한국 CAD/CAM 학회 논문집
제8권 제1호 2003년 3월 pp. 55-63

학술논문

ADE : 공학 에이전트 개발 환경

구본석*, 이수홍**

ADE: Agent Development Environment for Engineering

Gu, B. S.* and Lee, S. -H.**

ABSTRACT

ADE is a software tool for the design and implementation of multi-agent systems. ADE allows a designer to draw a multi-agent system graphically, specify the necessary properties, and deploy their applications. ADE offers a set of intuitive, easy to use interfaces that enable a designer to completely specify the agents and agent interactions in a multi-agent system. In this environment, JATLite[1] is improved significantly. Furthermore, ADE provides a unique set of features for a multi-agent system design tool. An agent description method based on Design roadmap[2] theory, a hierarchy of agents, and a fully featured Java-based Graphical User Interface are combined in ADE. This distinct combination of features makes ADE stand out among the existing multi-agent system design tools. This paper presents the research related to the application of the ADE, along with descriptions of its design and implementation.

Key words : Agent, ADE, KQML, KIF, K-Code, K-File, En-Router

1. 서 론

분산된 설계 시스템들을 하나의 일관된 통합 시스템으로 구축하기 위해서는 첫째 분산된 이 기종 시스템 간의 원활한 정보 교환 수단을 확보하고, 둘째 설계, 생산 정보 및 전문가의 경험적 지식을 공유할 수 있는 방법을 찾아야만 한다. 이의 해결 방안으로 인공지능 분야에서 주목받는 소프트웨어 에이전트 시스템을 들 수가 있다[3].

많은 연구자들이 에이전트 기술을 이용한 시스템 통합을 위한 방안들을 제시하였으며[4], 기존 시스템의 활용을 위한 구체적인 방안의 하나로 에이전트화(agentfication)가 제안되었다[5]. 이러한 에이전트 시스템의 채택으로 결함에 대한 내성 증대[6], 주어진 문제의 분해로 인한 유연성과 확장성[7], 소프트웨어 비용의 감소[8] 등과 같은 이점을 얻을 수 있다.

에이전트와 관련된 많은 시도들은, 에이전트를 사용한 통합에 따른 전체적인 시스템의 구조에 대한 제안[4,8,9], 에이전트 개발의 위한 도구의 개발로 나눌 수 있으며, 이 중 에이전트 개발을 위한 도구의 개발은 JATLite[1], JAFMAS[10]와 같은 비 GUI 도구와 JiVE[11], 에이전트 빌더[12]와 같은 GUI 도구로 구분할 수 있다.

본 연구에서는 지식의 에이전트화라는 개념 하에 개발된 새로운 에이전트 개발 환경인 ADE에 대해 다룬다. 개발된 시스템은 멀티 에이전트 시스템 개발을 위한 독자적인 요소들을 가진다. 디자인 로드맵 이론에 기초를 둔 개별 에이전트 기술 방법, 개체와 지식 파일로 구분된 에이전트 구조, 지식의 에이전트화, 계층 구조를 가진 에이전트 구성 그리고 자바 기반의 그래픽 사용자 인터페이스는 ADE의 주요한 특징들이다. 이러한 요소들은 ADE를 다른 멀티 에이전트 시스템 개발 도구들과 차별되게 한다.

2. 관련 연구

2.1 에이전트

에이전트라는 용어는 다양한 분야에서 통용되고 있으나, 각각 서로 다른 정의와 개념을 사용하고 있다.

Nwana[13]은 이러한 에이전트를 이동성의 유무, 심의/반응(deliberative/reactive), 에이전트의 역할, 에이

*연세대학교 기계공학과
**연세대학교 기계공학과
- 논문투고일: 2002. 03. 19
- 심사완료일: 2003. 01. 23

전트의 특성-자율성(autonomy), 학습성(learning), 협동성(cooperation) 등을 이용하여 다음과 같은 8가지로 분류하였다.

· Collaborative agents	· Reactive agents
· Interface agents	· Hybrid agents
· Mobile agents	· Smart Agents
· Information/Internet agents	· Heterogeneous agent systems

본 연구에서는 Genesereth와 Ketchpel[14]의 Agent-Based Software Engineering(ABE) 관점에 따른 에이전트 개념을 사용한다. ABE의 관점에서 에이전트는 에이전트 통신 언어(ACL, Agent Communication Language)로 타 에이전트와 메시지를 교환하는 소프트웨어 컴포넌트이다. 에이전트들은 ACL의 표현력 범위 내에서 이를 통해 논리적 정보와 명령, 데이터 등을 교환함으로써 협동적으로 일을 수행한다.

2.2 디자인 로드맵(Design Roadmap)[2]

디자인 로드맵 이론의 가장 기본적인 모델 표현 요소의 단위는 Information Processing Unit(IPU)이다. Fig. 1에서 보는 바와 같이 IPU는 정의된 작업을 처리하는 최소 단위의 표현 요소이다. IPU는 자신의 기능에 대한 설명과 기능의 수행에 필요한 자원에 대한 요구 사항을 가지고 있으며 이전 IPU와 다음 IPU와의 연결을 위한 소켓과 상위 IPU와 하위 IPU와의 계층 구조의 정의를 위한 소켓을 가지고 있다. IPU는 노드로 구체화되어 사용되며 링크(Link)을 이용하여 서로 연결되어 IPU간의 관계를 형성한다.

디자인 로드맵 이론에서 사용하는 노드는 크게 피처(Feature) 노드와 태스크(Task) 노드로 나뉘며 피처 노드는 태스크 노드를 수행하기 위한 입력이나 태스크 노드에서 출력된 결과를 표현하는 요소이며 태스크 노드는 기능이 할당되어 있는 작업 수행을 의미하는 표현 요소가 된다. 다른 하나의 노드는 어떠한 태스크 노드의 수행을 맡은 인력이나 도구를 의미하는 표현 요소인 에이전트(Agent) 노드이다. 링크는 일반적인 작업의 순서를 나타내는 표현 요소로의 기능을 가지고 있다. 부가적 기능을 가진 연결로는 태스크 노드에서 작업의 수행 결과로 출력된 피처 노드가 다시 태스크 노드에 영향을 미치는 것을 의미하는 표현 요소인 피드백 연결과 태스크 노드에서의 작업 결과가 피처 노드에 영향을 미치는 것을 의미하는 표현 요소인 작업기능효과(Side Effect) 링크가 있다. 또한 피처 노드 간의 제약 조건을 의미하는 표현 요소인 제약(Constraint) 링크와, 태스크 노드를 수행하는 역할을 담당하는 에이전트 노드와 해당 태스크 노드와 연결하는 표현 요소인 할당(Assignment) 링크가 존재한다.

3. 시스템 설계

3.1 개별 에이전트

본 연구에서는 Fig. 2의 상단과 같은 형태로 생명체의 외부 자극에 대한 반응을 파악하였다. 생명체는 외부 자극에 대한 자신의 지식을 바탕으로 이에 대한 적절한 반응을 하게 된다. 즉, "1 더하기 2가 무엇인가?"라는 외부 자극에 대해 해당 개체는 더하기에 대한 지식을 이용하여 그 결과가 3이라는 것을 처리한 후 이를 반환한다. 이런 형태를 이용하여 본 연구에서는 Fig. 2의 하단에서와 같은 개별 에이전트의 구조를 제안하였다.

개별 에이전트는 이진 컴퓨터 프로그램 코드인 개체와 요청된 메시지의 처리 방법인, 지식 파일 K-File/K-Code로 구성된다. 개별 에이전트의 개체는 동일하다. 즉 동일한 이진 프로그램 코드를 가진다. 개별 에

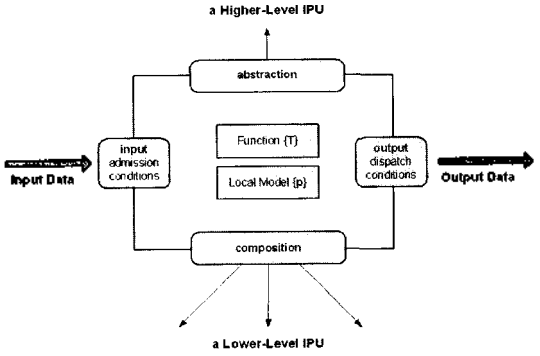

Fig. 1. Design roadmap : IPU[1].

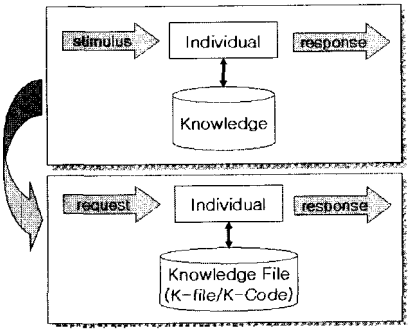

Fig. 2. Concept of an individual agent.

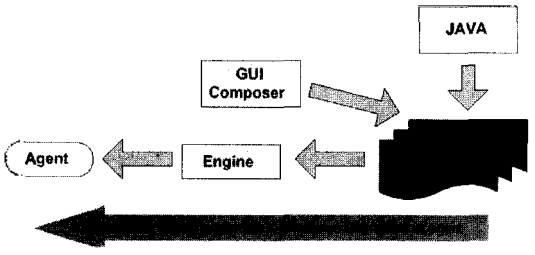

Fig. 3. Execution of an individual agent.

이전트는 동일한 개체를 가지나 K-File 또는 K-Code에 의해서 차별화가 이루어 진다. 해당 에이전트의 행동 형태는 K-File 또는 K-Code에 의해 정의되므로 서로 다른 K-File/K-Code는 서로 다른 에이전트를 만들게 된다.

개체와 지식 파일이라는 구조로 에이전트를 구성함으로 인해 에이전트들이 지닌 지식은 쉽게 전파될 수 있다. 해당 개별 에이전트의 개체는 동일하며, 그 기능은 지식 파일에 의해 정의됨으로 지식 파일의 전송만으로 해당 에이전트의 기능을 쉽게 다른 개별 에이전트로 전송할 수 있다

ADE의 개별 에이전트는 공통된 에이전트의 몸체인 개체와 개별 에이전트의 차별화를 위한 지식 파일인 K-File/K-Code로 구성된다. 그러므로 개별 에이전트는 작성된 K-File/K-Code를 에이전트 개체, 즉 엔진을 이용하여 수행한다.

3.1.1 K-File

일반적으로 에이전트의 개발은 자바나 C++와 같은 프로그램 언어에 의해 이루어 진다. 이러한 현상의 원인 중 하나는 에이전트가 프로세스를 가져야 하며, 이런 프로세스를 각종 프로그램 언어를 이용하여 정의를 하기 때문이다. 이에 본 연구에서는 K-File이라는 지식 파일을 제안하였으며, 이의 작성을 위한 툴인 K-Composer를 개발하였다. Fig. 4는 K-File의 개념을 보여 준다. ADE가 추구하는 바는 지식의 에이전트화이다. 개발자는 자신의 지식 중 프로세스로 표현 가능한 요소를 K-File이라는 형태로 표현하게 되며, ADE는 이렇게 표현된 지식, 즉 K-File을 에이전트화 한다.

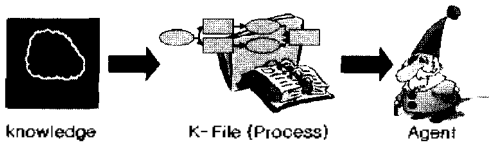

Fig. 4. Concept of the K-File.

```
<?xml version="1.0" encoding="KSC5601"?>

<K-File>

  <Cookie Name="Loop_Auction">
    <Version Value="" />
    <Writer Value="Jay" />
    <Date Value="2001. 1.3" />
    <Description Value="Description" />
  </Cookie>

  <AgentInfo AgentName="Loop_Auction" AgentPassword="Loop_Auction" AgentPort="0">
    <Regist RName="RouterRegistrar" RPort="4445" />
    <Router RName="Router" RPort="4444" RAddress="bach" />
  </AgentInfo>

  <Attribute Name="Transition" ID="40" Type="VARIABLE" VType="NUMBER">
    <Position posX="68" posY="292" />
  </Attribute>

  <Agent Name="Auction" ID="44" TIMEOUT="10">
    <Position posX="203" posY="213" />
    <Expression Value="( ( = v[1] v[49] ) )" />
    <PostExpression Value="( ( = v[2] v[45] ) )" />
  </Agent>

  ...
  <Task Name="Signal" ID="46" START="F">
    <Position posX="449" posY="278" />
    <Expression Value="( 7 ( signal v[45] ) v[54] )" />
  </Task>

  ...
```

Fig. 5. K-File.

K-File은 파서 구현의 편의를 위해 Fig. 5와 같이 XML 형태로 저장된다

3.1.2 K-Code

기존 프로그램과의 연동을 위한 인터페이스 에이전트의 작성에는 K-Code가 사용된다. XML 형태로 저장되는 K-File과 달리 K-Code는 이진 컴퓨터 코드이다. 개별 에이전트는 K-File이 아닌 K-Code를 이용함으로써 기존 프로그램과의 연동이 가능하게 된다. 개발자는 K-Code 작성을 위해 제공하는 KCodeBase 클래스를 상속 받아 K-Code를 작성한다.

3.2 에이전트 군

에이전트의 가장 큰 장점 중 하나는 주어진 문제를 분해하여 해결할 수 있다는 점이다. 각각의 에이전트는 하나의 요소로 취급되어 주어진 문제의 점진적인 성장이나 유연한 확장을 가능케 한다[15]. 그러므로 이러한 에이전트 사용의 장점을 활용하기 위해 ADE에서는 JATLite의 제한적인 확장성에 대한 보완책을 지녀야 한다. ADE에서는 두 가지 방향으로 이러한 확장성을 뒷받침한다.

첫째, 하나의 개별 에이전트는 단지 한 가지 역할만을 수행할 수 있다. 이러한 방식은 개별 에이전트를 최대한 단순화하기 위함이며, 결과적으로 주어진 문제를 최대한 분해하게 한다.

둘째, ADE는 Fig. 6에서 볼 수 있듯이 계층화된 에이전트 구성을 가진다. 개별 에이전트들은 전체적으로 에이전트 군을 형성함으로써 복합적인 기능을 수행하게 되며, 유지 보수 및 구현의 편의성을 위해

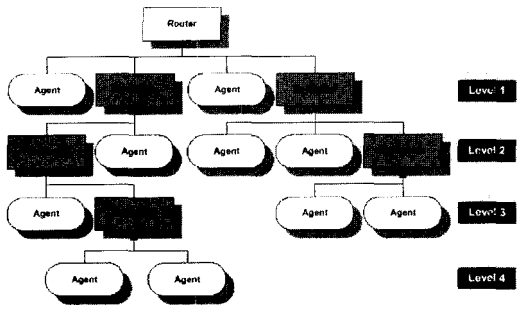

Fig. 6. Hierarchical structure : securing system's extensibility.

개별 에이전트들은 최대한 단순한 형태를 띄게 된다. 이러한 계층화를 위해 특별히 En-Router(Enhanced Router)가 제공된다. En-Router는 시스템 내의 En-Router들에 의해서는 개별 에이전트로 인식되며, 그 자신은 라우터의 기능을 담당하게 된다.

3.3 에이전트의 기술

ADE에서의 개별 에이전트는 디자인 로드맵 이론의 IPU로 매핑 가능하며, 개별 에이전트의 기술은 디자인 로드맵 이론에 바탕을 두고 확장/변경되었다.

3.1절에서 언급한 바와 같이 에이전트의 지식 파일은 K-File과 K-Code 형태가 존재하며 이 중 K-Code는 특별히 만들어진 자바 클래스를 상속 받아 자바 언어를 이용하여 작성한다. 그러므로 본 절에서는 K-File를 이용한 에이전트의 기술만을 다룬다. Fig. 8, Fig. 9, Fig. 10 그리고 Fig. 11에서 전형적인 K-File의 형태를 볼 수 있다.

3.3.1 K-File에서 사용되는 아이템

Fig. 7에서 보듯이 K-File에서는 4가지 형태의 아이템이 사용되며, 이 아이템을 사용하여 프로세스를 표현한다.

- **태스크 노드와 에이전트 노드** : 태스크 노드와 에이전트 노드는 기능이 할당되어 있는 작업 수행을 의미하는 요소이다. ADE에서 태스크는 사각형으로, 에이전트는 둥근 사각형으로 표시되며, 연결된 어트리뷰트 노드를 각각 입력과 출력으로 사용한다. 비록 표시되는 형태는 다르나 ADE에서는 에이전트와 태스크가 동일하게 취급된다. 디자인 로드맵 이론에서 IPU는 자신의 기능에 대한 설명과 기능에 수행에 필요한 자원에 대한 요구 사항을 가지는 정의된 작업을 처리하는 최소 단위로 정의된다. 이에 하나의 개별 에이전트는 하나의 IPU로 매핑이 가능하며 이는 곧 태스크 노드로의 매핑을 의미한다.
- **링크** : 링크는 일반적인 작업의 순서를 나타내는 표현 요소로의 기능을 가지고 있다. 즉, 링크는 태스크와 어트리뷰트 노드 간 연결을 하여 작업 흐름을 표현한다. 이러한 링크의 속성에 따라 링크에는 작업 흐름을 제어하기 위한 요소를 정의할 수 있다.
- **어트리뷰트 노드** : 어트리뷰트 노드는 태스크 노드를 수행하기 위한 입력이나 태스크 노드에서 출력된 결과를 표현하는 요소이다. 이와 함께 "Const" 설정을 통해 상수 값을 지정할 수 있다. Fig. 7에서 볼 수 있듯 어트리뷰트 노드는 세가지 형태가 있다. 일반형의 경우 태스크 노드를 수행하기 위한 입력이나 태스크 노드에서 출력된 결과를 표현하는 요소이며, "IO Input"과 "IO Output" 형태는 각각 에이전트의 시작과 끝을 담당하게 된다. 즉, 디자인 로드맵 이론에서 밝힌 IPU 간 연결을 위한 소켓 역할을 수행하게 된다.

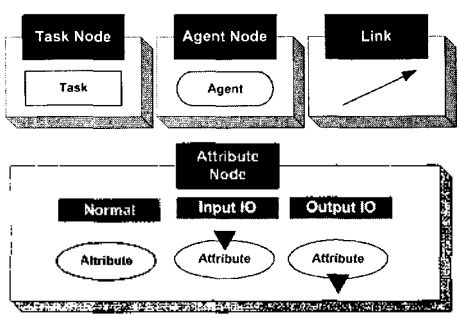

Fig. 7. Items appeared in K-File.

3.3.2 K-File/K-Code의 표현 범위

지식은 명시적인 지식과 암묵적인 지식이 존재한다.

$$\text{Knowledge} = \text{Explicit K} \cup \text{Implicit K} \quad (1)$$

식 (1)의 명시적인 지식은 식 (2)와 같이 표현할 수 있다.

$$\text{Explicit Knowledge} = \text{Process} \cup \text{K} - \text{Code} \quad (2)$$

식 (2)의 프로세스는 식 (3)과 같이 표현할 수 있다.

$$\text{Process} = \text{K} - \text{File} \cup (\text{Process} \cap \text{K} - \text{Code}) \quad (3)$$

그리고 K-File과 K-Code의 관계는 식 (4)와 같다.

$$\text{K} - \text{File} \cap \text{K} - \text{Code} = \emptyset \quad (4)$$

식 (1)에서 식 (4)는 모든 명시적인 지식은 K-Code와 K-File로 표현 가능하다는 것을 보여 준다. 즉,

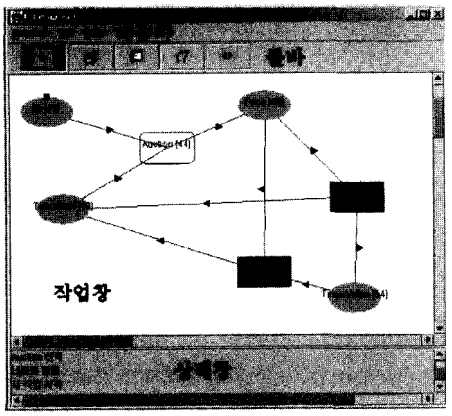

Fig. 8. K-Composer.

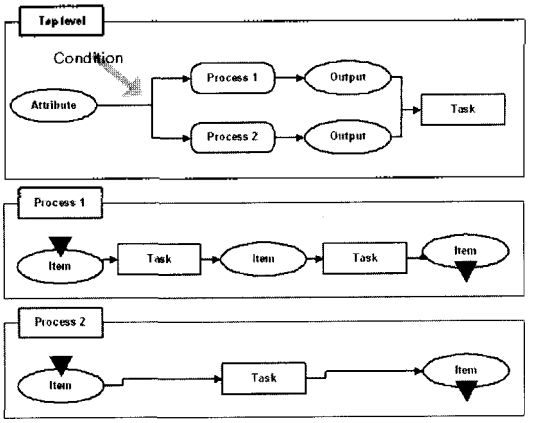

Fig. 9. Dynamic modification of process.

ADE에서의 에이전트는 명시적인 모든 지식을 표현할 수가 있다.

3.4 K-Composer

개발자는 K-Composer를 통해 태스크 노드, 어트리뷰트 노드, 에이전트 노드 그리고 링크를 적절히 배치함으로써 개별 에이전트를 작성할 수 있다. K-File의 작성 시 다른 에이전트를 에이전트 노드로 입력할 수 있어야 한다. 즉, K-Composer에서는 다른 에이전트의 정보를 사용할 수 있어야 하며 이런 기능을 위해 K-Composer는 ADE 내에서 하나의 에이전트로 취급된다.

GUI는 다음과 같은 기본 요소로 구성되어 있다.

- **메뉴바** : 사용자는 메뉴바를 통해서 모든 명령을 사용할 수 있다.
- **툴바** : 입력 모드를 선택한다. 입력 모드는 "선택", "태스크 노드", "어트리뷰트 노드", "에이전트 노드" 그리고 "링크"가 있다.
- **상태창** : 시스템은 상태창을 통해 사용자에게 필요한 정보를 제공한다. 표시되는 메세지는 오류, 라우터와의 통신 메세지 등이 있다.
- **작업창** : 에이전트의 구성을 표시한다. 개발자는 태스크 노드, 에이전트 노드, 어트리뷰트 노드, 링크를 추가, 삭제, 수정할 수 있으며 이들 관계를 설정하게 된다.

4. 효 과

4.1 동적 프로세스 변경

실행 시 프로세스를 동적으로 변경할 수 있다면 이를 통해 프로세스의 적응성을 높일 수 있을 것이다. 프로세스 적응성이란 프로세스를 수행하는 도중에 외부 환경 변화 또는 내부 업무 결과에 따라 이미 계획된 업무 프로세스를 변경할 수 있는 기능을 말한다.

ADE에서는 에이전트, 서브 프로세스 그리고 태스크를 개념적으로 동일시 함으로 인해 프로세스의 동적인 변경을 원활하게 한다.

Fig. 9는 "프로세스 1"과 "프로세스 2"라는 에이전트를 포함한 에이전트이다. "프로세스 1"과 "프로세스 2"는 동시에 수행되는 것이 아니라 그림에 표시된 Attribute 값에 따라 취사적으로 선택된다. 즉, 전체 프로세스는 환경에 따라 프로세스를 동적으로 변경한다.

4.2 지식의 에이전트 화

C 중전기 회사의 경우 사내에는 다양한 지식이 존재한다. 예를 들어 주어진 조건(VCB의 경우 정격용량, 차단 용량, 타입)을 만족시키는 부품이 다수 존재할 경우 부품의 선정 기준이 곧 설계자의 지식이다. 다른 예로, 캐비닛에 부품들을 위치시킬 경우 부품 간의 간섭(전기장, 자기장 등)으로 인한 조건이 발생하나 이런 조건들의 경우 수식화 내지는 수치화된 것이 존재하지 않는 설계자만의 지식이다. 그러나 사내 구성원들은 자신의 지식을 표현하는데 인색하다. C 중전기 회사의 또 다른 문제는 설계자들이 컴퓨터 프로그램에 익숙하지 않다는 점이다.

ADE는 이런 지식의 획득이라는 문제에 대해 다음과 같이 대응한다.

1. 해당 설계자의 지식을 에이전트화 한다. 회사는 에이전트를 가짐으로써 설계자의 지식을 보유할 수 있게 된다.
2. ADE에서의 에이전트는 곧 지식이다. 게시판과 같은 전통적인 형태의 지식 표현이 아니라, 사내 구성원들의 지식을 소프트웨어 모듈인 에이전트

로 직접적으로 표현하게 한다.

3. 에이전트의 개별화는 지식 파일인 K-File과 K-Code를 통해 이루어 진다. 이 중 K-File은 GUI 도구인 K-Composer를 통해 프로세스를 정의하는 과정 만으로 작성할 수 있다. 이런 특징은 에이전트 개발자의 저변을 확대시켜 보다 많은 지식을 에이전트화할 수 있게 한다.

4.3 개별 에이전트는 한가지 역할만 수행

하나의 개별 에이전트는 단지 한 가지 역할만을 수행할 수 있다. 즉, 하나의 개별 에이전트가 그리퍼의 토크를 계산하고, 해당 데이터 베이스로부터 원하는 데이터를 추출하는 것과 같은 복수의 기능을 가지지 못한다. 이러한 특징은 개별 에이전트의 단순화를 위함이다. 개별 에이전트를 최대한 단순화하는 것은 개별 에이전트 구현을 쉽게 하기 위함에 더하여 전체 시스템을 보다 강건하게 만들며 개별 에이전트들의 재사용성을 증가시키는 효과를 가져온다.

Fig. 10은 개별 에이전트를 단순화함으로써 시스템이 보다 강건해지는 일례이다. 부품 선정 에이전트가 통합된 기능을 수행하는 환경에서는 해당 에이전트가 동작하지 않을 경우 관련된 작업의 수행은 불가능하다. 이에 비해 부품 선정 에이전트 군을 이루는 방식은 이런 위험을 최소화하게 한다.

앞서 언급한 바와 같이 단순화된 에이전트는 재사용성이 증가한다.

에이전트의 범용성 ∝ 에이전트의 단순성

환언하자면, 에이전트의 단순성이 줄어드는 만큼 범용성이 떨어지게 되며 이는 곧 해당 에이전트의 재사용성을 감소시킨다. 에이전트가 단순하지 않을 경우

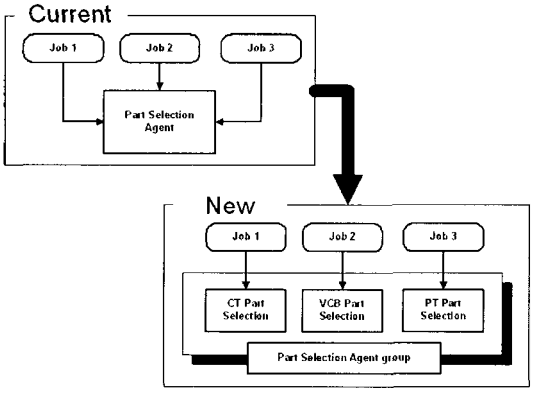

Fig. 10. Simplification of individual agent : improvement of robustness.

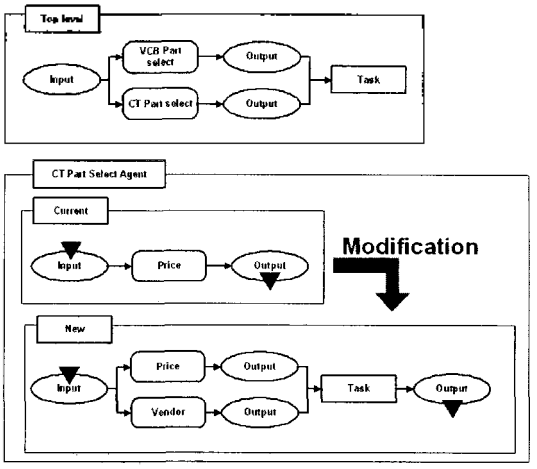

Fig. 11. System maintenance without difficulty.

해당 에이전트의 재사용성이 저해될 위험이 존재한다.

4.4 시스템의 지능화

자동화란 현재의 사내 지식을 시스템에 이식하는 과정이다. 사내 지식의 경우 시간이 지나면서 변화하기 마련이며, 변화된 지식을 시스템에 반영하기 위해서는 시스템의 업그레이드가 필수적이다. 이에 시스템의 유지/보수는 자동화 프로그램의 중요한 요소로 작용한다.

ADE는 시스템의 유지/보수에 탁월한 성능을 제공한다. ADE에서 개발되는 에이전트는 레벨을 가진다. 최상위 모델에서는 하위 에이전트가 추상화 되어 표현되기 때문에, 하위 레벨만을 변경함으로 시스템을 유지 및 보수할 수 있다. Fig. 11는 이런 예이다.

현재 시스템에서의 CT 부품의 선정은 단순히 가격만을 기준으로 이루어 진다. 그러나 환경 변화에 따라 가격과 함께 특정 공급업체의 제품을 기준으로 하여 CT 부품을 선정하게 시스템의 변경이 요구되었다. 이런 상황에서 ADE는 전체 시스템의 보수가 아니라 CT 부품 선정 에이전트의 프로세스만을 변경함으로 간단하게 상기 목적을 달성할 수 있다.

ADE에서의 에이전트란, 프로세스를 이용하여 지식을 표현한 것이다. 즉, 사내 지식의 변경이 있을 경우 IT 시스템을 개발한 업체에 의뢰하여 변경된 지식을 IT 시스템에 반영하는 것이 아니라 사내의 개별 작업자가 자신의 지식을 즉시 시스템에 반영할 수 있다.

5. 결론 및 향후 연구 과제

5.1 논의

본 연구에서는 에이전트를 활용하여 설계 및 생산

에 있어서 지능적이고 효율적인 환경 구축을 위한 방안으로 ADE의 개발에 대해 다루었다. 요약하자면, 새로운 에이전트 개념 정립하였으며, 에이전트 개발 방법을 고안하였고 이를 바탕으로 에이전트 작성 도구 개발하였다.

과거 에이전트에 관한 많은 정의가 논의되었다. 본 연구에서의 에이전트는 스탠포드 대학교에서 내린 정의에 기초를 두고 있다. 그러나 이에 더하여 ADE에서는 새로운 에이전트 개념을 추가하였다. ADE에서의 에이전트는 개체와 지식 파일인 K-File/K-Code로 구성된다. 개체와 지식 파일을 분리하여 에이전트를 파악함으로써 지식의 에이전트화라는 새로운 방법론을 제시할 수 있었다. 에이전트 개발자는 자신의 지식을 프로세스로 표현함으로써 지식 파일의 일종인 K-File을 작성할 수도 있고, 직접 자바 프로그램을 작성하는 방법으로 또 다른 형태의 지식 파일인 K-Code를 작성할 수도 있다. 작성된 지식 파일은 에이전트 개체인 엔진을 통하여 해석되어 지며, 에이전트 개체는 전체적으로 개발의 통일성을 가져온다. ADE의 에이전트의 또 다른 특징으로는 하나의 에이전트가 한가지 역할만을 수행하도록 할당된 작업을 에이전트화할 때 주어진 작업을 최대한 분해하여 고려토록 하고 있다. 마지막으로 에이전트 개발의 저변 확대를 위해 K-Composer라는 K-File 작성GUI 도구를 제공한다.

이러한 특징들을 바탕으로 ADE는 시스템 통합에 있어 필수 요건인 다음과 같은 요소를 만족시킨다.

• 유지 보수 : 시스템의 최하 단위는 개별 에이전트이다. 개별 에이전트는 단지 한 가지 기능만을 수행하며 이의 구현은 공통된 모듈인 에이전트 개체와 에이전트의 차별성을 주는 K-Code/K-File로 구성된다. 개발자는 K-Code/K-File만을 수정함으로 새로운 요구사항을 만족시킬 수 있다. 또한 시스템에 추가적인 기능이 요구될 경우 개별 에이전트를 추가하는 것 만으로 전체 시스템을 유지하면서 확장이 가능하다.

• 사용상의 편의/구현의 용의 : 시스템은 개별 에이전트의 추가 내지는 수정 만으로 특성이 달라 진다. 개별 에이전트의 작성은 K-File 작성기를 이용한 K-File이나 K-Code를 생성함으로 가능하다. 이 중 K-Code는 개발자가 직접 컴퓨터 프로그램을 작성하여야 하나, 단순화된 입출력 규약으로 구현 상의 편의를 최대한 도모하였다.

• 플랫폼 독립성 : 에이전트 간의 통신은 TCP/IP를 활용한다. 표준 통신 방법인 TCP/IP를 사용함으로써 개발된 에이전트는 플랫폼에 제약을 받지 않게 된다. 또한 현재는 자바 언어를 사용하여 에이전트가 개발되었으나, 개별 에이전트의 차별화는 K-Code/K-File에 의해 달성됨으로 에이전트 개체만을 여타 프로그램 언어로 이식함으로써 시스템은 유지될 수 있다.

• 분산성 : 에이전트 간의 통신은 TCP/IP를 활용한다. 또한 개별 에이전트의 관리는 라우터를 통해 이루어 진다. 라우터의 제공과 소프트웨어 컴포넌트인 에이전트 간의 의사 소통이 TCP/IP를 통해 이루어 짐으로써 에이전트는 위치 투명성을 확보하게 되며, 네트워크 상의 어떠한 시스템에도 존재할 수 있다.

• 시스템 통합 : 본 시스템은 스탠포드 대학교의 JATLite를 기반으로 한다. 개별 에이전트들은 라우터의 기능을 가진 En-Router에 의해 일차적으로 각 군이 관리가 되며, 전체적으로 총괄 에이전트에 의해 En-Router들이 관리된다. 또한 에이전트 언어로써, 표준인 KQML, KIF를 사용한다.

ADE는 향후 다양한 분야에 널리 활용될 수 있으리라 예상된다. 수치화가 불가능한 사내 설계 지식을 모델링한 설계 가이드 시스템, 하드웨어 유지 보수를 위한 훈련용 시스템 그리고 금융 분야에서 대출 전 실시하는 개인 신용 조회 시스템의 개발 등에 적용할 경우 가시적인 효과를 얻을 수 있으리라 예상한다.

5.2 향후 연구 과제

향후 연구 과제로는 ADE의 표현력 향상, 전역 어트리뷰트의 도입, 컴포넌트 지원, 어트리뷰트 기능 향상, 온토로지(Ontology)에 대한 고려, 개별 지식 에이전트의 관리 시스템 구축 그리고 병렬 프로세스 처리 방법의 향상을 들 수 있다.

5.2.1 표현력의 향상

3.3.2항에서, 모든 명시적인 지식은 K-File과 K-Code를 통해 표현할 수 있으며 이는 곧 ADE에서 모든 명시적인 지식을 표현할 수 있다는 의미로 파악하였다. 그러나 K-Code의 경우 개발자가 직접 자바 프로그램을 하여야 함으로써 에이전트 개발의 문턱을 높이게 된다. 그러므로 에이전트의 개발을 촉진시키기 위해서는 K-File로 표현할 수 있는 지식이 확대되어야 한다.

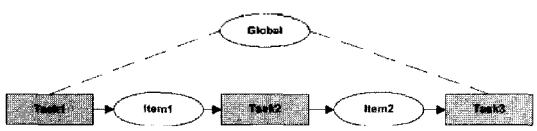

Fig. 12. Global attribute.

5.2.2 전역 어트리뷰트의 도입

ADE에서 태스크는 링크로 연결되어 있는 어트리뷰트만을 참조할 수 있다. Fig. 12와 같은 프로세스가 존재할 경우 태스크 3은 링크로 연결된 아이템 2의 값만을 참조할 수 있고 아이템 1은 참조할 수가 없다. 이러한 한계를 극복하기 위해 전역 어트리뷰트가 필요하다.

자바나 C++와 같은 프로그램 언어에서는 변수 영역이 존재하여 전역 변수와 지역 변수의 구별한다. 이와 마찬가지로 ADE에서도 전역 어트리뷰트의 존재가 필요하다.

5.2.3 병렬 프로세스 처리 방법의 향상

다중 프로세스가 존재할 경우 각각을 하나의 쓰레드로 처리하게 된다. 자바의 멀티 쓰레드를 활용할 경우 두 개의 프로세스는 각각 쓰레드 1과 쓰레드 2로 할당되어 개별 프로세스는 타 프로세스에 영향을 받지 않고 진행될 수 있다.

5.2.4 컴포넌트 지원

현재 ADE에서의 태스크는 사전에 정의된 기능, 즉 숫자의 사칙연산, 문자열의 연산 등과 같은 몇 가지 기능만을 수행한다. 이렇게 정의된 연산 외에 추가적인 기능은 에이전트를 구현함으로써 만족될 수 있다. 예를 들어, 숫자 목록에서 최대값을 찾는 기능이 필요할 경우 이런 기능을 구현한 에이전트를 만들고 이를 태스크에 대신하여 사용할 수 있다. 그러나 이런 기초적인 기능들을 모두 개별 에이전트로 구현할 경우 네트워크 상의 과부하를 초래할 수 있다. 그러므로 상기 예에서와 같은 기초적인 기능들은 ADE 엔진, 즉 에이전트 개체가 처리할 수 있도록 하는 것이 효과적이라 할 수 있다. K-Composer를 통해 기본적인 컴포넌트들이 제공되고 에이전트 개발자는 이를 태스크나 에이전트와 같이 K-Composer를 통해 입력할 수 있어야 한다. 작성된 K-File은 컴포넌트에 대한 태그들이 기술되어 있으며 에이전트 개체, 즉 ADE 엔진은 이를 해석하여 수행할 수 있어야 한다.

5.2.5 어트리뷰트 기능 향상

ADE에서 프로세스의 흐름 제어는 링크에 기술된 조건문에 의해 결정된다. 그러므로 현재와 같이, 어트리뷰트에 하나의 변수값 만을 저장할 수 있을 경우 조건문의 사용에 제약이 따른다. 그러므로 하나의 어트리뷰트는 복수의 변수값을 저장할 수 있어야 하며, 이는 곧 링크에 서술된 조건문의 표현력을 풍부하게 한다.

5.2.6 온토로지에 대한 고려

명확한 지식의 공유와 재사용을 위해서는 공유된 지식을 표현한 공통된 단어집(Vocabulary)를 정의하는 것이 필요하다[16]. 즉, 효과적인 지식의 공유를 위해서는 개념화의 명세(Specification of a Conception)로 정의[16]되는 온토로지에 대한 정의가 선행되어야 한다. 최근 주목받는 시맨틱 웹에서 웹 온토로지 언어(Web Ontology Language)[17]를 정의하는 것 또한 효과적인 지식의 재사용 및 공유가 하나의 요인이다. 본고에서는 하나의 동일 도메인 지식만을 가정하여 이에 대한 별도의 고려가 이루어 지지 않았다. ADE를 통해 보다 풍부한 지식의 공유 및 재사용을 이루기 위해서는 이에 대한 구현이 이루어 져야 한다.

5.2.6 개별 에이전트 관리 시스템 구축

ADE의 에이전트는 K-File/K-Code를 하나의 XML 파일 혹은 소프트웨어 모듈로 구성하고 있다. 이는 K-File/K-Code의 전송만으로 지식의 공유를 원활하게 하기 위한 고려이다. 그러나 이는 단지 콘텐츠의 작성만을 다루는 한계를 지닌다. 보다 원활한 지식의 공유를 위해서는 웹 서비스의 UDDI와 같은 옐로우 페이지 기능을 가진 K-File/K-Code 관리 시스템의 구축, 즉 개별 콘텐츠의 관리를 위한 시스템의 구축이 필요하다.

참고문헌

1. Heecheol Jeon, C. Petrie and M. R. Cutkosky, "JATLite: a Java agent infrastructure with message routing," *IEEE Internet Computing*, Vol. 4, No. 2, pp. 87-96, March-April, 2000.
2. Hisup Park, *Modeling of collaborative design process for agent-assisted product design*, PhD Thesis, Stanford University, 1994.
3. 최제민, 김봉제, 한상민, 연윤석, "에이전트 기반의 설계 구축을 위한 시험형 에이전트 통신 언어 번역기 개발," 한국 CAD/CAM 학술 발표회, 1998.
4. Weiming Shen and Douglas H. Norrie, "Agent-Based Systems for Intelligent Manufacturing : A State-of-the-Art Survey," *Knowledge and Information Systems, an International Journal*, Vol. 1, No. 2, pp. 129-156, May, 1999.
5. Shoham, Y., Agent-oriented Programming. Artificial Intelligence, Vol. 60, No. 1, pp. 51-92, March, 1993.
6. J. Hatvany, "Intelligece and Cooperation in Heterarchical Manufacturing Systems," *The 16th CIRP International Seminar on Manufacturing Systems Proceedings*, Tokyo, Japan, pp. 1-4, 1984.
7. A. D. Baker, "Chapter 8 - Metaphor or Reallity: A case study where agents bid with actual costs to

schedule a factory," *Market-beased Control*, Scott H. Clearwater(ed.), World Scientific, pp. 245-304, 1993.
8. Shen W.; Maturana F. and Norrie, D.H., "MetaMorph II: An Agent-Based Architecture for Distributed Intelligent Design and Manufacturing," *Journal of Intelligent Manufacturing - Special Issue on Distributed Manufacturing Systems*, Vol. 11, No. 3, pp. 237-251, May, 2000.
9. L. Stephens and M. Huhns, "Database Connectivity Using an Agent-based Mediator System," *Univ. of S. Carolina, Report A38*, 1999.
10. Deepika Chauhan., *JAFMAS:A Java-Based Agent Fram work for Multi-Agent Systems Development and Implementation*, PhD Thesis, University of Cincinnati, 1997.
11. Alan K. Galan, *JiVE : JAFMAS Integrated Visual Environment*, Master Thesis, University of Cincinnati, 2000.
12. AgentBuilder, http://www.agentbuilder.com.
13. "A brief introduction to the knowledge interchange format," http://www.cs.umbc.edu/kse/kif/kif101.shtml.
14. Genesereth, M. R. and Ketchpel, S. P., "Software Agents," *Communication of the ACM*, Vol. 37, No. 7, July, 1994.
15. Parunak and H. Van Dyke, "4 - Applications of Distributed Artificial Intelligence in Industry," *Foundations of Distributed Artificial Intelligence*, G. M.P. O'Hare and N. R. Jennings(ed.), John Wiley & Sons, pp. 139-163, 1996.
16. T. R. Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220, 1993.
17. Web-Ontology (WebOnt) Working Group, http://www.w3.org/2001/sw/WebOnt.

구 본 석

1999년 포항공과대학교 기계공학과 학사
2000년 Macroscape, Inc. 연구원
2001년 연세대학교 대학원 기계공학과 석사
2001년~(주)자이오넥스 선임 연구원
관심분야: 협업 설계, Neural Network

이 수 홍

1981년 서울대 기계공학과 학사
1983년 서울대 기계설계학과 석사
1991년 Stanford대학 Design Division Concurrent Engineering 전공, 박사
1991년~1992년 Lockheed Missile and Space Co. Cable Harness Design System개발 Post-Doc.
1983~1994년 KIMM CAD/CAM실, 선임 연구원
1994~현재 연세대학교 기계공학과 정교수
관심분야: 동시공학설계, 지식기반시스템 설계, DFM