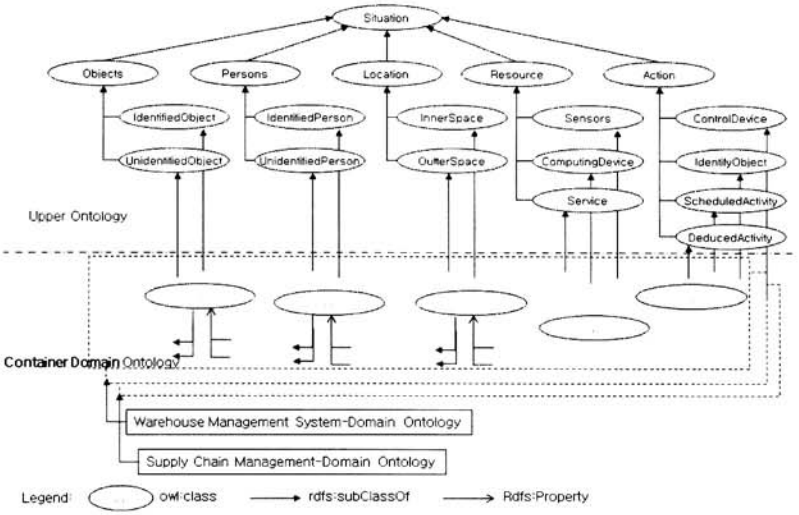
상황인식이라는 용어를 처음 사용한 Schilit와 Theimer는 "사람과 그 사람에 인접한 사물의 Location, Identity 그리고 그 사물의 변화"를 상황으로 정의하였다[18]. 그 이후로 상황에 대한 다양한 정의가 제시되었으나 가장 대표적으로 사용되는 Dey와 Abowd의 정의에 따르면 상황이란 "사용자와 애플리케이션 사이의 상호 작용에 고려되어야 할 사람, 장소, 객체의 상태를 나타내는데 활용되는 모든 정보"로 규정하고 있다[19].

상황에 대한 정보를 나타내기 위한 정보를 크게 사람(Persons), 객체(Objects), 장소(Location), 자원(Resource), 작동(Action) 등의 다섯 가지 형태로 정의하였다. 이 요소는 도메인에 따라 변경되지 않는 공통적인 정보로써 상황 모델에서 상위 계층을 구성하게 된다.

3.2 상황 모델링을 위한 스키마 정의

온톨로지는 관심 있는 도메인에 대한 지식을 표현하기 위해 사용되고 도메인상의 개념(concept) 및 개념들 사이의 관계(relation)를 표현한다[20]. 컨테이너 및 기기, 환경을 구성하는 다양한 것들을 온톨로지화된 표현을 통해 컴퓨터가 이해할 수 있는 언어로 그 관계까지도 표현하고 이러한 관계성에 대한 추론을 기반으로 해야 상황에 따른 맞춤형 서비스의 제공이 가능하다. 또한, 유비쿼터스 환경에서는 모든 객체들과 객체의 속성들을 특정 종류의 도메인 안에 계층적으로 표현해야 하고, 상황에 따라서 속성 값들을 변화시키는 규칙이 필요하므로 온톨로지를 사용하여 유비쿼터스 환경내의 모든 객체들을 계층적으로 구성, 관리해야 한다.

본 논문에서는 이를 위한 온톨로지 구조의 스키마로 CONON 모델의 2계층 구조를 반영하여 (그림 2)와 같은 형태로 구현하였다. 상황 모델링 측면에서의 컨텍스트에 대해서 외부의 상황을 인지하여 반영하기 위한 개체로 정의하고 이는 모든 도메인에 공통적으로 쓰이는 요소로써 모든 온톨로지는 상황(Situation) 클래스에서 상속받아 만들어지며 상위 온톨로지에는 공통적인 요소로 객체(Objects), 사람(Persons), 장소(Location), 자원(Resource), 작동(Action)을 두었다. 이 다섯 개의 클래스는 추상화된 서브클래스들을 가지며 각 엔티티는 고유의 속성을 가지고 다른 엔티티들과 연관 관계를 맺으며 특정 도메인에 특화된 온톨로지로 상속된다.



(그림 2) 상황인식 미들웨어용 온톨로지 모델 스키마

3.3 도메인 상황 분석 및 분류

상황 인식 시스템의 감시 대상의 특정 상황과 그것의 상태를 나타내는데 활용되는 모든 정보를 수집하여 분류한다. 컨테이너 도메인에 특화된 상황은 <표 1>과 같이 분석하였고 각각의 상황을 판단할 수 있는 기능들을 명시하고 분류하였다.

〈표 1〉 컨테이너 도메인 상황 분류

상황 분류	상황에 따른 기능
· 화재 감지	· 화재 발생 전 (평상시) - 온도 감지 - 조도 감지 - 습도 감지 · 화재 발생 후 - 경보음 - 카메라 - 메시지 전송 (소방관)
· 파손 감지	· 파손 발생 전 (평상시) - 충격 감지 - 위치 감지 - 스케줄 정보 (크레인 작업) · 파손 발생 후 - 메시지 전송 (수리 엔지니어)
· 냉각 장치	· 컨테이너 타입 정보 (냉동컨테이너) · 적정 온도 정보 · 위치 정보(냉동장치장) · 이상 발생 전 (평상시) - 온도 감지 · 이상 발생 후 - 메시지 전송 (전기 엔지니어)
· 도난 감지	· 침입 전 (평상시) - 문개폐 감지 - 충격 감지 - 조도 감지 - 위치 감지 - 스케줄 정보 (통관 작업) · 침입 후 - 경보음 - 카메라 - 메시지 전송(경찰)
...	...

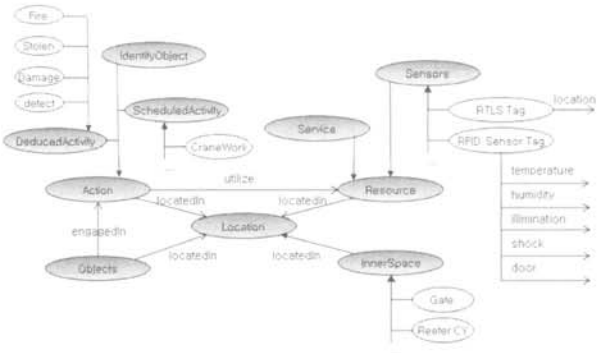
이를 바탕으로 서로 연관이 있는 클래스 간의 상관관계에 따른 속성을 <표 2>에서와 같이 계층적으로 컨테이너 온톨로지 속성으로 분류한다.

컨테이너 도메인에 특화 시킨 계층적 온톨로지 모델은 Objects에는 각종 컨테이너, Resource에는 센싱 데이터로 정의한 6가지 센서와 처리 디바이스, Location은 야드 내의 위치들, Person에는 작업자, 엔지니어 등 서비스와 관련된 사람들, Action은 컨테이너의 상태 감지 행위 및 필요로 하는 서비스를 도메인 특화된 레이어에 작성하였다.

이것은 추후에 여러 도메인에 적용할 수 있게 확장성을 고려한 것이며 도메인에 특화된 부분을 결합한 컨테이너 온톨로지 상황 모델은 (그림 3)과 같다. 상위 온톨로지 클래스들은 각각 컨테이너에서 사용될 수 있는 센서인 RFID 센서 태그와 RTLS 태그, 컨테이너 야드 내의 위치, 컨테이너에 발생할 수 있는 상황 등 컨테이너 도메인에 특화된 온톨로지 클래스들로 확장하였다.

〈표 2〉 상위 온톨로지에 따른 계층적 상황 분류

Service	Persons	Location	Resource	Action
· 화재	· 소방관	· 야드내	· 온도 센서 · 조도 센서 · 습도 센서 · 경보기 · 카메라	· 메시지 전송 (소방관)
· 파손	· 수리 엔지니어	· 야드내	· 충격 센서 · 위치 센서 · 스케줄 정보	· 메시지 전송 (수리엔지니어)
· 온도	· 전기 엔지니어	· 냉동 장치장	· 온도 센서	· 메시지 전송 (전기엔지니어)
· 도난	· 침입자 · 경찰	· 야드내	· 문개폐 센서 · 충격 센서 · 조도 센서 · 경보음 · 카메라	· 메시지 전송 (경찰)
...	...	...	...	...



(그림 3) 부분적인 컨테이너 도메인 온톨로지 상황 모델

이 온톨로지 상황 모델은 컨테이너, 환경, 기기, 상황, 시간, 서비스 등에 대한 정보를 내포하고 있으며 이를 통해 확장성 및 유연성을 증대시킬 수 있는 것이다. 하나의 커다란 클래스로 구성하지 않고 특성에 따라 여러 개의 클래스를 구성하고 이에 대한 관계성을 중심으로 전개한 것은 이후 다양한 특성이 추가되어지거나 새로운 환경이나 연동성 측면에서 클래스의 변화가 있을 때 추가 및 삭제가 용이하도록 구현한 것이다.

3.4 도메인 상황 모델 구현

본 논문에서는 온톨로지 개발 도구인 Protégé[21]를 이용하여 OWL로 컨테이너 도메인에 대한 온톨로지를 구현하였다. 온톨로지를 구축하는 과정은 클래스를 표현하는 과정, 클래스 간의 상관관계에 따른 속성을 표현하는 과정, 속성 간의 제약사항

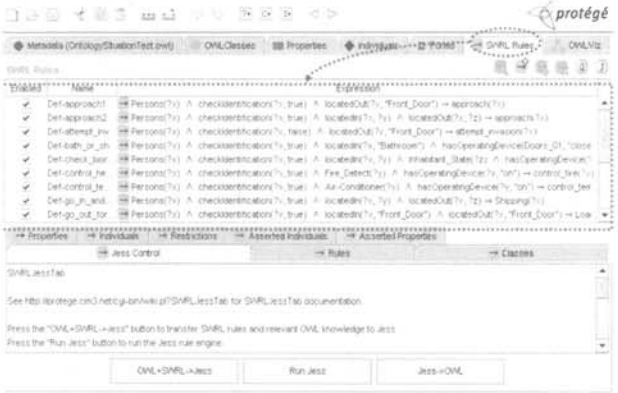
을 표현하는 과정, 인스턴스(instance) 값을 입력하는 과정을 거친다.

추론엔진은 구현된 온톨로지의 인스턴스화된 센서데이터를 바탕으로 실제 추상화된 상황 정보를 생성하게 되고 이는 향후 추론 규칙의 Fact로써 사용된다. 클래스를 표현하는 과정에서는 컨테이너 도메인 온톨로지 상황 모델을 바탕으로 클래스와 클래스간의 계층관계를 포함하여 클래스를 표현한다. 계층관계에 대한 속성이 정의 되고나면 표 2와같이 서로 다른 상위 온톨로지를 상속 받는 클래스들 간의 상황 분류를 바탕으로 서로 연관이 있는 클래스 간의 상관관계에 따른 속성을 컨테이너 온톨로지 속성으로 표현한다. 속성을 표현하는데 있어서 Object Property는 속성의 값이 클래스나 인스턴스일 때, Datatype Property는 속성의 값이 XML 스키마 형식이나 문자형일 때 사용한다.

3.5 상황 추론 룰 정의

속성 간의 제약사항을 표현하는 과정에서는 각 온톨로지 속성 간의 제약사항과 추가되는 관계를 작성하였다. 외부 상황과 상호 작용하는 객체의 의미 있는 행위인 Action에 대해서 SWRL을 이용하여 (그림 4)와 같이 작성하였다.

SWRL로 작성된 규칙들은 Jess 추론엔진에서 새로운



(그림 4) 컨테이너 온톨로지에 대한 제약사항 표현(SWRL)

〈표 3〉 룰 표현

```
Object(?x)^IdentifiedObject(?y)^hasResource(?y,?z)^
swrlb:equal(?z,"freezer")^abstractTemperature(?a,?arg1)^
swrlb.greaterThan(?arg1,"cold")^dataBoolean(?a,?arg1)^
swrlb:notEqual(?arg1, true)
→ abstract(?x, "freezer Problem")
```

Fact로 저장되어 이후 실시간으로 수집되는 센서들의 정보를 바탕으로 컨테이너가 현재 어떤 상태에 있는가를 판단하는 기준이 된다. 예를 들어 <표 3>은 냉동컨테이너 냉각장치 고장의 상황 정보를 생성하는 SWRL 추론 규칙이다. "이 컨테이너는 냉동컨테이너이고 내부 온도가 지정된 온도보다 높다." 라는 센서 정보를 통해 냉각장치가 동작해야 될 상황에 실제 냉각장치가 동작하지 않으므로 "냉각장치에 이상이 발생했다."라는 상황 정보를 생성한다.

이러한 형태로 Jess에서 사용할 룰을 구현하게 되는데 룰에서 앞부분의 조건으로 사용자가 특정 시간, 특정 위치, 특정 상황일 때 어떠한 서비스를 제공해야 하며 서비스가 중복되어질 경우에는 어떻게 처리를 하는 지 등을 결정하게 된다. 그리고 이러한 것은 if~then~else의 논리로 적용된다. 컴퓨터의 특성상 모든 일반적인 상황들에 대한 명시가 되어야만 실질적인 추론과 서비스의 제공할 수 있으며, 상황의 이해를 위한 시스템 룰의 구현 역시 필수적이다.

4. 상황인식 미들웨어 구현

4.1 상황인식 미들웨어 아키텍처

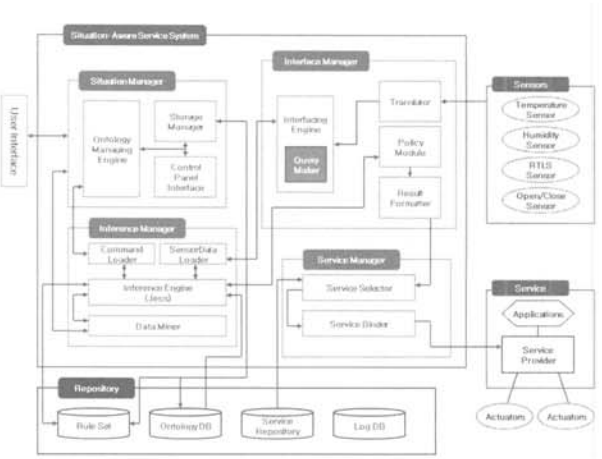
본 논문에서는 자바를 기반으로 함으로써 미들웨어의 다른 구성요소와의 연동성을 고려하여 상황 추론의 구현을 위해 룰을 중심으로 보다 다양하고 확장성이 있으며 룰에 대한 처리가 우수한 Jess를 기반으로 하였다. 또한 룰의 구현에 앞서 온톨로지로부터 기본적 컨텍스트를 적용받아 사용하며, 이러한 컨텍스트와 룰의 매칭을 통해 실질적인 추론이 이루어지도록 한다. 추론 엔진은 Fact, 저장되어진 컨텍스트 정보의 획득을 위해 필요한 값만을 추출하는 질의, 추

론을 위해 필요한 룰 및 질의 등에 대해 우선순위를 결정하거나 필요 룰에 적용을 위한 방안을 구현하는 시스템 룰, 실질적인 추론의 수행을 통해 서비스를 도출하기 위한 수행(processing) 룰 등으로 구성된다. 그리고 본 논문에서 제안하는 추론엔진은 이러한 룰을 기반으로 동작하게 되는 rule-based engine과 컨텍스트 정보에 대한 N-triple 형태로의 관리 및 질의 동작을 수행하며 다양한 정보의 관리를 하게 되는 Inference Manager, 그리고 다른 다양한 외부 센서나 클라이언트로부터 정보 또는 명령을 받거나 결과 값에 대한 전송을 위한 인터페이스(interface) 부분을 구현하여 Jess를 중심으로 구현된 추론 엔진과 자바로 구현된 인터페이스 부분을 연동할 수 있도록 되어 있다. 이런 특성을 바탕으로 본 논문에서 제안하고자 하는 미들웨어는 (그림 5)와 같은 아키텍처를 지닌다.

Inference Manager는 Jess 부분과 그 외부로의 연동이

<표 4> Inference Manager 세부 기능 정의

	기능	세부 구성
1	Situation Modeling 및 Situation Manager와의 연동 및 관련 기능	
1-1	Situation Manager로 부터 제공되는 Ontology Model에 대해 저장 및 관리	· Situation Manager로부터 제공되는 N-triple 형태의 정보 값에 대한 Fact로 저장 · Fact 값에 대한 관리 및 추론에 따른 값의 제공
1-2	Fact에 대한 실시간 관리 및 Update	· 동적으로 변화되는 센서 값에 대한 정보를 Fact에 저장, 센서 값에 따른 location update -> 컨테이너의 위치 tracking 지원
1-3	사용자 Command 처리 및 Inference Engine으로의 적용	· Situation Manager로부터 전송되어진 사용자의 Command에 대한 분석에 따른 Inference Engine으로의 적용
2	Interface Manager와의 연동 및 관련기능	
2-1	SensorData의 분류 및 Inference Engine으로의 적용	· Sensed Data의 분류 및 Update, Inference Engine으로의 적용
3	Inference Manager 내부 처리 관련 기능	
3-1	Ontology와 Rule에 대한 Matching	· Ontology를 통해 추론되어진 값에 대한 관련 Rule의 적용 · Fact 값과 룰 사이에 관련성 추론 및 Fact에 대한 연관성 검사 · 동일한 센서 데이터에 대한 필터링 적용
3-2	추론된 결과에 대한 policy 적용 및 정리	· Priority policy에 대한 Fact 값과 우선 순위 및 중복에 대한 처리 문제의 룰 · Conflict 발생시 우선순위를 통한 차등적 적용을 위한 룰
3-3	Rule에 대한 우선순위 관리 및 적용	· 다양한 룰 자체에 대한 우선순위 관리 기능
4	Service Manager와의 연동 및 관련 기능	
4-1	결과 값에 대한 Service Manager로의 전송	· 룰 및 Fact를 통한 추론된 결과값에 대한 의미적 정리 및 관리 · Service Manager로의 결과값 전송

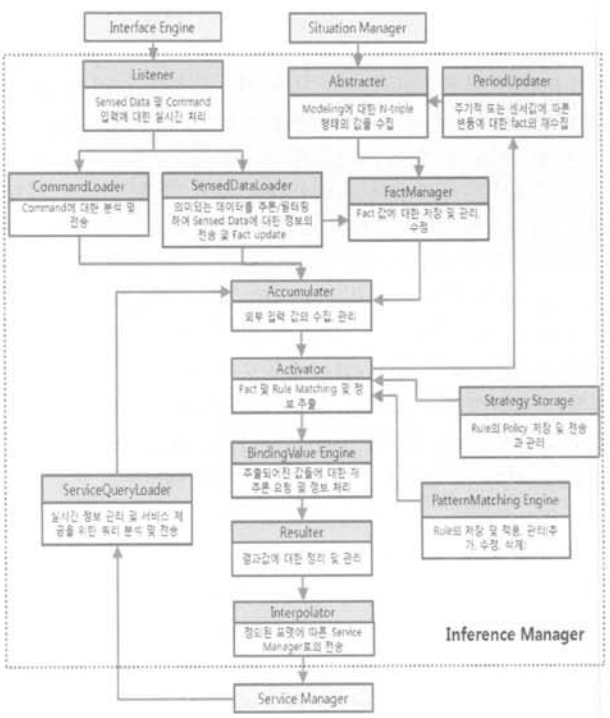


(그림 5) 상황인식 미들웨어 아키텍처

되는 Loader 부분으로 구성되며 또한 자바를 사용한 구현을 통해 외부 다른 모듈 및 매니저와의 연동을 구성한 부분과 Jess를 기반으로 한 룰 및 실질적인 추론 부분으로 나누어진다. Inference Manager 부분과 그 외의 Manager들의 연동에 대한 세부 기능적 정의는 <표 4>와 같다.

이러한 기능적 분류에 있어서 기능 및 적용 특성에 기반을 두어 세부 모듈로 구현이 가능하며, 각 모듈은 (그림 6)과 같은 흐름으로 순차적으로 적용이 된다.

본 논문에서 구현한 상황인식 미들웨어는 다양한 입력으로부터 정보를 전송받는 부분이 존재하며, 입력에 대한 특성, 정보의 추론 성향에 따른 특성에 따라 분류될 수 있다. 그리고 서비스 추론을 위한 기능과 룰, 그리고 Fact로부터 정보의 질의를 처리하는 부분으로 구성된다.



(그림 6) 상황인식 미들웨어 구성 및 흐름도

4.2 온톨로지 및 Fact를 활용하기 위한 질의 구현

상황인식 미들웨어는 룰을 온톨로지 모델 기반에서 Fact로 변환을 하여 working memory에 저장 후 이에 대해 질의를 통하여 필요한 정보를 수집, 추출한다.

질의 처리를 구현하기에 앞서 deftemplate이라는 Jess의 용어를 사용하여 온톨로지의 정보가 subject, predicate, object의 N-triple 형태로 저장되어질 수 있는 저장 공간을 naming을 통해 확보한다. 이렇게 확보된 공간에 loading되는 Fact를 저장함으로써 정보의 체계적인 관리가 가능하며 저장할 때는 기본적인 환경 및 사용자 정보 뿐 아니라 관계성도 저장된다. 또한 defquery라는 명령어를 사용하여 Fact에 저장되어 있는 정보, 즉 컨텍스트에서 필요한 것을 추출하는 형태를 지정한다. naming 된 공간 내의 subject나 predicate, object 중 하나 이상의 변수를 기반으로 그 이외의 것을 추출하는 형태이다. 각 naming이 다른 템플릿(template)은 다른 공간의 것으로 보게 되어 각기 다른 형태의 질의를 필요로 한다. 질의는 Fact에서의 세부 값을 추출하는 것 뿐 아니라 공간의 name을 기반으로 그 템플릿의 모든 값을 찾거나 name을 찾는 등 모든 형태의 질의를 구현할 수 있다. 질의는 정보의 추출을 위한 기본적인 틀로써 구성하는 것이 기본이지만, 서비스 또는 상황인식적 추론을 위해 특정 정보를 찾는 것에 적용하기도 한다.

4.3 룰 구현 및 적용을 위한 시스템 룰 구현

룰을 적용하기 위해서는 기본적으로 룰에 대한 적용을 위한 룰, 그리고 우선순위나 적용 방안을 위해 필요로 하는 룰이 존재한다.

defrule이라는 명령어를 통해 이러한 룰은 정의되며, 자바로부터 입력되어지는 정보나 명령에 대해 추론엔진은 sleep 또는 listening 상태로 대기를 하다가 입력이 이루어지면 wake-up 상태로 천이된다. 그러면 입력값을 분석하여 종류에 따라 각각 다음과 같은 스테이트(state) 부분으로 전송되어진다.

· command
- 사용자에 명령에 따라 특정 기기 동작이나 정적 패턴화된 서비스의 구현을 위한 부분

· sensed data
- 외부의 환경의 변화에 따라 입력되어지는 데이터로써, 외부 환경 변화에 기인하여 추가적인 서비스의 동작이나 외부 환경 변화에 대한 모니터링에 대한 동작을 위한 부분

· query
- 사용자가 위치 검색이나 기기 상태 검색, 또는 환경 정보에 대한 검색을 요청하는 경우에는 외부의 질의 API를 통해 생성되어서 정보가 들어오게 되는데 이를 처리하기 위한 부분

4.4 온톨로지 및 Fact를 위한 처리 룰 구현

Fact로부터 질의를 통해 수집되어진 정보에 대해서 룰을 통해 값을 처리함으로써 실질적인 추론이 가능하다. 다음에 나올 서비스의 구현을 위한 처리 룰과는 달리, 함수(function) 형태의 질의와 특정 입력과 저장되어 있는 정보의 비교, 분석을 통한 추론이기 때문에 1차적인 정적 형태의 추론만이 가능하다. 그리고 대부분의 현존하고 있는 추론 엔진들이 이러한 형태를 취하고 있다.

입력된 정보를 기반으로 정보값과 온톨로지로부터 loading된 Fact를 검색하는 동작을 명령하게 되는데, Fact에서 같은 값을 찾으며 그에 따른 관계성에 의존하여 다양한 값들을 출력하게 된다. 이는 N-triple(subject, predicate, object) 값 중에 하나 이상의 정보를 입력하게 되면 이외의 값들을 찾는 질의를 기반으로 하여 그 관계성에 따라 정보를 검색한다. 그리고 관계성 추론은 시스템 룰에서 정한 정책을 따른다. 입력값에 대해 1차적으로 Fact로부터 질의를 중심으로 값을 찾아내는 것을 반복하고 이를 룰에서 결과값을 생성하여 출력하는 것으로 추론을 구현하는 방안이다.

이러한 룰은 모든 서비스 룰에 기반이 되는 것으로 하나의 함수처럼 다른 룰의 기반으로 연동되며, 모든 서비스 룰에 활용되는 일부의 룰은 함수로 바꿔서 모든 룰에서 호출을 통해 활용할 수 있도록 구성하였다.

4.5 지능형 서비스 구현을 위한 처리 룰 구현

실질적인 서비스의 구현을 위해 존재하는 룰로써 함수 형태와 룰 형태가 복합적으로 적용됨으로써 필요한 서비스를 Fact와 입력에 대한 값으로부터 추론하는 것이다. 일반적으로 정적 패턴화되어진 서비스에서 활용되는 것으로 일정한 패턴의 서비스는 추론의 반복적 검증 또는 Fact에 대한 질의를 통하는 작업의 효율성을 위해 기능을 일반화하도록 룰을 구현한다.

두 가지 형태는 서비스의 구현을 위한 것으로, 서비스 매니저로부터 ACK/NACK가 왔을 경우, 즉 추론되어진 서비스에 대해 명령이 하달된 것에 대한 응답이 들어오는 경우, 이에 대한 대처 방안이나 서비스에 대한 정보의 저장, 관리를 위한 룰이 된다. 사용자의 명령에 대한 응답을 위한 서비스 룰로써 정적 패턴 서비스는 그 특정 명령이나 센서 값에 따라 동작을 하게 되며 동적인 경우, 그 상황에 맞는 기본 서비스에 대한 추론이 이루어진다.

4.6 우선순위 처리 및 conflict 해결을 위한 룰 구현

추론엔진은 입력값이 들어오는 경우 서비스를 추론하게 되고 서비스에 대한 conflict가 발생되었을 때 상황 및 서비스 우선순위를 검색한다. 그리고 공간이나 특정 상황에 대한 추론을 병행한다. 이러한 일련의 동작을 통해 서비스를 결정을 하게 된다.

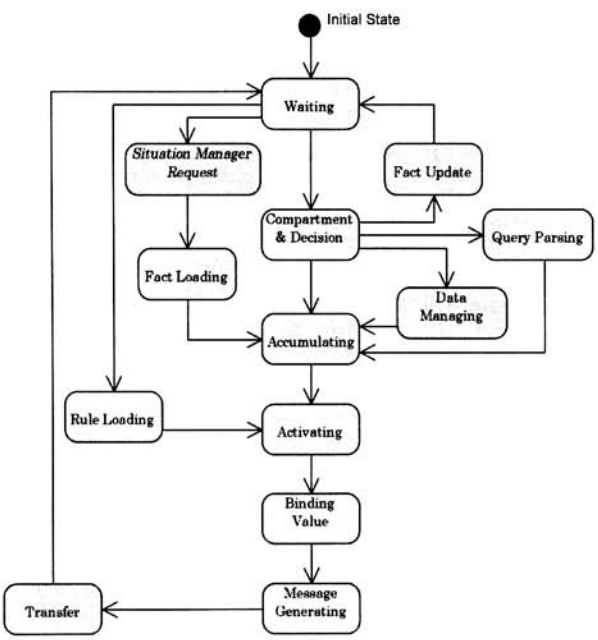
서비스에 대한 충돌 검사는 Fact에 저장되어 있는 서비스가 구현된 정보에 대한 비교, 분석을 통해 conflict를 인식하며 다양한 우선순위에 대한 순차적인 검증 작업을 시행함으

로써 컨테이너 및 상황, 환경에 따른 동적인 서비스가 제공 가능하고 우선순위를 통한 충돌 회피가 가능해 진다.

5 설계 검증 및 테스트 결과

상황인식 미들웨어 설계에 대한 검증에 앞서 룰 기반으로 구현되어진 엔진의 내부 흐름을 통해 추론엔진에 대한 이해를 높이고 실질적인 검증 작업이 선행되어야한다. 설계의 과정에 구현되어진 다이어그램(그림 6)과 Jess 내에서의 흐름도를 통해 각 모듈 간의 시간적 흐름과 전송되는 정보 값에 대한 좀 더 세부적인 이해를 가질 필요가 있다.

(그림 7)은 상태전이도표(State Diagram)를 통해 Jess 내에서 입력과 흐름에 따른 상태의 변화를 표시한 것이다. Jess 내에서는 입력을 대기하는 상태와 입력에 따른 룰과 Fact를 검증하는 내부로의 정보 전송 상태, 그리고 결과 산출에 대한 정보의 외부로의 전송 후 다시 대기 모드로 전환되는 상태가 존재한다.



(그림 7) 추론엔진의 상태전이도표

5.1 제약사항

온톨로지와 룰을 구현하여 실질적인 테스트를 하였고 서비스의 동적 환경에서의 적용 및 정책적인 부분에 중점을 두고 설계를 하였기 때문에 속도적인 측면에서 다른 추론엔진보다 뛰어나지 않을 수도 있다. 하지만 다양한 서비스를 커버하고 룰에 대한 효율성을 최대한 높일 수 있도록 구현하였기 때문에 속도적인 면에서는 다른 추론엔진과 크게 차이가 없거나 뛰어날 것으로 예상된다.

실험에 대한 정확도를 보다 높이기 위해서는 서비스의 개수 및 종류, 동적 환경에서의 적용에 대한 검증이 병행되어야 하지만 현재는 속도적인 측면과 정량적인 평가 지표에 의해서만 평가를 진행하도록 하였다. 또한 Jess의 활용은 자바 기반이며 룰 작성에 대한 작업이 용이하고 룰에 대한 처리가 우수하기 때문에 다른 소프트웨어에 비해 속도적인 면에서 약점을 지니지만 trade-off 측면에서 고려를 해볼 필요가 있다. 현재 구현되어진 OWL 기반의 온톨로지에 대한 사항과 추론엔진에 대한 사항은 <표 5, 6>과 같다.

〈표 5〉 온톨로지 분류표

컨테이너 도메인 온톨로지 모델		
분류	class 개수	Individual
Entity Relative	38	264
Object Relative	24	62
Policy	9	14
Service	8	12
Space	9	17
User Command	5	dynamic

〈표 6〉 추론엔진 분류표

추론 엔진			
분류	Rule	Function	Query
Sensor	5	11	7
Location	4	7	6
Configuration	5	9	6
Command	6	13	6
Service	4	7	5
Policy	4	9	13
총 합	28	56	43

5.2 테스트 환경

Window 시스템 환경에서 구동이 가능하도록 하였으며, 사용 언어는 다른 Manager와의 연동을 위한 자바 인터페이스로 구현하였다. Jess는 룰의 적용이 강력한 것에 반해 메모리의 사용량이 많기 때문에 메모리 크기가 크고, 제한 범위를 넓히는 작업이 수반되어야 한다. 상황인식 미들웨어 설치 환경은 <표 7>과 같다.

〈표 7〉 상황인식 미들웨어 설치 환경

하드웨어 플랫폼	Pentium core 2 duo CPU 2.33GHz , RAM 2GB
운영체제	Windows XP
사용언어	Java jdk1.6, Jess
온톨로지 및 룰 구현	OWL, SWRL

5.3 테스트베드 구축

테스트베드의 구현을 위해 상황인식 미들웨어를 탑재한 서버에 기본적인 센서 값을 통해 추론 결과가 나오는 시간을 측정하였다. 센서는 온도, 습도, 조도, 충격, 문개폐, 위치

(그림 8) 상황인식 미들웨어를 이용한 응용 애플리케이션

등의 여섯 가지 센서를 고려하였는데 테스트는 단시간 내에 여러 가지 조건의 상황 정보를 만들어야 되므로 센서데이터 제너레이터(generator)를 사용하였다.

제너레이터를 통해 전송되어지는 입력을 상황인식 미들웨어가 직접 받아 결과를 도출하여 이를 (그림 8)의 응용 애플리케이션으로 전송한다. 제너레이터로부터 클라이언트까지의 걸리는 시간과 상황인식 미들웨어 내부에서만 소요되는 시간을 체크하였으며 이를 기반으로 다른 추론엔진과 비교하였다.

5.4 테스트 결과 및 평가

본 논문에서 사용한 추론엔진은 기본적으로 컨테이너의 이동이 없는 상황에서 주기적으로 센서값이 전송되어지는 경우 226ms의 반응 시간이 걸린다. 그리고 컨테이너가 위치를 이동함으로써 다양한 기기의 동작이 발생하는 경우(서비스가 동적으로 변화하는 경우)에는 2463ms의 추론 시간이 소요된다. 본 논문의 경우 클래스가 총 88개와 individual은 최소 369개로 구현되어 있으며 이를 Fact의 triple로 변환하는 경우 1700개 이상으로 구성되어 있다. 그 내부에는 정책에 관련된 클래스 및 individual도 포함되어진다. 또한 추론엔진에서도 각각 다양한 정책적인 관리가 추가되어서 질의 수는 43개, 룰은 함수를 포함하여 84개로 구성되어 있다.

컨테이너의 이동에 따른 동적 서비스가 실시간 구현되어지는 경우 2463ms가 소요되어 서비스의 양이 늘어나고, 이동량이 증가할수록 급격하게 대기 시간이 증가되는 것을 알 수 있다. 그러나 이 문제는 실제 상황에서는 컨테이너에서 센싱 정보를 받아드리는 리더들을 효과적으로 배치해서 복수개의 미들웨어에서 분산 처리할 것이므로 테스트 환경에서와 같이 단일 미들웨어에서 극도로 많은 양의 컨테이너를 처리하지는 않을 것이다.

이를 통해 결과로 나온 시간만을 가지고 본 논문의 상황인식 미들웨어의 강점을 표현할 수는 없다. 하지만 위에서도 알 수 있듯이 추론에 있어서 시간적 문제는 큰 차이가 없으나 본 논문에서 제시한 추론엔진은 실질적 컨테이너의 동적 이동에 대한 추론이 가능하며 서비스 정책적 관리를 통해 서비스 충돌에 대한 검증도 어느 정도 이루어졌다.

6. 결론 및 향후 연구 과제

지속적으로 증가되고 있는 컨테이너 물동량을 효율적이고 안전하게 처리하기 위해서는 컨테이너의 현재 상태를 바탕으로 빠른 시간 내에 추론하여 신속하게 필요한 서비스를 제공하는 것이 필요하다. 이를 해결하기 위해서는 컨테이너의 현재 상태를 추론하여 추론된 결과들 간의 우선순위 처리, 적합한 서비스의 호출 등이 실시간으로 처리되어야한다. 본 논문에서 구현한 상황인식 미들웨어는 컨테이너 도메인 분석을 통한 온톨로지 모델링과 컨테이너 상황 모델을 기반으로 룰을 정의하였다. 그리고 이를 바탕으로 실시간으로 움직이는 컨테이너들의 상황을 추론하고 추론 결과의 conflict를 방지하면서 능동적인 서비스를 제공해 주는 상황인식 미들웨어를 구현하였다.

그러나 다중 객체의 동적인 움직임에 대한 다중 서비스를 처리하는 것에는 어려움과 지연시간의 문제가 있었다. 컨테이너가 장소를 이동함을 통해 적용 서비스의 변경이 발생하는 것을 측정하였으며, 추론 결과에 대한 conflict 측정은 이루어졌으나 다중 객체의 다중 서비스 환경에서의 지연시간에 대한 고려가 부족하였다.

향후 연구 과제는 현재의 상황인식 미들웨어를 컨테이너 도메인이 아닌 범용적인 도메인에서 사용 가능한 프레임워크 형태로의 진화이다. 범용적인 도메인에서 보다 세밀한 결과를 도출하고 실제적인 환경으로의 적용을 위해서는 자동적으로 온톨로지와 룰을 수정할 수 있는 방안도 고려되어져야하며 정적 패턴화된 서비스나 환경에 의존하는 일차적인 서비스에 국한되지 않고 추론 결과에 기반을 둔 재추론과 동적 상황에 대해서도 서비스가 구현될 수 있도록 룰과 온톨로지를 확장해야 할 것이다. 이를 위해서 동적 상황의 서비스를 위한 최근 상황에 대한 정보의 저장 및 관리, 삭제가 가능한 상황 정보 저장소에 대한 연구가 필요하다. 또한, 상황인식 미들웨어의 성능 향상을 위해 중복되어지거나 룰 내부에서 불필요하게 순환 검사 되어지는 부분에 대한 처리를 통해 효율성을 증대시킬 필요가 있고, 유비쿼터스 환경에서 문제점으로 대두되고 있는 개인 프라이버시 및 정책적인 보호, 접근 권한에 대한 요소들이 필요할 것이다.

참 고 문 헌

[1] Oxygen project at the MIT, http://oxygen.lcs.mit.edu
[2] Aura project at the CMU, http://www-2.cs.cmu.edu/~aura
[3] http://www.cooltown.com
[4] http://www.research.microsoft.com/easyliving
[5] S. Y. Lee, "Context-aware Middleware for Ubiquious Computing Systems," 유비쿼터스 컴퓨팅 기술 워크샵 UCT 2004-

winter, pp.81-102.
[6] M. Roman, C. Hess and R. Campbell, "Gaia: A Middleware Infrastructure to Enable Active Spaces," IEEE Pervasive Computing, pp.74-83, 2002.
[7] Web Ontology Language (OWL), W3C, Recommendation, http://www.w3.org/2004/OWL/
[8] T. Gu, H. K. Pung and D. Q. Zhang, "A Middleware for Building Context-Aware Mobile Services," In Proceedings of IEEE Vehicular Technology Conference(VTC), 2004.
[9] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S.Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," IEEE Pervasive Computing, joint special issue with IEEE Personal Communications, 1(3), pp.33-40, July-September, 2002.
[10] H. Chen, F. Tim, and J. Anupam, "An Intelligent Broker for Context-Aware Systems," Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA, October, pp.12-15, 2003.
[11] D. Ejigu, M. Scuturici and L. Brunie, "Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing", Journal of Computer, 2008.
[12] T. Gu, H.K. Pung and D.Q. Zhang, "A service oriented middleware for building context aware services," Journal of Network and Computer Applications, Vol.28, pp.1-18, 1. 2005.
[13] E. J. Friedman-Hill, Jess(Java Expert System Shell), http://herzberg.ca.sandia.gov/jess/, 1999.
[14] S.J.H. Yang, J. Zhang and I.Y.L. Chen, "A JESS-enabled context elicitation system for providing contextaware Web services," Expert Systems with Applications, Vol.34, Issue 4, pp.2254-2266, May, 2008.
[15] B.N. Grosof, M.D. Gandhe and T.W. Finin, "SweetJess: Translating DamlRuleML to Jess," Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, 2002.
[16] T. Strang, C. Linnhoff-Popien, "A Context Modeling Survey," Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004-The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September, 2004.
[17] Semantic Web Rule Language (SWRL), http://www.daml.org/swrl.
[18] B. Schilit and M. Themier, "Disseminationg active map information to mobile hosts," IEEE Network, Vol.8, Issue 5, pp.22-32, 1994.
[19] A. K. Dey and G.D. Abowd, "Toward a Better Understanding of Context and Context-Awareness," CHI'2000 Workshop on the What, Who, Where, When and How of Context-Awareness, 2000.
[20] S. M. Chantzara, I. Sygkouna, S. Vrotis, I. Roussaki, and M. Anagnostou, "Context Management for the Provision of Adaptive Service to Roaming Users," IEEE Wireless Communications Magazine, 11(2), pp.40-47, April, 2004.
[21] Protégé : http://protege.stanford.edu/

남 태 우

e-mail : kaluas@pusan.ac.kr
2007년 부산대학교 정보컴퓨터공학부(학사)
2009년 부산대학교 컴퓨터공학과(공학석사)
2009년 3월~현 재 부산대학교 컴퓨터공학과 박사과정
관심분야 : RFID 기반 미들웨어, 상황인식 기법, 프로덕트 라인 공학, 소프트웨어 아키텍쳐 등

염 근 혁

e-mail : yeom@pusan.ac.kr
1985년 서울대학교 계산통계학과(학사)
1992년 플로리다대학교(Univ. of Florida) 컴퓨터공학과(공학석사)
1995년 플로리다대학교(Univ. of Florida) 컴퓨터공학과(공학박사)
1985년~1988년 금성반도체 컴퓨터연구실 연구원
1988년~1990년 금성사 정보기기연구소 주임연구원
1995년~1996년 삼성SDS 정보기술연구소 책임연구원
1996년~현 재 부산대학교 정보컴퓨터공학부 교수
관심분야 : 소프트웨어 재사용, 프로덕트 라인 공학, 소프트웨어 아키텍쳐, 적응형 소프트웨어 개발, RFID 기반 상황인식 미들웨어 등