JADE 를 이용한 에이전트 구축과 응용
Agent Frame and Application using JADE

* # 임선종 [1], 송준엽 [1]
* # S. J. LIM[1](sjlim@kimm.re.kr), J. Y. SONG[1]
[1] 한국기계연구원, 지능기계연구센터

Key words : JADE, Agent, AMS, DF, RMA

1. 서론

JADE(Java Agent Development Framework)는 다중 에이전트 개발을 용이하게 하는 미들웨어이다. JADE 는 FIPA-OS(Foundation for Intelligent Physical Agents-Open Source)를 기반으로 하고 있다. FIPA-OS 에서 제공하는 플랫폼은 AMS(Agent Management System), DF(Directory Facilitator) 그리고 MTS(Message Transport System)으로 구성되어 있다. AMS 는 에이전트 플랫폼에 대한 접근과 사용에 대한 관리를 하며 하나의 플랫폼에 하나의 AMS 만이 존재한다. DF 는 에이전트에 대한 전화 번호부의 기능을 수행한다. MTS 는 같은 플랫폼 뿐 만 아니라 다른 플랫폼 내에서 교환되는 메시지를 관리한다.

본 연구는 FIPA 표준을 따르며 에이전트 구현을 용이하게 하는 JAVA 의 패키지를 분석하고, E-Beam(Electron-Beam) 가공기 운영 시스템에 확장될 수 있는 플랫폼의 프로토타입(Prototype) 구현을 보이고 있다.

2. JADE 패키지

JAVA 기반에서 에이전트가 실행되는 환경을 런 타임 환경(Run time environment)이라고 하며 이것은 컨테이너(Container)라고 하고 복수의 에이전트를 포함할 수 있다. 컨테이너의 집합을 플랫폼이라고 한다. 한 플랫폼에서 처음 시작된 컨테이너를 메인 컨테이너라고 하며 이 이후에 생성된 것을 노멀 컨테이너하고 한다. 모든 컨테이너는 메인 컨테이너에 등록을 한다. 메인 컨테이너는 AMS 와 DF 에이전트를 갖게 된다. AMS 는 플랫폼에 있는 에이전트 이름과 플랫폼의 권한 등을 표현한다[1-3]. Fig. 1 은 플랫폼, 메인 컨테이너 그리고 노멀 컨테이너의 관계를 보이고 있다.

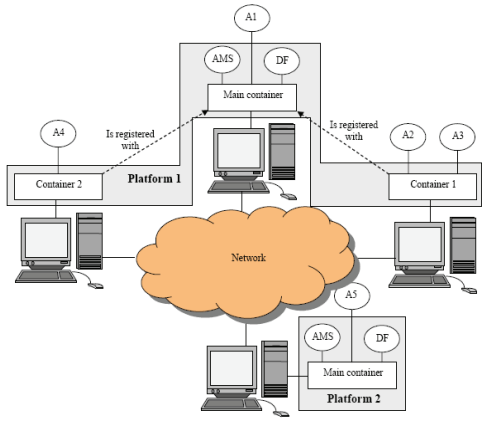

Fig. 1 Relationship between main container and normal container

에이전트는 에이전트 라이프 사이클(Waiting, Suspended, Transit, Initiated) 상태중의 하나에 있게 된다. 에이전트의 구현은 “Behaviour” 객체를 통해 구현되며 “addBehaviour(Behaviour”와 “removeBehaviour(Behaviour)”를 통해 확장시킨다. “Behaviour” 클래스는 “SimpleBehaviour”, “CyclicBehaviour”, “OneShotBehaviour”, “ParallelBehaviour”, “SequentialBehaviour”, “FSMBehaviour”로 구성되어 있다. “SimpleBehaviour”는 부가적인 목적으로 구성되 있지 않은 임무를 수행하기 위한 클래스며 주기적인 목적과 단발적인 목적의 실행으로 나누어진다. “ParallerBehaviour”는 서브 목적의 동시 실행을 수행하기 위한 클래스이며 “SequentialBehaviour”는 서브 목적의 순차적 실행을 위한 클래스이다. “CompositeBehaviour”는 많은 목적들로 이루어진 목적을 수행하는 클래스이다.

에이전트의 임무 실현은 쓰레드를 기반으로 구현된다. 에이전트는 쓰레드를 기반으로 함으로 몇 개의 에이전트가 동시에 실행될 수 있다. 이것은 한 에이전트가 하나의 쓰레드를 가지도록 설계하며 보다 나은 성능을 가지며 같은 자원을 공유하는데 생기는 동기 문제를 해결하는데 있어서 이점이 있다. DF 는 목적 달성을 위해 요청하는 서비스를 제공할 수 있는 에이전트를 찾을 수 있도록 전화 번호부 기능의 서비스를 제공한다. DF 에 대한 서비스는 jade.domain.DFService 를 통해 구현되며 “register”, “deregister”, “modify” 그리고 “search” 메소드(Method)를 이용해 구현된다. DF 에 대한 접근을 위해 “getDefaultDF()”를 이용하고 있다. 사용되는 통신 방법은 비동기 메시지 전송(Asynchronous message passing)이다. DF 는 에이전트가 수행할 수 있는 서비스 등을 나타내도록 허용하는 것으로 서비스가 필요한 에이전트는 DF 를 검색함으로 해서 에이전트를 찾게 된다.

JADE 는 통신을 위해 비동기 메시지 전송 방식을 사용하고 있다. 컨테이너의 각 에이전트는 다른 에이전트가 전송한 메시지를 담을 수 있는 메일 박스 기능을 수행하는 메시시 큐를 가지고 있다. 에이전트는 메시지 큐를 통해 메시지 수신 여부를 알게 된다. 통신을 위해 사용되는 메시지는 FIPA 에 정의된 ACL 언어를 사용한 포맷(Format)을 사용한다. 구성되는 포맷은 “sender”, “receivers”, “performative”, “content”, “language”, “ontology” 등이다.

ACLMessage 클래스는 에이전트 사이에 교환되는 ACL 메시지를 표현한다. ACLMessage 의 객체를 이용해 메시지를 만들고 “Agent.send()” 메소드를 이용해 전송된다. 메시지의 수신은 “receive()”가 이용된다.

jade.core 는 시스템의 커널(Kernel)을 구현하며 응용 프로그램을 위한 에이전트 클래스를 포함하고 있다. 서브 패키지로 jade.core.behaviours 를 가지고 있으며 이것은 에이전트의 임무를 구현한다. Jade.lang.acl 서브 패키지는 FIPA 표준에 따른 에이전트 통신 언어를 처리하기 위해 제공된다. Jade.content 는 사용자 정의 ontologies 와 content-languages 를 지원하는 클래스를 포함하고 있다. Jade.domain 패키지는 AMS 와 DF 에이전트 정의와 에이전트 관리를 위한 JAVA 클래스를 포함하고 있다. 서브 패키지인 jade.domain.JADEAgentManagement 는 ontology 를 포함하고 있다. Jade.gui 패키지는 에이전트 ID, 에이전트 표현, ACL 메시지를 표시하고 편집하기 위한 GUI 를 만드는데 유용한 일반적인 클래스를 포함하고 있다. jade.mtp 는 JADE 프레임워크에 쉽게 통합되도록 하기 위해 메시지 전송 프로토콜(Message Transport Protocol)을 구현해야 하는 JAVA 인터페이

스를 포함하고 있다. jade.proto 는 standard interaction protocols 를 나타내는 클래스를 포함하고 있다.

3. JADE 패키지 응용

E-Beam(Electron-Beam) 가공기에 대한 에이전트는 사용자 인터페이스(User Interface) 에이전트, E-Beam Machining 에이전트, 협력 에이전트, 센서 에이전트 그리고 빔 에이전트로 모델링하였다[4]. 사용자 인터페이스는 기본적으로 운영 화면을 통해 MMI(Man-Machine-Interface)의 기능과 협력, E-Beam machining 센서 그리고 빔 제어 에이전트의 상태 변화를 표시하는 기능을 수행한다. E-Beam machining 에이전트는 챔버(Chamber) 내의 진공도 유지, 진공 프로세스 관리, 진공 밸브 제어 그리고 초기화 기능을 수행한다. 초기화 작업은 완료된 후 고전압 발생기를 제어하며 전자빔을 발생시킨다. 빔 제어 에이전트는 전자빔 가공을 수행하는 것으로 경로에 따른 빔의 이동 및 차단, 빔의 초점 조절 기능을 수행한다. 협력 에이전트는 외부 에이전트와 협력을 위해 내부 그리고 외부 에이전트 주소, ID, 목적 그리고 상태 등을 관리한다. Fig. 2 은 에이전트 다이어그램을 보이고 있다.

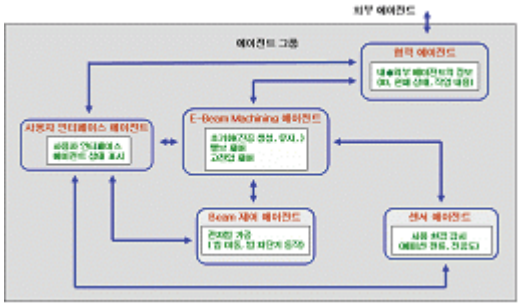

Fig. 2 Agent diagram for E-Beam system

메인 컨테이너와 노멀 컨테이너가 설정되면 local agent platform description 에서 확인할 수 있으며 Fig. 3 는 화면을 보이고 있다.

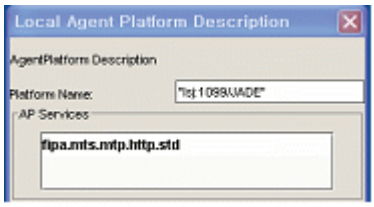

Fig. 3 Construction of agent platform

DF GUI 는 등록된 에이전트의 표현을 볼 수 있으며 에이전트를 등록 및 등록 철회, 등록된 에이전트의 표현 수정 그리고 에이전트 표현에 대한 검색 작업을 수행한다. Fig. 4 은 DF 에 등록된 에이전트 화면을 보이고 있다.

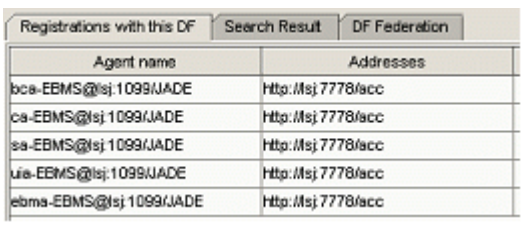

Fig. 4 DF window

각 에이전트 사이에 송수신되는 메시지를 확인하기 위해서는 Inspector 를 이용한다. Inspector 는 송수신되는 메시지의 내용을 볼 수 있는 기능을 제공하고 있다. Fig. 5 는 센서 에이전트에 대한 Inspector 의 화면을 보이고 있다.

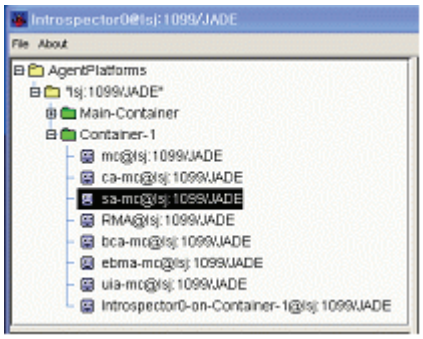

Fig. 5 Inspector window

RMA 는 jade.tools.rma.rma 의 인스턴스 JAVA 객체로 등록된 모든 에이전트와 에이전트 플랫폼의 라이프 사이클을 통제한다. 모든 인스턴트가 다른 로컬 이름을 가지고 있으며 하나 이상의 RMA 가 같은 플랫폼에서 시작될 수 있을 뿐 만 아니라 하나의 RMA 가 같은 에이전트 컨테이너 상에서 실행될 수 있다. RMA 는 별도의 윈도우를 통해 ACL 메시지를 선택된 에이전트에 보낼 수 있는 기능도 제공한다. Fig. 6 은 RMS window 를 보이고 있다.

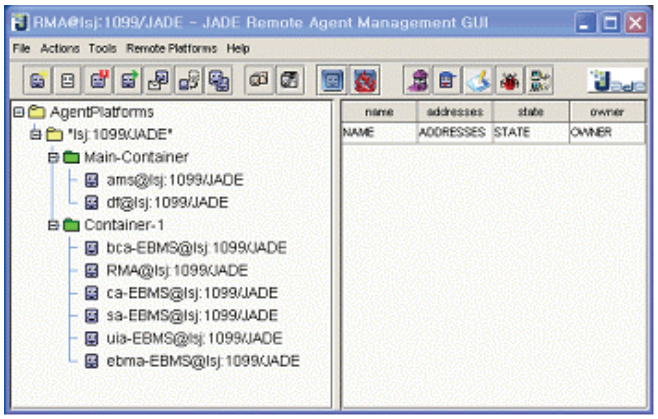

Fig. 6 RMS window

4. 결론

본 연구는 FIPA-OS 의 표준을 따르고 있는 JADE 패키지 3.3.을 분석하고 JDK 1.4 를 이용해 플랫폼 구현을 보이고 있다. 구축된 플랫폼은 platform window, DF window 그리고 inspector window 를 통해 에이전트 시스템의 기본 프레임 기능을 확인할 수 있었다. 이것은 독립된 E-Beam system 의 운영 시스템 뿐 만 아니라 가공 라인 구축에서 협력 시스템의 운영 시스템으로 활용할 수 있음을 보여주고 있다.

참고문헌

1. http://jade.tilab.com/
2. http://www.fipa.org
3. http://www.fipa.org/specs/
4. 임선종, 이찬홍, 송준엽, “전자빔 가공기에 대한 에이전트 응용,” 한국공작기계학회지, Vol. 16, No. 2, pp. 44-49, 2007