멀티 에이전트 기반의 상황 인지 시뮬레이션을 위한 통합 개발 프레임워크의 개발

김태형*, 최진우*, 우종우*
*국민대학교 컴퓨터공학과
e-mail:cwwoo@kookmin.ac.kr

Development of Integrated Development Framework for Context-Awareness Simulation based on Multi-Agent System

Tae-Hyung Kim*, Jin-Woo Choi*, Chong-Woo Woo *
*Dept. of Computer Science, Kookmin University

요 약

상황인지 시뮬레이션이란 상황자료의 수집, 추론 및 결론 도출의 과정을 실제 환경이 아닌 가상의 시뮬레이션 환경 안에서 실험해 볼 수 있는 것을 말하며, 상황인지 기술과 DEVS(Discrete Event System Specification), 페트리-넷(Petri-Net)등의 기반 기술이 사용된다. 본 논문에서는 사용자들이 보다 편리하게 상황인지 시스템을 구축하고 실제보다 적은 시간으로 구축된 시스템을 실험할 수 있는 시뮬레이션 환경을 제공할 수 있는 통합 개발 프레임워크를 개발하고자 한다. 시뮬레이션 통합 개발 프레임워크 특징으로는, 첫째 상황인지를 위한 추론 기능을 가지는 에이전트와 시뮬레이션 기능들을 플러그인 및 라이브러리로 제공할 수 있고, 둘째, 통합환경 안에서 제공되는 도구들을 사용하여 사용자들이 보다 편리하게 개발 및 실험을 할 수 있다는 장점이 있다. 따라서 본 논문에서는 상황 아키텍처를 위한 모델 표현 계층, 멀티 에이전트 시스템을 위한 연산 계층, 환경과의 상호작용을 위한 인터랙션 계층, 그리고 시뮬레이션 계층인 4-계층구조의 통합개발환경을 연구하였다.

1. 서론

상황인지 시스템(Context-Awareness System)이란 다양한 상황 정보(Context Information)들을 수집하고 가공하여 사용자의 상황에 맞는 정보를 제공해주는 시스템이다. 상황인지 시스템에서의 상황(Context)이란 실 세계에 존재하는 실체의 상태를 특징화한 정보라고 할 수 있다. 상황인지 시스템은 사용자와 서비스의 사이에서 연관 관계에 따라 적절한 결과를 도출해내는 시스템으로서 현재의 사용자들은 자주 새로운 기술 및 도구와 그것들에 의한 새로운 상황에 직면하게 되므로 상황인지 시스템의 응용 범위는 방대하다. 현 시점에서 사용자들의 모든 요구를 원활하게 해결할 수 있는 상황인지 시스템을 설계 및 구현하는 것은 매우 어려운 일이며, 사용자들의 환경이 변화하는 속도에 맞추어 상황인지 시스템을 매번 새로 설계 및 구현하는 것도 어려운 일이다. 또한 이러한 시스템은 구현의 완료와 동시에 사용이 가능한 것이 아니라 사용하려는 상황에 따라 올바른 추론을 하기 위한 규칙과 그것에 대해 신뢰 가능한지의 여부를 판단할 수 있는 충분한 실험이 필요하다. 특히 실제 사용자들이 겪을 수 있는 상황에 대한 실험은 설계 단계에서는 예상하지 못한 의외의 상황이 발생할 가능성이 매우 높고, 시스템의 기능이 많고 적용 범위가 넓을수록 더욱 많은 예외 상황의 발생하고 실험 시간은 길어질 수 밖에 없다. 이와 같은 특성 때문에 시스템이 급격하게 변화하는 사용자들의 상황과 요구를 적절히 수용하지 못할 가능성이 매우 높다. 본 논문에서는 상황인지 시스템을 구현 후, 실 세계의 환경에 적용시키기 이전에 시뮬레이션(Simulation)환경[1]에서 실 세계에서 발생 가능한 예외 상황들을 실험 해 볼 수 있는 방법을 제공하고 예외 상황에 따르는 위험을 줄여 시스템 구현으로부터 실 세계에 적용하는 데까지의 시간을 줄일 수 있는 시스템으로 상황인지 시뮬레이션 통합 개발 프레임워크를 제안한다.

본 논문의 구성은 다음과 같다. 2 절에서는 상황인지 시스템 및 통합개발환경과 관련된 기존 연구를 기술한다. 3 절과 4 절에서는 본 논문에서 제시하는 시스템의 전반적인 개요와 통합개발프레임워크 설계를 설명한다. 5 절에서는 시스템의 구현에 대하여 기술하고, 마지막으로 결론을 맺는다.

2. 관련연구

현재의 상황인지 시스템은 초기 센서의 데이터에 대한 표현과 전달에 관한 연구에서 시작해 현재에 이르러서는 상황정보의 모델링의 단계까지 진행되고 있다. 이러한 상황인지 시스템은 사용자의 의도를 파악하고 그에 따른 적합한 행동을 하는 것에 목표를 가지고 상황 정보에 대한 처리뿐만 아니라 상황 정보를 사용한 추론의 역할도 하고 있다. 이러한 역할을 수행하

기 위해 온톨로지 모델을 사용하는 SOCAM, GAIA 등의 미들웨어들이 고안되었지만, SOCAM 은 사용자의 피드백 정보를 활용할 수 없고, GAIA 는 사용자 피드백 정보를 활용할 수 있으나 추론기능이 없는 등의 제약이 있다[2]. 이에 따라 상황인지 시스템의 개발자들은 상황인지 시스템을 구축하는데 있어 각각의 시스템에서의 제약사항 안에서 개발을 하거나 필요한 기능을 추가 개발할 필요성이 있었으며 기존 시스템들의 유용한 기능을 사용하기 위해 개발자가 개발 시 사용할 시스템을 학습하고 실험을 거친 후 개발하는 현재까지의 형태를 벗어나 개발하려는 시스템을 설계한 후, 본 논문에서 제안하는 GUI 및 이클립스 플러그인이 내장되어있는 MUST 를 기반으로 개발자가 원하는 기능 등을 개발환경에 맞게 구축하여 시스템 자체의 제약사항을 최대한 줄이고자 하는 것이 본 통합 개발 프레임워크의 목표이며 이를 위해 사용된 대표적인 시스템들에 대한 특징을 아래와 같이 기술했다.

JCAF(Java Context-Awareness Framework)는 개발자가 상황인지 응용 프로그램을 쉽게 개발할 수 있도록 하는 목표를 가진 Java 로 구현된 프레임워크로서 상황인지 프로그램 개발을 위한 API 를 제공한다[3]. JCAF 에서의 가장 중요한 관점은 상황 서비스(Context Service), 개체와 상황(Entities and Context), 상황 클라이언트(Context Client), 상황 이벤트(Context Events)이다. 상황 서비스는 개체들을 위한 상황 정보들을 받고, 관리하고, 저장하며, 분산하는 기능을 한다. 개체와 상황은 상황 정보화 하고자 하는 대상과 상황 아이템(context item)들간의 관계(relationship) 집합으로 이루어져 있으며, 개체들에 대한 상황 정보를 등록하거나 개체의 변화를 감지하는 기능은 상황 클라이언트에서 담당한다. 또한 상황 이벤트를 두어 분산 이벤트(distributed event)의 사용을 지원한다. JCAF 는 위 네 가지의 관점에 기반하여 동작할 수 있는 개체, 상황, 관계, 상황 아이템, 상황 서비스, 개체 리스너(EntityListener)등의 라이브러리를 제공하며 이를 활용해서 현실 세계의 상황을 표현할 수 있게 해준다.

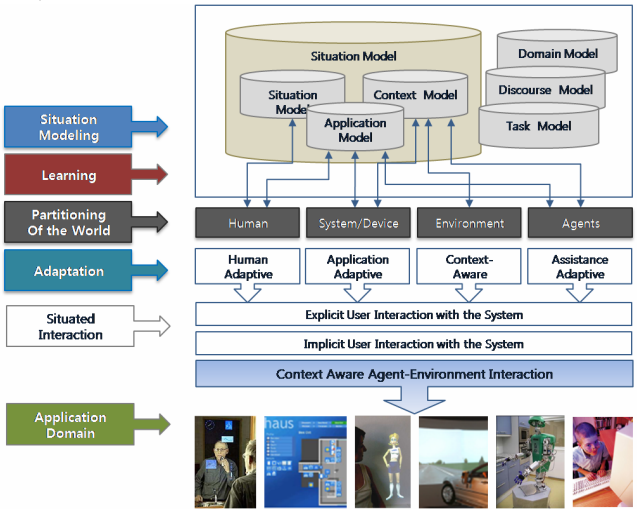
IDE(Integrated Development Environment)는 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어를 말한다. 통합 개발 환경(IDE)의 등장 이전에는 소프트웨어를 개발하기 위해 컴파일러, 소스코드 에디터, 디버거 등을 각각 따로 사용해야 했으나 현재는 통합개발환경이 주를 이룬다. 대표적인 통합 개발 환경으로 이클립스가 있으며 본 논문에서 제안한 시스템은 이클립스 플랫폼 위에서 사용 가능한 프레임워크의 개발을 목표로 하고 있다.

RBS(Rule-based system)는 특정 분야의 지식을 규칙으로 표현하여 해당 분야의 문제를 해결하는데 표현된 규칙을 적용하는 시스템이다. 규칙 기반 시스템은 다음과 같이 정의될 수 있다. 첫째, 인간의 지식을 if-then 규칙의 조건문 형식으로 표현한다. 둘째, 지식 기반(knowledge base)의 크기에 비례하여 문제해결 기술이 증가한다. 셋째, 적절한 규칙들의 선택과 조합으로 넓은 범위의 복잡한 문제들을 해결 가능하다. 넷째, 최적의 규칙을 결정하고 그들의 실행 순서를 결정할 수 있다. 마지막으로, 자연어로 추론 결과에 도달하기까지의 논리와 과정을 설명할 수 있다.

본 시스템에서는 규칙기반 추론엔진과 GUI 로 구성된 룰 에디터(Rule editor)를 포함하며 상황인지 시스템 개발자는 이러한 도구를 사용하여 개발과 운용 시 복잡하게 표현된 규칙들을 손쉽게 파악하고 추가, 삭제 및 변경을 할 수 있게 하였다. 이에 대한 자세한 설명은 다음 장 설계 부분에서 하도록 하겠다.

3. 시스템 개요

본 논문을 통해 소개하는 연구는 에이전트들의 다양하고 복잡한 과업 수행에 있어 환경과 상호작용하는 행위, 즉 의사결정을 위한 추론(상황인지), 수행계획 및 에이전트간 상호작용 등을 분석함으로써, 이해 및 설명 가능한 상황모델을 구성하여 적응력 있는 지능형 멀티 에이전트 시스템의 개발을 최종 목표로 한다 (그림 1).



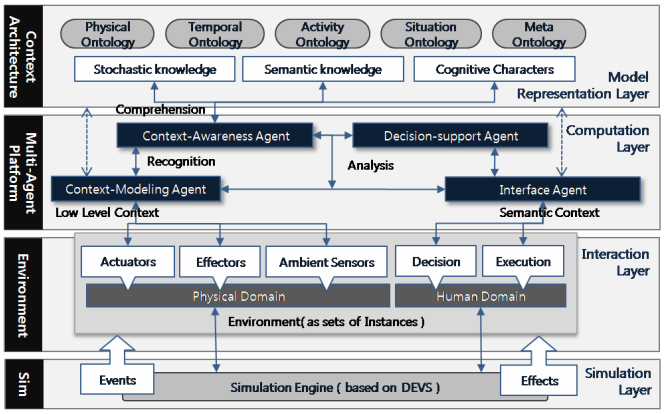
(그림 1) 시스템 개요

4. 시스템 설계

최종 연구의 일환으로 설계된 MUST(Multi-agent based United Simulation Toolkit) 시스템의 전반적인 이해를 돕기 위해 시스템 아키텍처를 기술하고, 현재 개발 완료된 단일 에이전트를 중심으로 기술한다.

가. 시스템 아키텍처

최종 목표 시스템(그림 2)은 상황 아키텍처를 위한 모델 표현 계층, 멀티 에이전트 시스템을 위한 연산 계층, 환경에 해당하는 인터랙션 계층, 그리고 시뮬레이션 엔진이 존재하는 시뮬레이션 계층인 4-계층구조로 구성되며, 이들 각각의 계층에 대하여 요약하면 <표 1>과 같다.

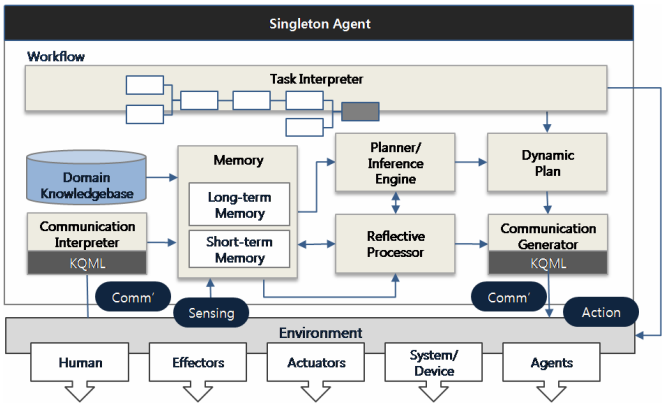


(그림 2) 시스템 아키텍처

<표 1> 4-계층구조

계 층	내 용
Model Representation Layer	상황 모델링(Context Modeling) 기술을 사용하여 이벤트-행위 기반의 에이전트 모델을 구축
Computation Layer	고유의 도메인 지식을 소유하는 에이전트는 상호간 협력을 통해 상황에 대한 이해 및 예측하며 목표를 성취하는 지능형 멀티 에이전트 시스템이 존재
Interaction Layer	시뮬레이션 환경을 구성하는 센서, 인간, 그리고 다른 에이전트의 반응으로부터 변화되는 정보를 획득한다
Simulation Layer	사실과 같은 환경변화를 생산 가능토록 DES(Discrete Event System) 기반의 시뮬레이션 엔진이 존재

- 단일 에이전트 아키텍처



(그림 3) 단일 에이전트 아키텍처

멀티 에이전트 시스템을 구성하는 다수의 단일 에이전트들은 (그림 3)과 같이 설계된 공통 기초 구조로부터 확장되어 구현되도록 설계한다. 이를 구성하는 4개의 주요 컴포넌트들로는 에이전트간 통신, 도메인 지식, 계획수립기, 추론엔진 컴포넌트로 구성되며, 이들 각각의 기능은 <표 2>와 같다.

<표 2> 단일 에이전트의 기본 구조

컴포넌트	내 용
Inter-Agent Communication	• 에이전트 간 상호협력을 위한 표준화된 통신 메커니즘(KQML/ACL) • 에이전트의 통신 메시지 해석기 • 연합구조(Federated Architecture)
Domain Knowledge	• 에이전트 고유의 도메인 지식 • 지식 토폴로지(DAG: Directed Acyclic Graph)
Planner	• Action 의 수행 절차 • FSM(Petri-net)
Inference Engine	• 지식 추론 • RETE Algorithm

- 추론엔진(Inference Engine) / 계획수립기(Planner)

에이전트의 추론엔진은 Forgy 에 의해 고안된 Rete 알고리즘을 사용한다. Rete 알고리즘[4][5] 은 정방향 추론으로써, 조건부(LHS)을 표현하기 위해 dataflow network 를 사용한다. 주된 특징으로는 규칙간 중복되는 조건부를 공유함으로써 기억공간의 낭비를 줄일 수 있으며, 이를 위해 전체 규칙의 조건부들을 컴파일하여 방향성 비순환 그래프(DAG) 기반의 분류 네트워크(Discrimination network)로 표현한다. 규칙의 조건부에 있는 패턴들에 따라 컴파일된 분류네트워크를 이용하여 규칙의 매칭작업을 효과적으로 처리하였다. 본 시스템에서는 라이브러리로 개발된 Rete 알고리즘으로부터 플러그인(Plug-In)을 개발한다.

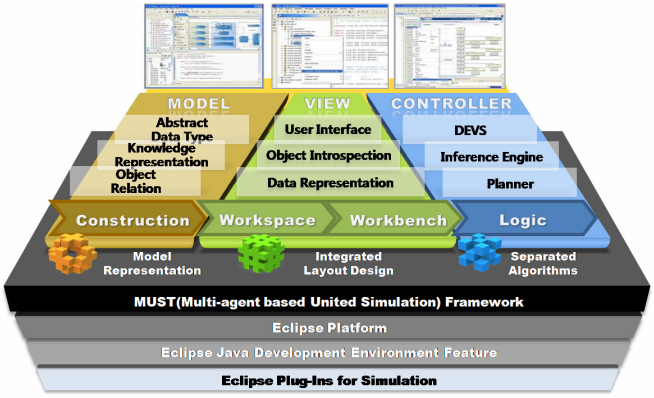
계획수립기는 유한상태기계(FSM: Finite State Machine)을 기반으로 에이전트의 Action 에 부합하는 일련의 행위에 관한 정보를 기반으로 기본적인 계획을 수립하고 이를 내부의 워크플로우에서 관리되도록 설계한다. 이러한 유한상태기계 기반의 계획수립기는 시스템을 구성하는 모든 에이전트들의 상태를 확인할 수 있으며, 시뮬레이션 설계 모델을 단순화 할 수 있는 장점이 있다. 이를 가능케 하기 위하여 에이전트의 계획수립기는 페트리-넷(Petri-Net)으로 구현된다.

이들 추론엔진과 계획수립기 모두는 통합개발프레임워크 상에서 용이한 조작이 가능하도록 사용자 인터페이스를 가지는 플러그인의 형태로 개발된다. 이를 위해 모델과 엔진은 상호 분리하여 설계함으로써, 그래픽 편집 도구를 사용하여 쉽게 모델을 구축하고 구조적 검증이 가능토록 한다. 엔진은 외부 라이브러리로부터 플러그인의 형태로 개발되도록 설계함으로써, 다수의 플러그인 에서의 재사용성을 높이도록 구현한다.

나. 통합 개발 프레임워크

전술한 시스템 아키텍처(그림 2)를 구성하는 컴포넌트들은 통합개발환경인 이클립스 기반의 플러그인(Plug-in)들[6][7] 로 개발되며, 기존 라이브러리 혹은 새롭게 개발된 플러그인들 간의 상호연동 및 조립을 통해 구축한다. 이클립스는 확장 가능한 개발 환경을 제공하는 개방형 소프트웨어로써, 제공되는 다수의 플러

그인(Plug-Ins)들을 조립하여 새로운 개발 프레임워크 구현을 지원한다. 이를 위해 제공되는 플러그인 개발 환경(PDE)과 자바 개발 키트(JDT) 환경 하에 통합 개발 프레임워크를 설계 및 개발한다.

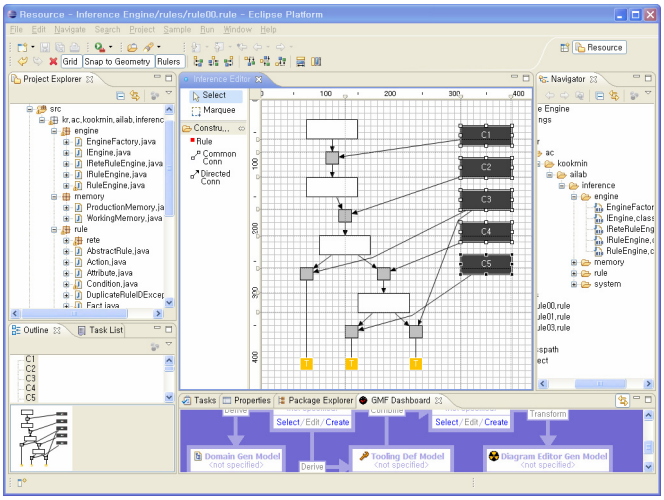


(그림 4) 시뮬레이션 통합 개발 프레임워크의 구조

시뮬레이션을 위한 통합 개발 프레임워크(그림 4)는 두 가지 관점, 즉 구조적 관점과 절차적 관점을 고려하여 설계하며, 이들은 다음과 같다. 첫째, 종 방향의 구조적 관점은 모델(Model), 뷰(View), 그리고 컨트롤러(Controller)로 구분되며, 설계를 위한 플러그인들의 기능에 따른 집합이다. 둘째, 횡 방향의 절차적 관점은 생성(Construction), 워크스페이스 (Workspace), 워크벤치 (Workbench), 논리(Logic)로 구분되며, 통합개발 환경에서 플러그인 사이의 상호통신을 의미한다.

5. 시스템 구현

본 논문에서 제안한 시뮬레이션 통합 개발환경을 구현한 통합 개발 프레임워크 화면이다. 중앙의 규칙 에디터는 이클립스의 GEF/GMF[8] 기반으로 작성된 플러그인으로써 규칙의 용이한 편집을 가능케 하였으며, 외부라이브러리 기반의 추론엔진 플러그인은 구축된 규칙으로부터의 추론과정을 검증해 볼 수 있도록 구현하였다(그림 5).



(그림 5) 시뮬레이션 통합 개발 프레임워크

6. 결론

본 논문에서 제안한 4-계층구조의 시뮬레이션 통합 개발 환경은 상황 아키텍처를 위한 모델 표현 계층, 멀티 에이전트 시스템을 위한 연산 계층, 환경과의 상호작용을 위한 인터랙션 계층, 그리고 시뮬레이션 계층구조를 제시하였으며, 지능형 에이전트를 위한 Rete 알고리즘[9] 기반의 추론기관과 에이전트의 절차적 행위들을 위해 페트리-넷 기반의 계획수립기를 제시하였다. 이러한 제시된 방법들을 기초로 하여 상황인지 시스템의 개발을 위한 시뮬레이션 환경 구축을 보다 효율적으로 제공하는 통합 개발 프레임워크를 설계 및 구현하였다.

현재 본 논문에서 제안한 시뮬레이션 통합 개발 프레임워크의 구현이 진행 중이며, 시스템과 상호작용하는 다양한 환경의 변화를 지원할 수 있는 시나리오 기반의 DEVS[10] 플러그인과 에이전트간 상호통신을 위한 KQML/ACL 기반의 통신 플러그인 개발에 대한 연구도 진행되고 있다

Acknowledgement

본 논문은 서울시 산학연 협력사업 (10848) 의 지원을 받아 연구 수행된 논문입니다.

참고문헌

[1] 우종우, 김대령, “멀티 에이전트 기반의 지능형 시뮬레이션 도구의 개발”, 한국컴퓨터정보학회 논문집, 제 12 권 제 6 호, 2007

[2] 장인우, 우종우 “CBR 을 이용한 상황인지 시스템의 학습능력”, 한국정보과학회 가을 학술발표논문집 pp235-240, 2009

[3] Jakob E. Bardram “Design, Implementation, and Evaluation of the Java Context Awareness Framework (JCAF)”April, 2005

[4] Robert B. Doorenbos “Production Matching for Large Learning Systems” January 1995

[5] Erich Gamma, Kent Beck. Contributing to Eclipse: Principles, Patterns, and Plugins, Addison-Wesley Professional, October 31, 2003

[6] Eclipse.org Home, www.eclipse.org

[7]Eclipse.org, "Eclipse Platform Technical, Overview", www.eclipse.org/articles/Whitepa

[8] IBM, "Eclipse Development-using the Graphical Editing Framework and the Eclipse Modeling Framework", ibm.com/redbooks, 2004.

[9]Forgy C. L, “Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem”, Artificial Intelligence, 19 17-37, 1982

[10] Ziegler, B.P., and Kim, T.G., and Praehofer, H. “Theory of Modelling and Simulation”, 2000