KQML(Knowledge Query and Manipulation Language)을 이용한 에이전트의 상용화

이상열* · 정성호** · 황병곤***

1. 서 론

에이전트는 사용자가 원하는 일을 컴퓨터가 대신해서 자동적으로 해결할 수 있게 해 주는 소프트웨어이다. 사용자는 찾고자 하는 정보를 에이전트에게 맡겨놓고 다른 일을 할 수 있으며, 이때 에이전트는 필요한 정보를 찾아 사용자에게 맞게 분류, 정리, 요약을 한다.

최근에 컴퓨터와 통신 기술의 발전으로, 사용자가 처리하고자 하는 정보량도 급속하게 증가하여 분산 환경 하에서 정보를 처리하고 있다. 이러한 분산 네트워크 시스템에서 에이전트도 지역적으로 떨어진 서로 다른 에이전트의 도움을 받아 처리하는 분산 협동 처리의 개념으로 전환되고 있다[1,2].

분산 협동 처리를 위해서는 그림 1에서와 같이 서로 다른 에이전트간 통신이 필수적이다. 에이전트간 통신의 목적은 정보나 작업처리를 서로 공유하거나 교환하여 작업처리를 서로 분산시켜 컴퓨터 시스템의 부하를 줄이는데 있다. 즉, 모든 정보를 혼자 처리하는 완전한 에이전트를 개발하는 것은 불가능하므로 자신이 가지고 있지 않는 정보를 다른 에이전트에게 도움을 청하는 단계가 반드시

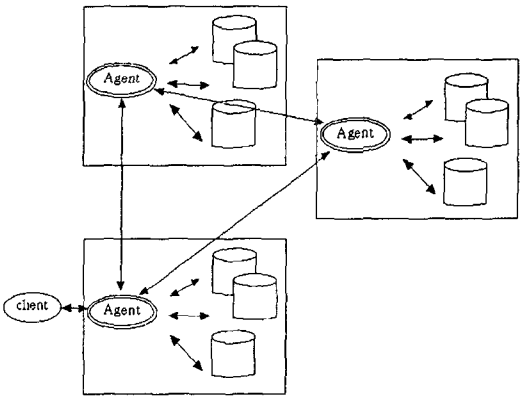


그림 1. 에이전트간 통신

필요하게 되는데, 이 도움을 청하는 방법으로 에이전트간 통신을 이용하는 것이다. 에이전트간 통신을 위한 가장 큰 걸림돌은 각 에이전트끼리의 호환성 결여이다. 즉, 각각의 에이전트는 서로 다른 사람에 의해, 서로 다른 플랫폼을 바탕으로, 서로 다른 목적으로 개발되었기 때문에 이들간의 통신을 위해서는 상호 호환 가능한 언어와 프로토콜이 필요하다. 이러한 요구를 충족시키기 위해 만들어진 언어가 KQML이라 할 수 있다[3].

2. 에이전트의 개요

2.1 에이전트의 정의

에이전트는 매우 광범위하게 사용되고 있어 그

*정회원, 대구대학교 박사과정
**정회원, 포항1대학 박사과정
***정회원, 대구대학교 컴퓨터정보공학부교수

정의를 내리기가 힘들다. 지금까지 많은 사람들이 정의하려는 노력을 하고 있으며, 여러 학자들의 공통된 부분을 정리해보면 에이전트는 "자동적으로 실행할 수 있는 능력과 추론을 수행할 수 있는 능력을 갖춘 것"이라고 할 수 있다. 이러한 에이전트로 전자우편 에이전트(E-Mail agent)를 예로 들어 수 있다[4]. 전자우편 에이전트의 목적은 사용자가 수 작업으로 해왔던 전자우편의 분류, 검색, 삭제, 전달 등의 작업을 대신해주는 것이며, 특성상 Unix나 Windows 등의 운영체제 환경 하에서 동작한다. 지식의 처리를 위해서 사용자의 요구사항이 조건적 규칙(conditional rule)의 형태로 지식베이스에 저장되고, 이 규칙의 조건 부분은 추론 메카니즘을 통해 전달된 메시지와 매치가 되는지 결정된다. 가령 'security'에 관한 전자우편이 도착했을 때 사용자가 현재 있는 곳으로 전달해 달라는 규칙이 있을 경우, 'security alert'라는 제목의 전자우편이 오면 조건이 만족되었으므로 사용자의 위치를 파악하여 전화나 팩스 등의 방법으로 메시지를 전달하게 된다. 전자우편 에이전트는 한 번의 수행으로 종료되는 것이 아니고 데몬(daemon) 처럼 항상 동작되면서 새로운 전자우편 메시지를 모니터링하게 되며, 전화번호 데이터베이스나 사용자의 일정 등과 같은 시스템의 다른 자원과도 메시지 교환을 통해 협력하게 된다.

2.2 에이전트의 특성

기본적으로 에이전트는 자율성(autonomy), 지능(intelligence), 이동성(mobility), 사교성(social ability)의 크게 4가지 특성을 갖는다.

자율성은 사용자나 다른 에이전트의 직접적인 지시나 간섭없이도 스스로 판단하여 행동하는 성질을 의미한다. 일반 프로그램이 수동적인 것에 비해 에이전트는 자율성을 가짐으로써 능동적으로 작업 수행을 진행하게 되며, 이를 위해 에이전트는 수행 동작이나 내부 상태 변화 등에 대한 제어권도 가지고 있다.

지능은 지식 베이스와 추론 능력을 갖추고 사용자의 의도를 파악하여 계획(planning)을 세우고 학습(learning)을 통하여 새로운 지식을 스스로 터득하는 성질로 인공지능에서 많이 연구된 결과를 근거로 한다. 지능은 사실상 자율성과 밀접한 관계를 가지게 되며, 지능을 바탕으로 에이전트는 같은 작업이라도 계획과 경험을 통해 더 나은 효과를 기대할 수 있다[5,8].

이동성은 사용자가 요구한 작업을 현재의 호스트에서 수행하지 않고 실제 그 작업을 처리하는 호스트로 이동시켜 수행함으로써 수행의 효율을 높이고 네트워크 부하를 줄이는 효과를 가져온다. 이동성은 특히 인터넷의 보급으로 컴퓨터 네트워크를 통해 제공되는 정보의 수가 급증하면서 그 중요성이 강조되고 있다.

사교성은 에이전트간 통신 능력을 의미한다. 즉, 하나의 에이전트로는 처리하지 못하는 작업의 수행을 위해 다른 에이전트의 도움을 필요로 할 때 에이전트간 메시지 교환에 의존하게 된다. 에이전트 통신 언어(agent communication language)를 이용한 에이전트간 통신은 메시지 전달(message passing)이나 공유 메모리(shared memory) 방법을 이용할 수도 있고 다른 에이전트의 메소드(method)를 불러 수행하기도 한다.

이러한 기본적인 특성 외에도 에이전트가 가지는 성질은 여러 가지가 있는데 그 중에서 환경 변화에 대하여 반응할 수 있는 반응성(reactivity), 틀린 정보를 주고받지 않는 정직성(veracity), 그리고 반드시 목적을 달성하는 방향으로 작업을 수행한다는 이성적 행동(rationality) 등이 있다. 물론 이러한 성질들을 모두 갖추어야 에이전트가 된다

는 것은 아니며 단지 이러한 성질을 많이 만족할 수록 좀 더 지능형 에이전트에 근접한다는 얘기가 될 수 있으며 그러한 방향으로 에이전트 연구와 개발이 진행되어야 할 것이다.

2.3 에이전트의 구조

하나의 에이전트는 기본적으로 에이전트 엔진, 영역 지식(domain knowledge), 통신 모듈 등으로 그림 2와 같이 구성된다.

에이전트 엔진은 에이전트의 생성과 작업수행, 에이전트 종료 등의 일련의 작업을 조정하기 위한 제어 지식과 추론 능력 등을 가진다.

영역 지식은 특정 응용 분야의 작업 수행에 필요한 지식으로서 에이전트의 역할을 특징지우는 부분이다. 에이전트는 생성시에 자신의 영역 지식과 관계된 작업 능력을 공개함으로써 다른 에이전트와의 작업 공유를 시도한다.

에이전트 통신은 다른 에이전트와의 메시지 교환을 담당하는데 대부분 자신이 해결하지 못하는 문제에 대하여 다른 에이전트의 도움을 청하는데

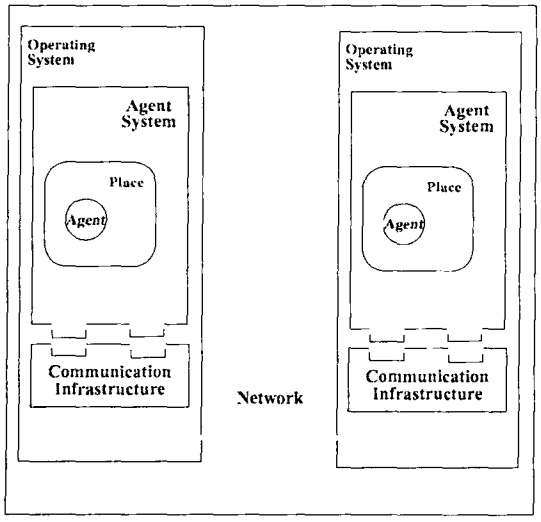


그림 2. 에이전트의 구조

많이 이용된다.

2.4 멀티 에이전트

에이전트 통신의 주된 목적이 다른 에이전트와의 협동을 통한 문제 해결이라고 할 때 몇 가지 문제점이 발생한다[6]. 즉, 다른 어떤 에이전트가 존재하는 지, 또는 다른 에이전트가 해결할 수 있는 능력이 무엇인지를 알고 있어야 한다는 것이다. 그렇지 않고는 누구에게 도움을 청할 지 판단할 수 없기 때문이다. 또 다른 문제점은 모든 에이전트끼리 서로 연결되어 있어야 하기 때문에 발생하는 네트워크 사용의 비용과 성능 저하를 들 수 있다. 이런 문제점을 해결하기 위한 접근 방향이 바로 멀티 에이전트 시스템이다.

멀티 에이전트 시스템은 하나의 에이전트로 문제점을 해결하지 못하여 다른 에이전트끼리 서로 협동을 하여 효과적으로 질의를 수행할 수 있도록 제안되었다. 이러한 멀티 에이전트 시스템을 구성하려면 우선 각 에이전트간의 프로토콜을 구성해야한다. 이러한 프로토콜의 구성을 위한 대표적인 언어가 KQML로 각 에이전트간의 통신을 위한 언어이다.

3. KQML (Knowledge Query and Manipulation Language)

서로 다른 에이전트간의 의사소통은 지능형 에이전트의 필수적 속성이다. 그 의사소통 언어는 에이전트들로 하여금 서로 효율적인 의사소통을 가능케 할 뿐 아니라, 서로 다른 하드웨어, 운영체제, 아키텍쳐, 프로그래밍 언어에도 불구하고 다른 에이전트와 지식을 공유할 수 있게끔 한다.

이러한 에이전트간의 의사소통을 위한 언어로서 ARPA KSE(Knowledge Sharing Effort)의 한

연구 그룹인 External Interface Working Group에 의해 만들어진 것이 KQML이다. KQML은 수행문(performative)이라는 요소로 구성되어 있는데, 이것은 에이전트가 교환하려고 하는 정보의 내용에 대한 지향적 태도를 표현하며, 그러한 태도를 다른 에이전트와 의사 소통을 할 수 있게 한다. 다음의 소개는 KQML의 구조와 문법들의 사용 방법이다.

3.1 KQML의 구조

KQML은 그림 3과 같이 내용 레이어(content layer), 메시지 레이어(message layer), 통신 레이어(communication layer)의 3개의 레이어 언어로 구성되어 있다[7].

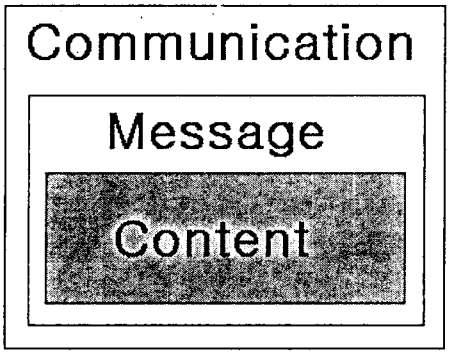


그림 3. KQML의 3 Layers

3.1.1 내용 레이어

KQML을 사용한 소프트웨어 에이전트는 내용(content) 메시지를 자신의 언어로 구성한 후, 이를 KQML 메시지 안에 싸서 보낸다. 내용 메시지는 ASCII, binary 등 어떤 표현 언어로 구성해도 상관없다. 내용 메시지는 KQML 구현과는 상관이 없고 단지 내용 메시지의 시작과 끝이 KQML의 관심 대상이다. 실제 이용되는 KQML 응용 시스템의 경우, KIF, LOOM, Prolog, KRSL 등 다양한 내용 언어(content language)를 이용할 수 있다. 심지어는 KQML 자신을 content language로 사용할 수 있다.

3.1.2 메시지 레이어

내용 레이어에 기술된 내용에 부가적인 정보를 포장하여 메시지를 생성해 내는 레이어이다. 흔히 화행 레이어(speech act layer)이라 부르며 추가되는 부가적인 정보는 주로 내용이 어떠한 종류의 화행을 띄고 전달되는가에 대한 것이다. 여기서 생성되는 메시지는 그 성격에 의해 크게 두 가지로 나뉘어지는데, 질의(query), 주장(assertion) 등과 같은 직접적인 지식 전달에 관련된 내용 메시지와, 직접적인 지식 전달은 아니나 이러한 활동에 관련된 메타 정보에 대한 선언(declaration) 메시지로 나뉘어 진다.

특히, 이 선언 메시지는 에이전트 사회에서 자신의 존재를 그 기능성과 함께 선언하거나 자신이 받고자 하는 어떠한 서비스에 대해 등록 등의 중요한 부분에 이용된다.

3.1.3 통신 레이어

KQML에서는 가장 바깥 계층으로 위에 언급된 내용 레이어, 메시지 레이어를 거친 메시지를 통신에 관련된 부가 정보를 추가하여 펙키지의 형태로 만든다. 여기에서 추가되는 정보는 sender, receiver와 메시지 ID 등이다.

3.2 KQML의 문법

KQML의 문법은 그림 4에서와 같이 <ASCII>, <alphabetic>, <numeric>, <double-quote>, <backslash>, <whitesp- ace>로 구성되어 있다. 구문의 대체적인 모습은 performative name와 parameter list로 이루어져 있다. 그리고 performative parameter는 그 parameter name으로

```
<performative> ::= (<word> {<whitespace>
        :<word> <whitespace> <expression>}*)
<expression> ::= <word>|<quotation>|<string>|
        (<word> {<whitespace> <expression>}*)
<word> ::= <character><character>*
<character> ::= <alphabetic>|<numeric>|<special>
<special> ::= <|>|=|+|-|*|/|&|^|~|_|@|$|%|:|.|!|?
<quotation> ::='<expression>|'<comma-expression>
<comma-expression> ::= <word>|<quotation>|
    <string>| ,<comma-expression>(<word>
    {<whitespace> <comma-expression>}*)
<string>::="<stringchar>*"|#<digit><digit>*
    "<ascii>*
<stringchar> ::=\<ascii>|<ascii>-\-<double-quote>
```

그림 4. KQML의 구문

구분될뿐, 그 순서와는 상관이 없다[9].

그림 5에서는 KQML이 사용하는 한 예로써 ask-if가 수행문이고 :content의 매개변수인 moter 값을 질의하는 예제이다. 그리고 수행문은 키워드 뒤에 오는 변수들에 의해 정의되어지며 그들의 키워드를 결정하고 수반하는 값의 의미를 말한다. 아래의 키워드들은 수행문에서 이미 KQML에서 예약되어진 것이다. 이 키워드와 정보 변수들의 의미는 표 1에 요약되어 있다.

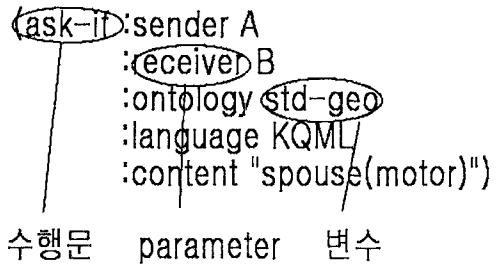


그림 5. KQML 메시지의 보기

기본적인 수행문들에 대해서 예를 들어 설명하기로 한다. KQML 사양은 그 구현에 있어서 개발자에게 많은 융통성을 제공하고 있지만 KQML의

표 1. KQML 변수의 정의

Keyword	의 미
:sender	수행문을 보내는 실제 송신인
:receiver	수행문를 받는 실제 수신인
:reply-with	회신을 원하는지를 표시한다.
:in-reply-to	어떤 내용의 회선인지 나타내는 label
:content	수행문이 나타내는 행위나 태도의 대상
:language	:content 부분에 가용된 표현 언어
:ontology	:content 부분에서 가정하는 ontology 이름
:force	송신측이 미래에 수행문의 의미를 부정할지의 여부

문법에서 수행문의 형식은 그림 5와 같은 형식을 갖추어야 한다. 수행문은 performative name 과 parameter list로 구성되어 있다. parameter는 키워드라고 하는 parameter로 이루어져 있다. 현재 KQML은 30여개의 수행문들로 구성되어 있으며 그 유형 및 종류는 그림 6에 나열되어 있다.

Basic query performatives
-evaluate, ask-if, ask-in, ask-one, ask-all
Multi-response
-stream-in, stream-all
Response performatives
-reply, sorry
Generic information performatives
-tell, achieve, cancel, untell, unachieve
Capability-definition performatives
-advertise, subscribe, monitor, import, export
Networking performatives
- register, unregister, forward, broadcast, route

그림 6. KQML performative

3.3 KQML의 적용 방법

다음의 수행문들은 한 에이전트가 다른 에이전

트에게 질의의 내용을 삽입할 수 있는 수행문이다. 사용하는 형식은 아래와 같다.

insert
:content <expression>
:language <word>
:ontology <word>
:reply-with <expression>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>

송신자는 수신자에게 :content 구문을 추가하라고 요청하고 수행문을 실행하여 결과값을 받아 문장이 성공적으로 구성되었거나 실패할 수 있는데 그 결과 값을 반환하여 질의한 에이전트에게 통보할 수 있다.

- 내용이 이미 질의한 구문에 있다.
- 내용이 질의의 구문과 모순된다.
- 송신자가 INSERT할 수 있는 권한이 없다.

delete
:content <performative>
:language KQML
:ontology <word>
:reply-with <expression>
:in-reply-to <expression>
:sender <word>
:receiver <word>

delete일 경우에도 마찬가지로 송신자는 수신자의 에이전트에게 구문을 없애기 위해 삭제하라고 요청한다. delete한 결과의 반환값을 송신자의 에이전트에게 돌려준다. 이는 다음과 같다.

- 내용이 삭제되지 않았다.
- 내용인 에이전트 구문에 없다.
- 송신자가 삭제할 수 있는 권한이 없다.

error
:in-reply-to <expression>
:sender <word>
:receiver <word>
:commet <string>
:code <integer>

error는 송신자가 보낸 수행문의 내용중에 :in-reply-to 변수에 의해 참조되어지는 메시지를 이해하거나 고려할 수 없다는 것을 가르킨다. :CODE 변수는 에러 형식을 분류할수 있는 return 값으로 수치 코드로 이루어 진다. :COMMENT 변수는 송신자가 잘못된 형식의 메시지를 어떻게 고려하는 가를 표현하는데 쓰여질 수 있다.

sorry
:in-reply-to <expression>
:sender <word>
:receiver <word>
:commet <string>

sorry는 수행문을 보낸 송신자가 이해했지만, :in-reply-to 변수에 의해 참조되어지는 메시지에 아무런 응답을 제공할 수 없다는 것을 가리킨다. 이러한 형식의 수행문은 evaluate나 ask-one 질의에 대한 응답으로 쓰여질 수 있다. 선택적 :COMMENT 변수는 주어진 응답이나 부가적인 응답에 대하여 거절하는 상황을 표현한 스트링을 전할 때 쓰여질 수 있다.

아래 그림 7은 에이전트 A에서 에이전트 B에게 수행문을 보내는 예이다.

A)는 에이전트 A가 에이전트 B에게 motors에서 자동차의 회전에 관한 정보를 구하는 예제이

```
A)
(evaluate :sender A
          :receiver B
          :language KQML
          :ontology motors
          :reply-with a1
          :content (val(torque motor1)
                   (sim-time 5)))
B)
(reply    :sender B
          :receiver A
          :language KQML
          :ontology motors
          :in-reply-to q1
          :content (scalar 12 kgf))
```

그림 7. 에이전트 A와 B의 KQML 예제

고, B)는 에이전트 B가 에이전트 A에게 회신하는 예제이다.

4. KQML 응용 사례

현재 KQML의 형식에 따른 에이전트가 많이 개발되고 있다. 그 중에서 대표적인 것은 UNIK-AGENT, LOOM and SIPE ARPA RLPI, Agent-base Software Integeration and ACL, PACT, Next-Link 등이 있다[12].

UNIK-AGENT(UNIK : UNIfied Knowledge)는 현재 KAIST 전자 상거래 연구실에서 prototype으로 개발이 완료된 전문가 시스템(Expert system) 기반 전자 상거래 시스템이다. 이 시스템에서는 판매자나 구매자를 위한 각 에이전트가 상거래를 위해 3개의 통신 레이어(communicator layer)를 통해 서로 통신을 하게 되는데, 이때 가장 상위 레이어인 ACL (Agent Communication Language)로 KQML을 사용한다. ACL이 운반하는 contents를 표현하는 언어로는 전자 상거래를 위한 고유 언어를 사용하는데, 이 부분이 제 2, 3 레이어인 전자 상거래 레이어, 제품 명세 레이어를 이루게 된다. 실제 UNIT-AGENT의 ACL 레이어는 KQML의 극히 일부 수행문만을 사용하고 있으나, KQML의 다양한 수행문을 수용함으로 좀 더 복잡한 작업을 수행할 수 있는 여지가 있다.

ARPA RLPI(Rome Lab Planning Initt- iative)에서 수행된 실험 중 하나로, SIPE란 planning 에이전트와 scheduler, LOOM으로 쓰여진 지식 베이스(Knowledge Base), reasoning tool 등을 결합하여 군 수송 전략을 위한 통합 계획 관리 시스템을 구축하였다. 기존에 이미 존재하던 각 시스템에 에이전트 기능을 추가함으로써 효과적인 협동적 분산 환경으로 통합시킨 특징을 가지고 있다.

Stanford에서 개발된, 에이전트 기반 통합 소프트웨어와 ACL은 복잡한 기능을 하는 촉진자를 이용하여 일반 소프트웨어 시스템을 위한 통합 시스템으로 KQML을 적용한다. ACL은 KQML의 변형으로, 새로운 표준으로 제시하고 있는 에이저트간 통신 언어이다. ACL은 지식 교환 포맷(Knowledge Interchange Format) 내부 표현 언어로 사용하여, 지금까지 여러 가지 대규모 소프트웨어 통합에 사용되어졌다.

PACT는 Stanford 대학과 Lockheed Palo Alto Research Labs, Hewlett-Packard, Enterprise Integration Technology 등이 함께 모여 구성된 연구 그룹이다. 이 그룹은 궁극적으로 다른 여러 사이트, 여러 서브시스템, 여러 전문 분야(multi-discipline)를 서로 협력하는 공학(concurrent engineering)의 기반 구조를 확보하는데 목표를 두고 있다. 이 환경에서 각 시스템들은 지식 기반 통신 언어와 서비스를 통해 서로 상호작용을 하게 된다. 이러한 PACT 구조는 서로 자율적으로 상호작용하는 에이전트에 기초를 두고 있는데, 이 상

호 작용은 서로 다른 분야간에 지식 교환을 위한 세가지 레벨의 개념, 어휘 체계를 공유 기반으로 두고 있다. 이때 첫 번째 레벨의 공통된 ACL로 이 KQML이 이용된다. 그외에 두 번째, 세 번째 레벨에서는 지식 교환 포멧와 Ontology 등이 이용된다. 이 PACT 프로젝트는 시나리오를 시연하기 위해 SHADE 프로젝트 일환으로 개발된 KAPI를 사용하였다.

Next-Link는 Design 연구센터에서 개발된 시스템으로 분산된 디자인과 이에 대한 필요를 연결시켜 주는 에이전트 조정 (coordination)에 대한 연구이다. 궁극적인 목적은 엔지니어들이 기존의 디자인들을 쉽게 이용 가능하게 하는 것이다. 그러나 실제 많은 디자인들은 호환성이 결여된 시스템과 툴들을 이용해서 이루어진 것들이기 때문에 이를 위해 이들 에이전트간에 존재하는 디자인, 엔지니어링에 대한 공통된 Ontology를 기반으로 해야 한다. 그래서 이들은 공통된 ACL을 통해 확보되는 조정 계층을 통해 Next-Link 구조를 구축하고 이를 조정 에이전트로 구축하는 방법을 택하고 있다. 이 Next-Link 구조에서 공통된 통신 인터페이스로는 SHADE의 KAPI를 이용하고 있다. 현재는 배선 케이블 제어와 관련해서 이 조정 에이전트를 개발 중에 있다[10,11].

5. 결 론

기존의 에이전트는 각각이 독립적으로 자신의 컴퓨터 시스템에 가장 적합하게 만들어 졌기 때문에 독립적으로는 많은 기능을 보유하고 있었어나, 서로 다른 하드웨어, 운영체제, 아키텍쳐, 프로그래밍 언어로 되어 있기 때문에 에이전트간의 통신에는 많은 불편함이 있었다. 그러므로 서로 비슷한 에이전트도 시스템 환경이 다르면 기능이 중복적으로 활용할 수밖에 없었다.

그러나 에이전트간의 통신 프로토콜인 KQML을 사용함으로써 서로 중복되지 않는 구현과 서로 다른 에이전트 간의 통신이 원할하게 된다. 한편, 분산 시스템에 기반 둔 분산 에이전트가 가능하게 되므로 시스템 활용도가 높게 된다.

앞으로 분산 환경 하에서의 지능형 에이전트의 개발로 사용자의 행위를 관찰하여 사용자의 취향을 추출하고 이를 이용하여 사용자에 적응할 수 있는 에이전트가 개발되기를 바란다.

참 고 문 헌

[1] Bird S., "Toward a taxonomy of multi-agent systems," Int. J. of Man-Machine Studies, Vol. 39, pp.689-704, 1993.

[2] Cohen P., Cheyer A., Wang M., and Baeg S., "An open agent architecture," Working Notes of AAAI Spring Symposium on Software Agents, pp.1-8, 1994.

[3] Finin T., Fritzson R., McKay D., and McEntire R., "KQML as an agent communication language," Proc. of CIKM '94, pp.126-130, 1994.

[4] Franklin S. and Graesser A., "Is it an agent, or just a program? : A taxonomy for autonomous agents," Proc. of Third International Workshop on Agent Theories, Architectures, and Languages. 1996.

[5] Guilfoyle C. and Warner E., Intelligent Agents: the New Revolution in Software, Ovum Limited, 1994.

[6] Singh M., Huhns M., Stephens L., "Declarative representations of multiagent systems," IEEE Trans. on Knowledge and Data Eng., Vol. 5, No. 5, pp.721-739, 1993.

[7] Genesereth M. and Ketchpel S., "Software agents," Comm. ACM, Vol. 37, No. 7, pp 48-53, 1994.

[8] Bigus, J. P., Bigus, J. 1998. Constructing Intel-

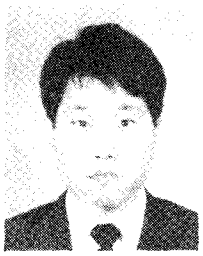
ligent Agents with Java, Wiley Computer Publishing.

[9] Labrou, Y. and Finin, T. 1997. A Proposal for a new KQML Specification, TR CS-97-03, February 1997, Computer Science and Electrical Engineering Department, University of Maryland.

[10] Charles J. Petrie et al., "Using Pereto Optimality to Coordinate Distribute Agent", AIEDAM special issue on conflict management Vol. 9, pp.269-281, 1995.

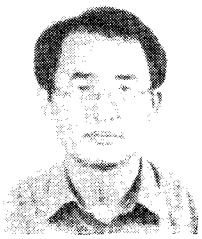
[11] http://cdr.stanford.edu/NextLink/NextLink.html

[12] http://www.cs.umbc.edu/kqml



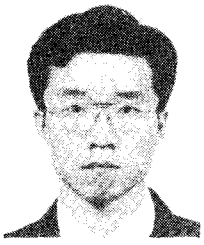
이 상 열

- 1992년 3월 대구대학교 전자계산학과 졸업(공학사)
- 1998년 6월 대구대학교 산업정보대학원 졸업(공학석사)
- 1998년 7월~현재 대구대학교 컴퓨터정보공학과(박사과정)
- 1992년 7월~1997년 9월 대구 태창정보통신(주)
- 1999년 3월~현재 성덕 대학 겸임 교수
- 관심분야 : 멀티미디어 정보검색, 인터넷 응용, 컴퓨터 그래픽스



황 병 곤

- 1974년 2월 경북대학교 전자공학과 졸업(공학사)
- 1980년 3월 경북대학교 전자공학과 졸업(공학석사)
- 1990년 2월 경북대학교 전자공학과 졸업(공학박사)
- 1975년 2월~1976년 2월 한국 기계금속 연구소 연구원
- 1976년 3월~1978년 7월 해군 통신 장교
- 1980년 9월~1982년 2월 한사 실업 전문대학 전임강사
- 1982년 3월~현재 대구대학교 컴퓨터 전보공학부 교수
- 관심분야 : 멀티미디어 정보검색, 인터넷 응용, 컴퓨터 그래픽스



정 성 호

- 1992년 2월 대구대학교 전자계산학과 졸업(공학사)
- 1994년 8월 대구대학교 컴퓨터정보공학과(공학석사)
- 1997년 3월~현재 대구대학교 박사과정
- 1998년 3월~현재 포항1대학 전산정보처리과 전임강사
- 관심분야 : 멀티미디어 정보검색, 인터넷 응용, 컴퓨터 그래픽스