DBpia

논문 00-25-4B-7 한국통신학회논문지 '00-4 Vol.25 No.4B

이동 에이전트 언어에서 위치개념을 이용한 이동성 확장

정회원 유 재 우*, 김 성 근*, 김 영 철*

Mobility Extension using Location Concept in Mobile Agent Languages

Chae-Woo Yoo*, Sung-Keun Kim*, Young-Chul Kim* *Regular Members*

요 약

이동 에이전트 언어에서 이동성은 중요한 문제중의 하나이다. 에이전트가 이동할 수 있는 범위와 크기는 에이전트가 사용자에게 서비스할 수 있는 일의 범위와 밀접한 연관을 가지고 있다. 본 논문에서 도입한 위치 개념은 이동 에이전트의 목적지를 논리적, 물리적, 에이전트 위치로 구분하여 정의함으로써 에이전트의 활동 범위를 다양화 할 수 있도록 하였다. 본 논문은 에이전트 언어에 위치개념을 지원하기 위해 에이전트의 생성, 삭제, 실행을 기술해 주는 자바클래스 및 기술된 프로그램을 네트워크에 연결된 컴퓨터에 구동시키는 에이전트의 서비스 시스템을 설계 구현하였다. 이러한 에이전트 언어에 위치개념의 도입은 에이전트의 이동성을 향상시키며, 이러한 에이전트는 사용자에게 보다 나은 서비스를 제공할 것으로 기대된다.

ABSTRACT

Mobility in the mobile agent language is an important issue. The scope range of the agent which can move is closely related with the work which the agent can do for users. The location concept proposed in this paper can make an activity scope of agent various by defining and classifying the destination of mobile agent as a logical and physical location. We designed and implemented a java class that describes creation, deletion, and execution of agent and a service system of agent that executes the described program on the computer which is linked to network to assist the location concept. The location concept will advance the mobility of agent and this agent is expected to provide much better service for users.

I. 서 론

현대 사회는 많은 부분에서 네트워크에 의존적인 형태를 취하고 있다. 네트워크 환경은 다양한 형태와 구조로 이루어져 있으며 수많은 방법으로 그 구조를 이용하고있다. 또한 네트워크를 이용하는 의존률도 계속적으로 높아지고 있는 추세이다. 하지만 현재의 네트워크는 포화 상태에 가까워지고 있으므로 좀 더 효율적인 체계가 요구된다. 현재의 클라이언트-서버 모델을 대신할 네트워크 체계로 에이전트 환경이 각광받고 있다.

원래 에이전트라는 것은 단순히 사용자나 관리자가 하는 반복 작업이나 복잡한 일 등을 대신 수행해 주는 형태로써, 각각의 에이전트마다 고유하고 특정한 역할이 주어진 상태로 개발되어졌다.

* 숭실대학교 컴퓨터학부
논문번호 : 99225-0603, 접수일자 : 1999년 6월 3일
※ 본 연구는 한국과학재단 핵심전문연구(과제번호:981-1212-036-2) 지원으로 수행되었습니다.

www.dbpia.co.kr

하지만 이와 같은 특수하고 전용적인 목적의 에이전트가 아닌 다양한 용도로 사용할 수 있는 에이전트를 생각해 볼 수 있다. 즉 전용적이고 특수 목적의 에이전트를 개발하는 대신 사용자가 특징적으로 표현하는 간단한 형태의 명령들이 에이전트의 형태로 실행될 수 있는 것을 말한다. 여기에 네트워크 환경을 추가한다면 간단한 형태의 명령들이 에이전트의 형태로 네트워크를 통해 다른 호스트 또는 기계들로 옮겨다니며 해당되는 기능을 수행하도록 에이전트를 만들고 이 에이전트들이 실행될 수 있는 시스템 환경을 제공하는 것이다[6].

에이전트 언어의 가장 큰 문제는 효율적인 이동성을 달성하는 것이다. 이동성은 여러 다른 플랫폼으로 이동할 수 있는 능력을 말한다. 에이전트의 이동 목적지의 다양화는 에이전트 프로그램의 효율성을 극대화할 수 있으며 이로 인해 사용자에게 보다 나은 서비스를 제공할 수 있다. 본 논문에서는 에이전트 이동의 목적지를 물리적 위치, 논리적 위치, 에이전트 위치로 구분하여 설계 구현함으로써 에이전트 이동성을 극대화하였다.

Ⅱ. 이동 에이전트 언어

이동 에이전트는 분산시스템 환경에서 통신 가능한 상대방에게 전달되는 코드가 포함된 객체를 말한다. 오직 실행가능하지 않은 데이터만 교환을 허용하는 시스템의 반대로서 이동 에이전트는 효율적인 성능과 기능을 달성할 수 있는 시스템이다.[1]

1. 이동 에이전트 언어

에이전트 언어는 에이전트 이론에서 제시된 여러 정형화된 개념을 이용하여 에이전트를 프로그래밍할 수 있는 시스템을 가리킨다. 에이전트 언어는 크게 에이전트 기술언어와 에이전트 프로그래밍 언어로 구분된다[11].

에이전트 기술 언어는 에이전트의 속성과 행동 등을 기술하는데 주로 에이전트가 수행할 수 있는 기능, 목적, 계획, 지식, 믿음 스키마(belief schema) 등의 형태로 에이전트를 표시하게 된다. 에이전트 프로그래밍 언어는 바로 수행 가능한 에이전트를 프로그램하기 위한 언어로써 대부분 해석 언어와 객체지향 기법을 사용하여 플랫폼에 독립적인 에이전트의 개발을 가능하게 한다.

이동 에이전트 프로그래밍 언어는 다음과 같은 방향으로 발전하는 경향이 있다. 첫째, 스크립트 언어적 성격을 가지는 컴파일 언어의 사용이다. 에이전트 공간을 돌아다니면서 임무를 수행하는 에이전트는 여러 기종의 호스트를 만나게 된다. 따라서 에이전트 코드, 즉 절차에 관한 지식은 특정 컴퓨터에서 컴파일된 이진 코드로 표현할 수 없고, 어디서나 통용될 수 있는 코드로 표현해 주어야 한다. 이러한 이유로 일반적으로 해석기(interpreter)에 의하여 해석되고 '대리' 수행된다. 이와 같이 해석되는 언어로는 Tcl과 Perl 등을 들 수 있다.

한편, 자바와 같이 해석이 가능한 중간 단계 표현 언어로 컴파일하는 형태의 스크립트 언어도 있다. 그러나 지금의 자바는 능동문서의 하나로서 제한된 조건에서 에이전트 일부 기능만 제공하는데 그치고 있다. 따라서 세계적으로 자바를 이동 에이전트의 기능을 첨가하여 확장하려는 연구가 진행되고 있다.

스크립트 언어가 가지는 문법 오류 검사나 의미 분석을 할 수 없는 단점을 극복하기 위해 컴파일 언어를 선호할 것으로 보인다. 또한 컴파일 언어는 스크립트 언어가 가지는 이동성의 장점을 살리기 위해 중간 단계의 표현 언어로 바꾸는 가상 머신을 사용하는 것이 주를 이룰 것으로 보인다.

둘째, 기존 언어의 확장을 통한 새로운 이동 에이전트 언어의 등장이다. 기존의 프로그래밍 언어의 사용자를 흡수할 수 있도록 기존 언어의 모습을 갖는 새로운 에이전트 언어가 주류를 이룰 것이다. 즉, 자바 언어는 C++ 언어와 흡사하지만 객체지향적 요소나 문법 등에서 많은 차이를 가진다.

본 논문은 이러한 발전추세에 맞추어 자바 언어의 확장, 즉 이동에이전트 중요한 요소인 이동성을 확장하기 위해 위치개념을 가지는 클래스를 구현하였다.

2. 이동에이전트 언어의 속성

에이전트 언어의 본질적 속성은 첫째, 코드가 포함된 객체들을 조작, 전송, 수신 그리고 실행을 지원해야한다. 둘째, 이기종 시스템을 지원해야 한다.

가. 코드 조작(Code Manipulation)

이동 에이전트의 생성과 사용은 코드 조작의 여러 종류를 지원하는 언어를 요구한다. 언어는 다른 에이전트와 프로그램 코드를 구분할 수 있도록 인식의 수단을 제공해야 한다. 예를 들면, 만일 에이전트가 다수의 함수로 구성되어 있다면, 하나의 지시하는 함수가 자신의 인지를 가지고 실행되어야 한다. 또한 프로그래머가 에이전트를 전송할 수 있

www.dbpia.co.kr

도록 표현할 수 있는 적당한 언어 프리미티브(primitive) 나 라이브러리 함수를 제공해야 한다. 에이전트의 수신과 실행(receipt and execution)을 위해 언어는 실행시간 환경에서 수신된 에이전트를 수신자의 주소 공간에서 실행할 수 있어야 한다.

나. 이기종 시스템 지원

분산시스템에서는 기계 구조나 운영체제의 이기종적 특성을 가진다. 인터넷과 같은 오픈 네트워크에서 사용자는 다수의 다른 기계 환경의 서비스를 접근해야 한다. 에이전트 기술이 널리 사용되기 위해서는 서로 다른 기계간의 에이전트의 이동 및 실행이 가능해야 한다. 이러한 이기종 시스템을 지원하는 연구가 많이 진행되었다.

다. 확장 이동성(Extend Mobility)

이동 에이전트의 중요한 이슈중의 하나가 이동성이다. 이동성은 여러 다른 하드웨어 플랫폼으로의 이동할 수 있는 능력을 말한다. 에이전트가 이동할 수 있는 범위의 크기는 에이전트가 사용자에게 서비스할 수 있는 범위와 밀접한 연관성을 가지고 있다.

라. 원격 자원 접근(Remote Resource Access)

에이전트는 실행 환경에서 지역 서비스 및 자원을 이용할 수 있어야 한다. 즉, 에이전트는 라이브러리 루틴, 시스템 데이터, 그리고 작업을 수행하는데 필요한 그 밖의 값을 이용할 수 있어야 한다. 에이전트 언어에서는 이러한 접근을 규정하고 통제할 수 있는 수단을 제공해야 한다. 프로그래머는 에이전트가 사용할 자원을 지시하기에 충분한 수단을 강구해야 한다.

마. 에이전트간 통신

에이전트는 플랫폼을 이동하면서 수행한 결과(지식)를 다른 에이전트에게 전달할 수 있다면 보다 효과적인 에이전트 시스템이 될 것이다. 이를 위해 에이전트 프로그래밍 언어는 통신을 위한 규약(protocol)을 규정하고 지시할 수 있는 프리미티브나 라이브러리가 요구되어진다.

이동 에이전트의 중요한 이슈중의 하나가 이동성이다. 이동성의 확장은 에이전트의 목적지를 논리적, 물리적 위치로 구분하여 에이전트의 이동하는 목적지를 구분하여 보다 효과적인 이동을 달성시키며, 이동성이 확장된 에이전트는 사용자에게 보다 나은 서비스를 제공할 것으로 기대된다.

Ⅲ. 이동성과 위치 개념

1. 이동성

이동성은 여러 다른 하드웨어 플랫폼으로의 이동할 수 있는 능력을 말한다. 이러한 이동성을 구현하는 방법은 원격실행(remote execution)과 이주(migration)의 방법이 있다. 원격 실행(약한 이동성)은 규정된 위치에서 새로운 에이전트를 실행하는 메카니즘이다. 반대로 이주(강한 이동성)는 에이전트가 다른 위치에 현재의 실행을 계속하는 메카니즘을 말한다[4].

가. 원격 실행

그림 1의 (a)는 원격실행을 묘사한 것이다. 원격실행은 단순한 메카니즘으로 에이전트가 실행되기를 바라는 위치로 전송된다. 이 메카니즘은 프로그램 코드를 전송하고 즉각적으로 새로운 에이전트가 생성되고 시작되어진다. 또 다른 방법은 새로운 에이전트가 원위치에서 생성되어지면 수행되기를 바라는 위치로 이주하여 수행되어지는 방법이 있다.

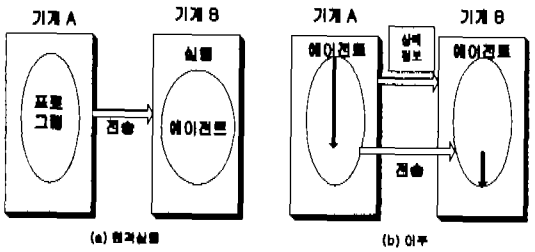

그림 1. 이동성의 두 가지 방법

나. 이주

이주는 에이전트의 현재 수행을 다른 위치에서 계속적으로 수행하는 메카니즘이다. 그림 1의 (b)는 이주를 묘사한 것이다. 이주는 에이전트에 의해 초기화되고 이주 명령으로 에이전트의 수행이 현재 위치에서 중단되고 다른 위치로 전송되고 있다. 에이전트가 가지는 상태정보를 이주가 필요하다. 상태정보는 프로그램 코드, 인스턴스 변수의 내용, 실행 상태 등이다.

2. 위치개념

여기서 우리는 이동 에이전트가 이동할 수 있는 위치(목적지)를 다음과 같이 정의할 수 있다[4].

가. 에이전트 위치

에이전트의 이주의 목적 주소가 다른 에이전트인 경우이다. 이것은 하나의 에이전트가 근접해 있는

www.dbpia.co.kr

다른 에이전트로 움직일 수 있다는 것이다. 그림 2는 에이전트가 이주할 수 있는 대상을 묘사하였다. 기계 A와 B, 기계 B와 C, D, 기계 C와 C, E, F는 에이전트를 의미한다. 이주하려고 하는 에이전트는 에이전트를 목적지로 이주할 수 있다. 즉, 기계 B의 C 에이전트는 기계 C의 특정한 에이전트에 C 에이전트로 이주하였다.

나. 추상화 위치

추상화 위치는 에이전트가 실행되는 "장소"로 표현된다. 이러한 경우에서 물리적 위치 투명성과는 구별된다. 일반적으로 추상화 위치는 여러 물리적 기계에 분산되어있다.

그림 2의 추상화 위치를 표현하는 A는 에이전트이다. 그러나 A는 기계 A와 기계 B에 존재할 수 있다. 즉, 이주하려고 하는 에이전트가 A에이전트로 이주할 때 실제는 기계 A나 기계 B로 이주하는 것을 말한다.

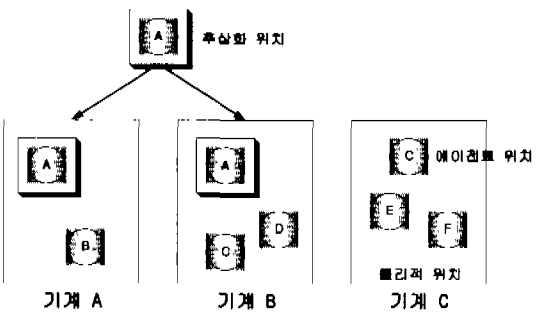

그림 2. 위치개념의 에이전트

다. 물리적 위치

물리적 위치는 역시 에이전트가 실행되는 "장소"로 표현된다. 추상화 위치의 반대 개념이다. 물리적 위치는 항상 위치가 실행되는 기계를 의미한다. 명백히, 물리적 위치 방법은 현대 분산 응용프로그램에 적합한 방법은 아니다. 그림 2에서의 물리적 위치는 기계 A, 기계 B, 기계 C를 의미한다. 위와 같이 에이전트가 이동할 수 있는 위치를 구분함으로써 사용자가 에이전트 프로그램을 작성 시에 추상화된 위치로 에이전트의 이동을 표현할 수 있으며, 이것은 분산환경에서 보다 효율적인 위치 투명성을 제공할 것으로 기대된다.

Ⅳ. 위치개념을 위한 클래스 설계 구현

위치개념을 가진 자바 언어를 구현하는데는 크게 두 가지 요소를 가진다. 첫째, 위치개념을 가지는 그림 3과 같은 클래스를 프로그래밍 시에 사용하여 에이전트의 생성 및 삭제, 실행 등을 기술해 주어야 하며, 둘째, 이렇게 기술된 프로그램은 네트워크에 연결된 컴퓨터에서 구동하는 서버와의 통신을 통하여 에이전트의 이동 및 정보교환, 에이전트 실행을 수행하게 된다.

1. 이동에이전트 언어의 클래스 상속 구조

본 논문에서 구현한 자바 클래스의 상속 구조는 그림 3과 같다. 다음 그림은 자바 클래스를 모아놓은 패키지로 구성되어있으며, 이 패키지는 구현된 다른 클래스에서 참조하고 있다.

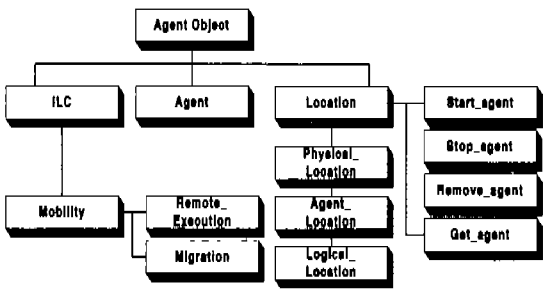

그림 3. 이동에이전트의 자바클래스 상속구조

Location 클래스는 에이전트의 실행되는 장소인 위치 서버(Location Server)를 정의하는 클래스이다. 위치 클래스는 에이전트의 실행,(start) 중지(stop), 제거(remove), 획득(get)의 함수를 정의하였다. 그림 4는 Location 클래스의 인터페이스이다. Location 클래스는 Physical_Location 클래스와 Agent_Loca-

```
public class Location extends AgentObject
        implements java.io.Serializable {
public String hostname;
public int port;
.....
public Location(String hname, int portno);
public void get_Agent(String agentObject);
public void start_Agent(String agentObject);
public void stop_Agent(String agentObject);
public void remove_Agent(String agentObject);
public native void rexec(String agentObject,
        String[] args);
.....
}
```

그림 4. Location 클래스의 인터페이스

www.dbpia.co.kr

tion 클래스 및 Logical_ Location 클래스의 상위클래스이다.

ILC 클래스는 에이전트간 통신 및 시스템 자원을 제공하는 에이전트 서비스 서버와의 연결을 담당하기 위해 정의된 클래스이다.

Mobility 클래스는 에이전트의 이동을 담당하는 클래스며, 두 가지 종류(원격실행, 이주)의 이동을 허용한다.

2. 이동에이전트 언어를 지원하는 시스템 하부구조

본 논문에서 구현한 이동에이전트 시스템 하부구조는 그림 5와 같다. 에이전트는 에이전트 서비스 서버에서 제공되는 서비스를 제공받으면서 위치 서버(Location Server) 위에서 실행된다. 위치 서버는 위치 개념을 지원하기 위해 다수의 에이전트를 수용하여 실행하도록 하였다. 에이전트 서비스 서버는 시스템의 자원을 접근 및 통제를 담당한다. 클래스 서버(Class Server)는 구현 시 필요한 다수의 서버를 통칭하였다.

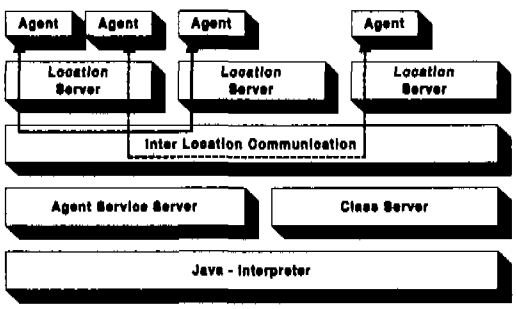

그림 5. 이동에이전트 시스템 하부구조

3. 이동성

본 시스템의 위치 서버는 원격 위치 서버의 자바 에이전트 객체의 실행을 요구할 수 있다. 또한 위치 서버는 에이전트 코드를 원격 위치 서버에 전달할 수 있으며 원격 위치 서버는 새로운 자바 에이전트 객체를 생성하여 실행한다. 이러한 방법은 자바의 RMI(Remote Method Invocation)방법 및 객체 전송을 위한 클래스의 사용으로 구현되었다.

이동성을 달성하기 위해 객체공간 접근을 위한 메소드는 다음과 같다.

- 네트워크를 통해 전달되거나 파일로 저장하기 위해 바이트 스트림(byte stream)으로 객체의 내용을 덤프하는 메소드(dumpObject)
- 바이트 스트림으로부터 바이트의 집합을 읽고 재 저장하기 위한 메소드(restoreObject)

위치 서버가 제공하는 각 에이전트 위치는 독특한 이름을 가진다. 이 위치 이름은 DNS (Domain Name Services) 이름을 이용한다. 응용프로그램에서 위치를 DNS를 사용하는 것은 물리적 위치 투명성을 제공하기 위한 방안이다.

그림 6은 쓰레드 이주를 지관하는 클래스의 인터페이스를 보여준다. MigrationThread 함수는 MigrationThread의 생성자이다. runSubclass는 에이전트를 해당 위치 서버에 전송하는데 사용하는 함수이며, resumeParent 함수는 이주시 정지된 에이전트 객체의 부모 객체를 실행하기 위한 클래스이다.

```
public class MigrationThread
 extends AgentThread
{
  private LocationName target_name;
  private Location location;
  private Thread parent_thread;
  public MigrationThread(AgentObject agent,
        LocationName dest_name,
        Location currentLocation,
        Thread thread);
  void runSubclass();
  protected synchronized void resumeParent();
}
```

그림 6. 쓰레드 이주를 위한 클래스

V. 결론 및 향후과제

분산 객체지향 시스템에서 이동 에이전트는 기존 패러다임에 도전하는 새로운 분야이다. 본 논문은 위치개념을 지원하기 위해 에이전트의 생성, 삭제, 실행을 기술해 주는 자바클래스 및 기술된 프로그램을 네트워크에 연결된 컴퓨터에 구동시키는 에이전트의 서비스 시스템을 설계 구현하였다. 위치개념은 에이전트의 목적지를 논리적, 물리적 위치로 구분하여 에이전트의 이동성을 향상시키며, 이러한 에이전트는 사용자에게 보다 나은 서비스를 제공할 것으로 기대된다. 향후과제로는 이동성의 두 가지 요소인 원격실행과 이주 중에서 아직 본 논문이 미진한 이주에 관한 보다 발전된 구현 및 에이전트간의 정보교환을 위한 높은 수준의 프로토콜(KQML)의 적용, 에이전트의 실행 시에 이루어지는 자원과 원격코드의 동적링킹(dynamic linking)에 관한 연구가 기대된다.

www.dbpia.co.kr

참 고 문 헌

[1] R. S. Gray. "Agent Tcl: A transportable agent system". *In Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore, Maryland, December 1995.

[2] M. Abadi, and L. Cardelli, "A theory of primitive objects: untyped and first-order systems". *Proc. Theoretical Aspects of Computer Software*. Springer-Verlag. 1994.

[3] A.D. Birrell, "An introduction to programming with threads". *In Systems Programming with Modula-3, Chapter 4*, G. Nelson, ed. Prentice Hall. 1991.

[4] M. Straer, J. Baumann and F. Hohl, "Mole - A Java based Mobile Agent System". In: M. Muhlhauser: (ed.), Special Issues in Object Oriented Programming. dpunkt Verlag 1997, pp. 301-308, 1997.

[5] F. Hohl, P. Klar and J. Baumann, "Efficient Code Migration for Modular Mobile Agents". *3rd ECOOP Workshop on Mobile Object Systems: Operating System support for Mobile Object Systems (MOS'97)*.

[6] E. Jul, H. Levy, N. Hutchinson, and A. Black, "Fine-grained mobility in the Emerald system". *ACM Transactions on Computer Systems* 6(1), 109-133. 1988.

[7] P. Agre and D. Chapman. "Perigi: An implementation of a theory of activity". *In Proceedings 6th AAAI*, pp. 268-272, 1987.

[8] R. A. Brooks. "A robust layered control system for a mobile robot". *IEEE Journal of Robotics and Automation*, 2(1), March 1986.

[9] A.D. Birrell, and B.J. Nelson. "Implementing Remote Procedure Calls", *ACM Transaction Computing System*, 2, pp. 39-59, 1994.

[10] L. Cardelli, "A Language with Distributed Scope", *Computing Systems*, 8(1), pp.27-59, 1995.

[11] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence*, 60(1), pp. 51-92, 1993.

유 재 우(Chae-Woo Yoo) 정회원

1976년 : 숭실대학교 전자계산학과(공학사)

1978년~1985년 : 한국과학기술원 전산학과 (공학 석사, 박사)

1986~1987, 1996~1997년 : Cornell Univ.와 Pittsburgh Univ.의 Visiting Scientist

1983년~현재 : 숭실대학교 전자계산학과 교수

<주관심 분야> 컴파일러, 프로그래밍 환경, 인간과 컴퓨터

김 성 근(Sung-Keun Kim) 정회원

1993년 : 숭실대학교 전자계산학과 졸업

1995년~현재 : 숭실대학교 컴퓨터학과 박사과정

1998년~현재 : 카톨릭상지대학 컴퓨터정보계열 교수

<주관심 분야> 프로그래밍 환경, 에이전트 프로그래밍 언어

김 영 철(Young-Chul Kim) 정회원

1990년 : 한남대학교 전자계산학과 졸업

1996~현재 : 숭실대학교 컴퓨터학과 박사과정

<주관심 분야> 컴파일러, 웹 프로그래밍 언어, 망 관리

www.dbpia.co.kr