KIPS Tr. Software and Data Eng.
Vol.4, No.3 pp.135~142 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.3.135

Cloudboard: A Cloud-Based Knowledge Sharing and Control System

Jaeho Lee† · Byung-Gi Choi†† · Jae-Hyeong Bae†††

ABSTRACT

As the importance of software to society has grown, more and more schools worldwide teach coding basics in the classroom. Despite the rapid spread of coding instruction in grade schools, experience in the classroom is certainly limited because there is a gap between the curriculum and the existing computing environment such as the mobile and cloud computing. We propose an approach to fill this gap by using a mobile environment and the robot on the cloud-based platform for effective teaching. In this paper, we propose an architecture called Cloudboard that enables knowledge sharing and collaboration among knowledge providers in the cloud-based robot platforms. We also describe five representative architectural patterns that are referenced and analyzed to design the Cloudboard architecture. Our early experimental results show that the Cloudboard can be effective in the development of collective robotic systems.

Keywords : Cloud, Blackboard, Knowledge Sharing, Framework

클라우드보드: 클라우드 기반 지식 공유 및 제어 시스템

이 재 호† · 최 병 기†† · 배 재 형†††

요 약

소프트웨어가 경쟁력의 핵심이 되는 소프트웨어 중심 사회로 이행되면서 프로그래밍 교육의 중요성이 새롭게 부각되고 있다. 세계적인 조기 코딩 교육 열풍도 이러한 추세를 반영하고 있으나 이를 지원하기 위한 교육 환경에서는 클라우드 컴퓨팅과 같은 새로운 컴퓨팅 환경을 효과적으로 활용함에 제약이 있는 것이 현실이다. 본 논문에서는 클라우드 환경에서 모바일 기기와 로봇을 이용하여 손쉽게 소프트웨어를 개발하고 프로그래밍 교육을 할 수 있는 클라우드 기반 지식 공유 및 제어 시스템인 클라우드보드(Cloudboard)를 제안한다. 특히 군집 로봇 시스템에서 개별 로봇의 센서 정보를 다수의 로봇이 공유하고 협업하여 공동 임무를 수행하도록 할 경우 클라우드보드 기능을 이용하여 손쉽게 프로그램을 개발하여 교육에 활용할 수 있음을 보인다. 클라우드보드의 기능은 기존의 대표적인 아키텍처 패턴을 비교 분석한 결과를 토대로 설계되었으며 실험적으로 효과와 성능을 검증한다.

키워드 : 클라우드, 블랙보드, 지식 공유, 프레임워크

1. 서 론

현대 사회는 정보화라는 제3의 물결에 접어들면서 기존의 체계와는 근본적으로 다른 새로운 국면으로 접어들고 있다. 1990년대부터 시작된 컴퓨터와 인터넷의 보급을 통해 ICT로 대표되는 정보화는 빠른 속도로 우리의 생활에 침투하고 있으며, 이미 대부분의 현대인들은 ICT 관련 장치의 도움 없이는 생활을 영위하기 힘든 현실에 맞닥뜨렸다. 이와 같은 환경의 변화는 단지 사람들이 이에 적응하여 생활하는 것에서 나아가 더 많은 비율의 ICT 기술자의 수요와 ICT 기술의 활용성이 제공되어야 함을 시사한다.

특히 프로그래밍 교육은 국가적, 사회적으로 빠르게 요구가 확산되고 있는 분야이며, 세계적인 추세를 살펴보아도 대부분의 국가가 경쟁력을 높이기 위해 추진하고 있는 분야이기도 하다. NIA의 보고서에 따르면 미국은 1996년 클린턴 대통령과 고어 부통령이 모든 학생들이 기술적인 문맹을 탈피하도록 하는 21세기 비전으로 Technology literacy challenge를 발표하였고, 또한 모든 교사와 학생들이 멀티미디어 컴퓨터를 가지고 온라인 학습을 할 수 있도록 학습 자료를 교육과정에 통합하였다. 일본은 2002년부터 일반계 고등학교 학생을 대상으로 컴퓨터과목인 정보A, 정보B, 정보C 세 과목을 신설하여 필수 선택과목으로 지정하고, 주당 2시간 이상을 이수해야 하며, 대학 입시에도 반영하는 정책을 시행하고 있다[1].

이와 같이 이미 전 세계적으로 프로그래밍 교육과정을 필수 과목으로 도입하고 있는 추세에 있어, 우리나라 또한 이러한 환경 및 사회적 변화에 발맞춰 프로그래밍 교육의 중요성을 인식하고 있으나 효과적인 커리큘럼 개발에 어려움

※ 이 논문은 2013년도 서울시립대학교 연구년교수 연구비에 의하여 연구되었음.
† 종신회원 : 서울시립대학교 전자전기컴퓨터공학부 교수
†† 비 회 원 : 서울시립대학교 전자전기컴퓨터공학과 박사과정
††† 비 회 원 : 서울시립대학교 전자전기컴퓨터공학과 석사과정
Manuscript Received : February 6, 2015
Accepted : February 12, 2015
* Corresponding Author : Jaeho Lee(jaeho@uos.ac.kr)

을 겪고 있다. 근래 이를 극복하기 위한 대안으로 로봇을 이용한 프로그래밍 교육에 대한 관심이 증대하고 있는바, 한국교육학술정보원에서는 이미 2007년 관련 연구보고를 통해, 새로운 교육 미디어로서의 지능형 로봇이 네트워크와 결합함으로써 기존의 e-러닝이나 u-러닝 시스템의 한계 극복뿐 아니라 미래 학교 현장의 교육 서비스의 질적 개선에 기여할 수 있을 것이라 보고 있다[2].

로봇을 이용한 교육은 이미 각 국가에서도 활용되고 있는 측면이 있으며, 감성발달, 지능교육 등의 용도를 위한 로봇은 이미 여러 형태로 개발되어왔다. 국내의 경우, 정보통신부의 IT839 전략 추진의 일환으로 지능형 로봇 개발 프로젝트가 추진 중이며, 일반 가정 및 유치원, 초등학교에 지능형 로봇을 보급하여 교육 및 오락 서비스를 시범적으로 제공하고 있다. 국외의 경우 자폐증 치료 및 장애 아동을 위한 로봇 활용에 관한 연구 및 쓰레기 재활용과 같은 생활지도에 로봇을 활용한 연구가 이루어지고 있으며, 휴머노이드 로봇을 교실 환경에 투입하여 학습자와 로봇 사이의 상호작용 관찰 및 교육효과에 관한 연구가 이루어지고 있다[2].

그러나 로봇을 이용한 교육에도 역시 한계가 존재하는데, 크게 3가지의 요인을 뽑아보자면 첫 번째로 저용량 및 저성능으로 인한 로봇의 메모리 한계로 다양한 교육 콘텐트의 제공과 콘텐트 누적이 불가능하다는 점이고 두 번째는 단순 작업을 반복하는 로봇의 행위가 사용자의 흥미를 저하시킬 수 있다는 점, 마지막 세 번째는 다양한 미디어의 활용이 부족하여 효과적인 로봇 활용이 불가능하다는 것이다.

위에서 언급한 문제를 해결하기 위한 클라우드 기반 로봇 학습시스템을 위해서는 클라우드 환경 내에서 통합 인터페이스를 제공하고 지식을 한데 모아 통합 관리할 뿐 아니라, 지능을 구현하는 각각의 요소들이 지식을 공유하고 협업할 수 있게 하는 시스템적 지원이 요구된다. 또한 Data-Driven적인 특성을 가지고 클라우드 상의 지식이나 상황에 변화가 있을 때에 비동기적으로 로봇에서 정보를 전달할 수 있어야 한다.

우리는 이러한 상황에서 사용할 수 있는 클라우드 기반 지식 공유 아키텍처를 제시함으로써 로봇이 지식의 제공자로서 클라우드 상의 지식을 공유하고, 다른 여러 로봇들이 같은 정보를 가지고 병렬적으로 행동할 수 있도록 하는 방법을 제안한다. 클라우드 기반 지식 공유 시스템은 클라우드 통합 인터페이스를 제공하여 로봇의 용량 및 성능의 한계를 극복할 수 있도록 하고, 클라우드 네트워크를 지원하는 로봇을 통해 클라우드 환경의 자원을 활용하여 개인화된 로봇의 용량 및 성능의 한계를 극복하고 네트워크 상에서 통합 관리할 수 있다. 또한 로봇의 지식 공유 및 다수 로봇 간의 협업을 위하여 로봇이 센서로 받아들인 정보는 클라우드 환경 내에서 공유되며 클라우드에 접속한 모든 로봇들이 이 정보들을 지식으로써 이용할 수 있게 된다.

우리는 이러한 문제를 해결하기 위한 방안으로서, 교육용 로봇 통합 프레임워크를 개발하기 위한 목표를 전제로 연구를 진행하였으며, 특히 논문에서 제안하는 것은 위의 한계적 측면 중에서도, 정보 공유를 위한 지식 공유 프레임워크를 개발하는 것을 목표로 삼고 있다.

2. 관련 연구

2.1 아키텍처 패턴 분석

기존에 사용된 소프트웨어의 아키텍처 패턴을 분석함으로써, 소프트웨어의 기본 설계를 결정하고 시스템 전반에 걸친 구조적 특성을 확인하며 서브시스템과의 관계를 조직화할 수 있다. 제안하는 클라우드보드는, 현재 많이 사용되고 효용성이 높은 대표적 소프트웨어 아키텍처들을 분석한 결과를 토대로 설계되었는바 이들 아키텍처들의 핵심 아이디어와 구조, 그리고 장점과 단점들 순서로 살펴볼 것이다.

1) 블랙보드 아키텍처

a) 개요 : 블랙보드 소프트웨어 아키텍처는 블랙보드라 불리는 공용의 지식베이스를 지식 소스(Knowledge Source)라 불리는 전문가들이 서로 공유하며 새로운 정보나 주어진 문제에 대한 부분 결과를 게시하여 종합적으로 문제를 해결하는 구조를 갖는다. 각 전문가들은 전체 임무 중 일부분을 해결하기 위한 특화된 프로그램으로 구성되고 각기 독립성을 유지하며 작업을 수행한다[3].

b) 구조 : 블랙보드 아키텍처는 블랙보드(blackboard)라는 공용 지식베이스, 지식 소스(knowledge source)들, 제어 컴포넌트(control component)로 구성된다. 블랙보드는 중앙 데이터 저장소 역할을 하며 모든 지식 소스들이 정보를 읽거나 기록할 수 있도록 하는 인터페이스를 제공한다.

지식 소스는 전체 문제 중에서 자신의 전문 영역에 해당되는 부분을 해결하는 독립적인 서브시스템으로서 이 서브시스템들이 모여 전체 문제 도메인을 구성한다. 이들은 단독으로는 시스템의 전체 문제를 해결할 수 없으며 서로 직접적으로 통신하는 대신에 블랙보드가 제공하는 읽기/쓰기 인터페이스만을 이용하여 정보를 교환한다.

제어 컴포넌트는 블랙보드에 일어난 변경사항들을 감시(monitor)하고 다음에 어떤 동작을 수행할지 결정하는 루프를 반복한다. 이 컴포넌트는 지식 소스들에 대한 응용 전략에 근거해서 언제 지식 소스를 평가(evaluation)하고 활성화(activation)할 것인지에 대한 스케줄을 잡는다. 이 전략은 블랙보드에 있는 데이터들에 근거해 수립된다.

c) 장점 :

- 서브시스템인 지식 소스들에게 문제를 분해하여 해결하게 할 수 있으며 서브시스템 간의 직접적인 상호작용 없이 서로 다른 알고리즘과 전략을 적용하여 독립적이며 병렬적으로 문제를 해결할 수 있다.
- 블랙보드 아키텍처는 각 지식 소스들, 제어 알고리즘, 중앙 데이터 구조가 엄격히 구별되어 독립적으로 존재하기 때문에 가변성과 유지보수성이 지원된다.
- 각 지식 소스는 특정 태스크에 대한 독립적인 전문가 역할을 할 수 있고, 블랙보드 아키텍처는 지식 소스를 재사용하는 것을 가능하게 한다.

d) 단점 :
- 시스템의 계산이 결정적인 알고리즘에 따라 수행되는 것이 아니기 때문에, 계산 결과가 동일하게 재현되지 않는 것이 일반적이다. 게다가 해법 프로세스 중에 잘못된 가설들이 포함되기도 한다.
- 제어 전략을 간단한 방식으로 설계할 수 없으며 실험적 방식을 사용하여야 한다.
- 올바르지 않은 가설들을 걸러내기 위한 계산에 과부하가 걸릴 수 있다.

2) 브로커 아키텍처

a) 개요 : 브로커 아키텍처 패턴은 애플리케이션 개발에서 불가피하게 발생되는 복잡성을 줄이기 위해 중개자(broker)를 도입하여 컴포넌트들을 분리하는 구조를 갖는다[4].

예를 들어 네트워크 프로그램의 클라이언트와 서버를 분리하기 위해 중개자로서 브로커 컴포넌트를 도입하면 서버들은 스스로를 브로커에 등록(register)하고, 메소드의 인터페이스를 통해 클라이언트가 서버의 서비스를 사용할 수 있도록 해준다. 클라이언트는 브로커를 통해 요청(request)을 보냄으로써 서버의 기능에 액세스 할 수 있다. 브로커의 태스크에는 적절한 서버를 찾는 작업, 클라이언트에 결과와 예외를 전송하는 작업이 포함된다.

브로커 시스템은 분산 기술과 객체 기술, 이 두 핵심 기술을 통합하는 역할을 할 수 있다. 또한 브로커 시스템은 단일 애플리케이션을 분리된 컴포넌트들로 구성된 분산 애플리케이션의 형태로 객체 모델을 확장할 수 있는 패턴이기도 하다. 확장된 이 분산 애플리케이션은 이기종 머신에서 실행되거나 각기 다른 프로그래밍 언어로도 작성될 수 있도록 설계 가능하다.

b) 구조 : 브로커 아키텍처 패턴은 클라이언트, 서버, 브로커, 브리지, 클라이언트 측 프록시, 서버 측 프록시, 이렇게 여섯 가징 유형의 컴포넌트들로 참여되어 구성된다.

서버는 오퍼레이션(operation)과 애트리뷰트(attribute)로 구성된 인터페이스들을 통해 기능을 제공하는 객체를 구현한다. 서버는 두 가지 목적에 따라 두 가지 종류로 나눌 수 있는데 하나는 여러 애플리케이션 도메인들에 공통 서비스들을 제공하는 서버이고, 또 다른 하나는 단일 애플리케이션 도메인이나 단일 태스크를 위한 특정 기능을 구현하는 서버이다.

클라이언트는 적어도 하나 이상의 서버에서 제공하는 서비스들에 액세스하는 애플리케이션이다. 클라이언트는 원격 서비스를 호출하기 위해 브로커에게 요청을 보낸다. 오퍼레이션이 실행된 이후 클라이언트는 브로커로부터 응답(response)이나 예외(exception)를 받는다.

브로커는 클라이언트에서 서버로 요청을 전송할 뿐만 아니라 그에 대한 응답과 예외를 다시 클라이언트로 전송하는 책임을 맡는다. 브로커는 고유의 시스템 식별자(identifier)에 근거해서 요청을 받을 수신자의 위치를 찾는 방법을 구현하고 있어야 한다. 브로커는 서버를 등록하는 오퍼레이션과 서버의 메소드를 호출하기 위한 오퍼레이션을 제공하는 API를 클라이언트와 서버에 제공한다.

클라이언트 측 프록시는 클라이언트와 브로커 간의 레이어 역할을 한다. 이 추가 레이어는 투명성(transparency)을 제공하는데, 이 덕분에 원격 객체는 클라이언트의 입장에서 보았을 때 로컬 객체로 간주된다.

서버 측 프록시는 클라이언트 측 프록시와 대체로 유사하나 차이가 있다면, 서버 측 프록시는 요청을 받고(receive) 들어온 메시지를 원래대로 풀고(unpack) 매개변수를 언마샬링(unmarshaling) 하고, 적절한 서비스를 호출한다는 것이다.

브리지는 두 브로커가 상호 운용적일 때 세부 구현을 숨기기 위해 사용되며 필요에 따라 선택하거나 선택하지 않을 수 있는 옵션 컴포넌트이다. 브로커 시스템이 이질적인 네트워크에서 실행되고 있다고 가정했을 때, 요청이 네트워크로 전송되면, 각 브로커들은 사용하고 있는 각기 다른 네트워크와 운영체제에 좌우되지 않고 독립적으로 통신해야 한다. 이때 브리지는 특정 시스템과 관련된 모든 세부 구현 내용을 캡슐화한 레이어 역할을 한다.

c) 장점 :
- 위치 투명성이 제공된다. 브로커가 고유 식별자를 사용해서 서버의 위치를 찾는 책임을 맡고 있기 때문에, 클라이언트는 서버가 어디에 있는지 알고 있을 필요가 없고 마찬가지로 서버 역시 자신을 호출한 클라이언트가 어디에 있는지 알 필요가 없다.
- 컴포넌트의 가변성과 확장성이 보장된다. 만약 서버가 변경되더라도 서버의 인터페이스만 동일하게 유지된다면, 클라이언트가 동작하는 데에 어떠한 영향도 미치지 않는다.
- 플랫폼 간의 이식가능성이 제공된다. 브로커 시스템은 API, 프록시, 브리지 등과 같은 우회 레이어를 사용함으로써 클라이언트와 서버로부터 운영체제와 네트워크 시스템을 숨긴다.
- 서로 다른 브로커 시스템들 간의 상호운용성이 지원된다. 서로 다른 브로커 시스템들일지라도 메시지를 교환하기 위해 공통 프로토콜만 맞춘다면 상호 운용될 수 있다. 이 프로토콜은 브리지에 의해 구현되고 처리된다.
- 재사용성이 확보된다. 새로운 클라이언트 애플리케이션을 만들 때, 기존에 있던 서비스의 애플리케이션 기능을 토대로 삼을 수 있다.

d) 단점 :
- 효율이 낮다. 브로커 아키텍처 패턴을 적용한 애플리케이션은 이식가능성, 유연성, 가변성을 발휘할 수 있도록 우회 레이어를 사용하기 때문에 대체로 느리다.
- 분산 소프트웨어 시스템이 아닌 시스템과 비교했을 때, 브로커 시스템은 낮은 장애 허용성을 제공하는 경향이 있다.

3) PAC 아키텍처

a) 개요 : Presentation-Abstraction-Control(PAC) 아키텍

처 패턴은 계층구조를 이룬 에이전트(agent)들이 서로 협력을 이루어 상호작용 소프트웨어 시스템의 구조를 형성한다. 모든 에이전트는 애플리케이션 기능의 특정 측면을 담당한다. 또한 각 에이전트는 프레젠테이션(presentation), 추상(abstraction), 컨트롤(control), 이렇게 세 가지 컴포넌트로 구성된다. 이렇게 구성함으로써 어떤 에이전트의 핵심 기능 및 그와 통신하는 다른 에이전트들을 사람-컴퓨터 간에 이루어지는 상호작용과 분리시킬 수 있게 된다[5].

b) 구조 : PAC 아키텍처 패턴에서는 에이전트들을 트리 형태 계층구조로 구성하여 상호작용 애플리케이션을 구축한다. 여기에는 오직 하나의 최상위 레벨 에이전트가 있으며, 그다음에는 몇 개의 중간 레벨 에이전트가, 아래에는 여러 개의 최하위 레벨 에이전트가 자리를 잡는다. 모든 에이전트는 애플리케이션 기능의 특정 측면을 담당한다.

이때 각 에이전트는 프레젠테이션 컴포넌트, 추상 컴포넌트, 제어 컴포넌트, 이렇게 세 가지 컴포넌트로 구성된다. 전체 계층구조는 에이전트들 간의 전이 종속성(transitive dependency) 관계에 따라 구성된다. 각 에이전트는 계층구조 상으로 최상위 에이전트에 이르기까지 자신보다 상위에 있는 모든 상위 레벨 에이전트들에게 종속된다.

최상위 레벨 PAC 에이전트(top-level PAC agent)는 시스템의 핵심 기능을 제공하며 다른 대부분 PAC 에이전트들은 이 핵심 기능에 의존하거나 이 기능을 처리한다. 또한 최상위 레벨 PAC 에이전트는 특정 서브태스크에는 할당될 수 없는 자체 사용자 인터페이스의 부분들을 포함한다.

최하위 레벨 PAC 에이전트(bottom-level PAC agent)들은 스프레드시트나 차트처럼 시스템의 사용자가 다루는 자체 독립적(self-contained) 의미 개념(semantic concept)들을 포함한다. 최하위 레벨 에이전트는 사용자에게 이런 의미적 개념을 제공하며, 사용자가 해당 에이전트에서 수행할 수 있는 모든 오퍼레이션을 지원한다.

중간 레벨 PAC 에이전트(intermediate-level PAC agent)들은 하위 레벨 에이전트를 조합하는 역할을 하거나 하위 레벨 에이전트들 사이의 관계를 연결하는 역할을 한다.

c) 장점 :
- 역할(concern)이 명확히 분리된다. 애플리케이션 도메인에서 각기 다른 의미적 개념(semantic concept)들은 독립적인 에이전트에 의해 표현된다. 각 에이전트는 자체 상태와 데이터를 저장하며, 다른 PAC 에이전트들과 함께 조정되더라도 독립성은 유지된다.
- 가변성과 확장성이 지원된다. PAC 에이전트의 프레젠테이션 컴포넌트나 추상 컴포넌트 내에서 변경이 일어나더라도 시스템에 있는 다른 에이전트들에 어떤 영향도 미치지 않는다. 그렇기 때문에 PAC 에이전트에 기반한 데이터 모델을 개별적으로 수정하고 조절할 수 있으며 그 에이전트의 사용자 인터페이스도 변경할 수 있다.
- 멀티태스킹이 지원된다. PAC 에이전트들은 서로 다른 스레드, 프로세스, 머신에 쉽게 분산시킬 수 있다.

d) 단점 :
- 애플리케이션 내의 모든 의미적 개념을 자체 PAC 에이전트로 구현하면, 시스템 구조가 복잡해진다. 또한 에이전트가 조정되고 제어되기 위해서는 조정(coordination) 에이전트가 추가로 필요하게 된다.
- 제어 컴포넌트가 복잡해진다. PAC 시스템에서 제어 컴포넌트들은 에이전트의 추상 부분과 프레젠테이션 부분 간의 통신을 매개하는 역할을 한다. 그러므로 제어 컴포넌트가 얼마나 잘 구현되는지는 에이전트들 간의 효과적인 협력에 의해 결정되며, 이것이 시스템 아키텍처 전체의 질까지 좌우한다.
- 효율성이 떨어진다. PAC 에이전트들 간의 통신에서 과부하가 발생하면 시스템 효율성에 치명적인 영향을 미친다.
- 적용가능성이 떨어진다. 애플리케이션의 원자적(atomic) 의미적 개념들이 작으면 작을수록, 사용자 인터페이스는 점점 더 유사해지며, 이 패턴의 적용가능성(applicability)은 점점 더 떨어진다.

4) 마이크로커널 아키텍처

a) 개요 : 마이크로커널 아키텍처는 지속적으로 변화하고 발전하는 요구사항을 소프트웨어에 적용하기 위하여 고안된 아키텍처이다. 애플리케이션의 핵심 컴포넌트를 마이크로커널 컴포넌트로 캡슐화하고 캡슐화 된 컴포넌트는 다른 컴포넌트들 간의 통신을 담당하고 파일이나 프로세스와 같은 시스템 자원을 저장한다. 또한 각 컴포넌트들이 자체 기능에 접근할 수 있는 인터페이스를 제공한다[6].

b) 구조 : 마이크로커널 패턴은 크게 내부서버(internal server), 외부서버(external server), 어댑터(adapter), 클라이언트(client), 마이크로커널(microkernel)의 총 5가지 컴포넌트로 구성된다.

마이크로커널은 핵심 컴포넌트로서 통신 기능과 리소스 핸들링과 같은 서비스를 제공하며 다른 컴포넌트들은 이 서비스의 전체 또는 일부를 구성한다.

내부서버는 마이크로커널의 서브시스템으로서, 마이크로커널이 제공하는 기본 기능을 확장하는 역할을 한다. 추가 서비스를 구현하거나 특정 시스템에 맞춰진 구현을 캡슐화 하며 마이크로커널이 요구하는 경우에만 활성화되거나 부름을 받는다.

외부서버는 마이크로커널 패턴에서 독립된 컴포넌트로서, 자체 클라이언트를 위한 인터페이스를 제공한다. 애플리케이션의 도메인에서 독립적인 뷰를 구현하기 위해 마이크로커널 컴포넌트를 사용하기도 한다.

클라이언트는 오직 외부서버 하나에만 연결된 하나의 애플리케이션이다. 그렇기 때문에 해당 외부서버가 제공하는 프로그래밍 인터페이스에만 접근할 수 있으며 이러한 이유들 때문에 클라이언트와 외부서버 간의 직접 종속성을 방지하기 위하여 서로 간의 인터페이스를 도입해야 하는 필요성이 있으며 이 역할을 어댑터가 맡는다.

어댑터는 클라이언트가 외부서버에 서비스를 요청할 때마다 적절한 서버로 호출을 전달하는 역할을 한다. 이를 위해 어댑터는 마이크로커널이 제공하는 통신 서비스를 사용한다.

c) 장점 :
- 다른 환경에 시스템을 이식할 때, 대부분의 경우 외부서버나 클라이언트 애플리케이션을 이식할 필요가 없어 이식성이 보장된다.
- 유연성과 교환가능성이 확보되고, 정책과 메커니즘을 엄격히 분리하여 전체 시스템의 유지보수성과 가변성을 향상시킬 수 있다.

d) 단점 :
- 성능의 향상이 어렵다. 특정 뷰를 제공하기 위하여 설계된 시스템에 비하여 각기 다른 뷰를 지원하는 마이크로커널 시스템은 성능 하락이 우려되기 때문에 유연성과 교환가능성이 필요한 시스템이 아니라면 고려해보아야 한다.
- 설계와 구현이 복잡해진다. 마이크로커널 컴포넌트가 제공해야 하는 서비스를 분석하거나 예측하기 쉽지 않고, 정책과 메커니즘을 분리하기 위해서는 시스템 도메인에 대한 높은 수준의 지식이 필요하기 때문에 요구사항 분석 및 설계에 상당한 어려움이 뒤따른다.

2.2 아키텍처 패턴 비교

여기서는 앞서 제시된 대표적인 아키텍처들을 비교하여 클라우드 기반 지식 공유 아키텍처와의 적합성을 가변성 및 확장성, 컴포넌트 재사용성, 효율성, 지식 재사용성, 이식성을 기준으로 비교한 결과를 Table 1과 같이 정리하였다.

Table 1. Comparison of Architecture Patterns

	블랙보드	브로커	PAC	마이크로커널
가변성/확장성	○	○	○	○
컴포넌트 재사용성	-	-	○	○
효율성	○	×	×	×
지식 재사용성	○	-	-	-
이식성	-	○	-	○

위에서 살펴본 바와 같이 블랙보드 아키텍처는 로봇이 지식제공자들이 지식을 공유하는 데에 있어서 유용한 특징을 가지고 있으나 클라우드와 같은 환경을 고려하여 개발된 된 아키텍처는 아니다. 이에 따라 본 논문에서는 블랙보드 아키텍처를 클라우드에 적용함으로써 로봇의 지식을 공유할 수 있도록 하는 새로운 아키텍처를 제시하며 이를 클라우드 보드(Cloudboard)라 부른다.

3. 연구 내용

3.1 연구배경

본 논문의 배경인 스마트 로봇 개발 사업의 목표는 로봇을 활용한 교육을 통해 사용자가 프로그래밍 교육 서비스를 더욱 흥미롭고 효과적으로 받을 수 있는 방법을 제공하기 위한 시스템적 지원체계를 갖추는 것에 있다. 현실적으로 교육 서비스를 제공하기 위한 소형 로봇은 다음과 같은 문제를 가지고 있다.

- 로봇의 공간 및 자원의 한계로 콘텐트가 한정적이다.
- 로봇의 행동이 단순하고 반복적으로 수행되어 흥미를 끌기 힘들다.
- 로봇이 미디어와 연동할 수 없어 교육 시나리오가 단조롭다.

이러한 문제는 로봇을 이용한 효과적인 교육을 저해하는 요소로 작용한다. 이와 같은 문제를 벗어나 로봇이 다양한 종류의 콘텐트를 능동적으로 제공하기 위해서는 하드웨어적인 한계를 벗어나 외부 자원을 활용하는 것이 필수적이다. 이러한 요구사항을 충족하는 클라우드 기반 통합 프레임워크는 클라우드 환경 내의 자원을 활용하여 로봇이 더욱 다채로운 콘텐트를 제공할 수 있도록 지원하며 더욱 흥미로운 교육 콘텐트를 제공할 수 있을 것으로 기대된다. 클라우드 기반 통합 프레임워크는 다음과 같은 요소로 구성된다.

- 클라우드 기반 지식 공유 체계
  로봇 성능의 한계를 극복하기 위한 자원을 외부의 환경에서 확보하기 위해 클라우드 환경을 활용하고자 한다. 이를 통해 로봇은 소형의 크기를 유지하면서도 더욱 다양한 콘텐트를 제공할 수 있을 것이다.
- 지식 기반 지능형 작업관리 체계
  절차적 패러다임의 프로그래밍 환경을 발전시켜 지능적이고 다채로운 행동을 구현할 수 있는 지식 기반 작업관리 체계를 적용한다. 이는 단순하고 반복적인 콘텐트의 한계를 극복하고 다채로운 로봇의 행동을 구현할 수 있게 지원한다.
- 미디어 기반 로봇작업 개발 환경
  로봇의 교육적 목적을 제고하기 위해 미디어를 통하여 로봇 작업을 개발할 수 있는 환경을 제공한다. 스마트 장치와 같은 미디어를 교육 콘텐트와 연계하여 활용함으로써 교육적 효과를 높이고 다양한 시나리오를 개발할 수 있게 된다.

3.2 시스템 개발 과정

1) 시나리오

사용자는 로봇을 이용해 교육 서비스를 제공받기 위하여 로봇을 구매한 뒤 커리큘럼을 통해 로봇의 행동을 개발하는 과정을 따르게 된다. 사용자는 최초 단순한 행동을 스마트 장비를 통해서 개발하는 과정을 거쳐 점점 지능적이고 복잡한 문제를 해결하기 위한 논리적인 절차를 학습하게 된다.

사용자는 로봇을 개발하기 위한 환경을 스마트 기기를 통해 다운받는다. 이 환경은 로봇의 센서와 같은 다양한 이벤트를 통해 로봇의 행동을 정의하고 실시간으로 결정할 수 있는 프로그래밍 패러다임을 수용하여 실시간으로 클라우드와의 연동을 통해 로봇의 행동에 영향을 미치는 과정을 학습할 수 있도록 제공한다.

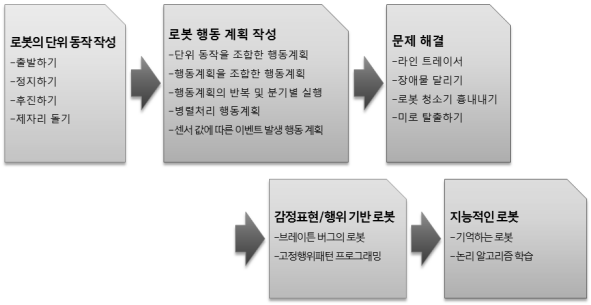

Fig. 1. Curriculum for Robot Programming

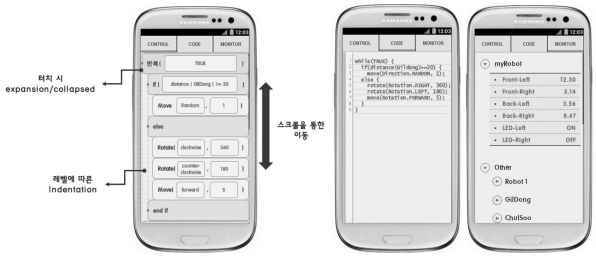

Fig. 2. Mobile Interface for Robot Programming

로봇의 교육 커리큘럼은 장애물 회피와 같은 단순한 것으로부터, 다른 로봇과의 상호작용과 같은 복잡한 이벤트 처리에 이르기까지 다양한 형태로 구성될 것이다.

이러한 이벤트 기반 프로그래밍은 클라우드와의 연동을 통해 로봇 자체의 성능과는 무관하게 개발할 수 있도록 구성될 것이며, 로봇이 구동될 때 실시간으로 적용되는 모습을 관찰할 수 있을 것이다.

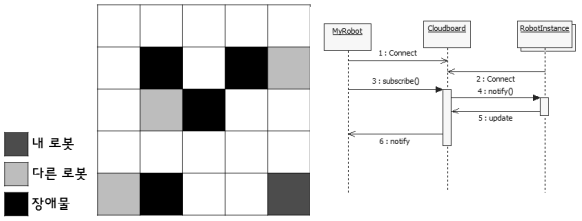

Fig. 3. Robot Programming in Cloud Environment

2) 아키텍처

위와 같은 시나리오를 만족할 수 있는 시스템을 개발하기 위하여, 필수적인 요소는 다음과 같다.

- 로봇이 단독, 혹은 다수의 로봇이 상호작용하는 상황에서도 효과적으로 정보를 공유하여 행동을 결정할 수 있도록 지원하는 지식 공유 체계
- 로봇이 알고 있는 혹은 외부 자원을 통해서 알게 될 수 있는 지식을 바탕으로 다음의 행동을 실시간으로 결정할 수 있는 작업관리 체계
- 로봇의 행동을 직관적으로 프로그래밍 하고, 로봇의 제한된 성능을 극복할 수 있도록 제어 및 연산을 대체할 수 있는 스마트 장치 기반 사용자 인터페이스 체계

이러한 요소와 더불어 다음의 요구사항을 만족하여야 한다.

- 각 시스템은 클라우드, 모바일 환경, 교육용 로봇이 유기적으로 연결되어 활용될 수 있어야 한다.
- 클라우드 환경은 사용자의 등록정보, 로봇의 센서 및 구동기 정보, 로봇의 행동 계획 등 다양한 정보가 저장되어 공유될 수 있어야 하며, 또한 비동기적으로 로봇의 행동에 영향을 미칠 수 있는 인터페이스를 제공하여야 한다.
- 모바일 장비는 로봇의 행동을 초급의 사용자도 쉽게 프로그래밍 할 수 있는 UI를 제공하여야 하며, 내부적으로는 로봇의 제한된 자원을 대체하여 작업을 관리하기 위한 연산을 수행하고, 로봇과 통신하여 센서 정보를 분석할 수 있는 작업관리 체계를 갖추어야 할 것이다.
- 로봇은 이를 위해서 각 센서와 구동기가 실시간으로 송/수신되는 정보를 통해 보고 및 제어를 수행할 수 있도록 인터페이스를 제공하고 프로토콜을 확립하여야 할 것이다.

다음 그림은 위의 요소와 요구사항을 반영한 클라우드 기반 로봇 통합 프레임워크의 구성을 보여준다. 다음 절에서는 이 프레임워크의 핵심 요소인 클라우드보드의 필요성과 더불어 기능 및 성능을 설명한다.

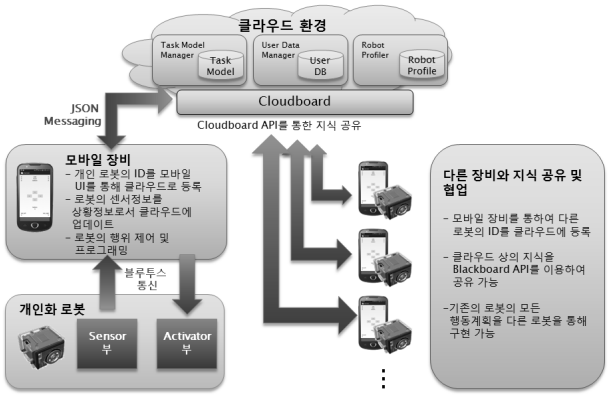

Fig. 4. Cloud Robot Programming Framework

4. 클라우드 기반 통합 프레임워크 구축

클라우드 기반 지식 공유 체계를 구축하기 위하여 본 사업에서는 블랙보드 아키텍처를 차용하여 클라우드 환경에 적용한 클라우드보드 시스템을 제안한다. 블랙보드 아키텍처는 로봇이 필요로 하는 지식정보를 공유하는 목표를 달성하기에 적합한 구조를 띄고 있으며, 나아가 비동기적으로 지식정보를 작업관리 체계에 알려줄 수 있는 인터페이스를 제공한다. 이러한 특징은 앞서 언급한 목표를 달성하기에 충분한 기능을 제공할 뿐만 아니라, 향후 여러 로봇들의 협업을 위한 체계로 발전시키기에도 적합한 확장성을 지니고 있다.

모바일 장비와 클라우드보드의 인터페이스는 블랙보드 아키텍처에서 정의하는 인터페이스를 차용하면서, 클라우드 상의 지식 공유에 적합한 메시지 구조를 정의하여 활용한다. 메시지의 구조는 Table 2와 같다.

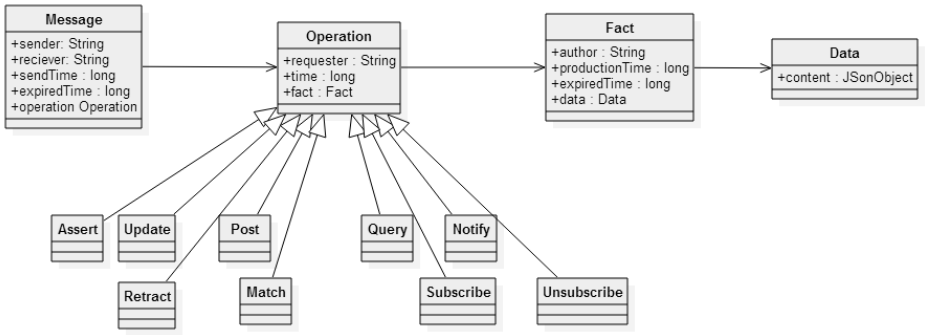

Fig. 5. Interface to the Cloudboard

Table 2. Cloudboard Interface

메시지	구 성
Data	클라우드보드에서 저장하는 실제 데이터 Content : Key-Value 형태의 구성 데이터
Fact	저장하는 데이터의 메타정보 Author : 데이터를 생성한 저자 productionTime : 데이터를 생성한 시간
Operation	클라우드보드를 통해서 수행하는 게시, 삭제, 수정 등의 행위를 정의 requester : 작업의 요청을 수행한 주체 time : 작업이 요청된 시간 operationType : 요청된 작업의 종류
Message	클라우드보드와 로봇 시스템이 주고받는 메시지를 위한 Envelop sender : 메시지를 보낸 주체의 URI receiver : 메시지를 수신하는 URI sendTime : 메시지가 송신된 시간

클라우드 기반 통합 프레임워크는 앞서 언급한 바와 같이 블랙보드 아키텍처의 기본 구조를 기반으로 하고 있다. 블랙보드 아키텍처의 장점은 데이터를 공유하기 위하여 기존의 1:1 메시징 방식보다 데이터의 전송량이 매우 줄어든다는 점에 있다.

에이전트 N개가 1:1 메시징 방식의 정보 공유를 위해서는 임의의 에이전트 i가 공유하기 위한 메시지 개수 M_i에 대해서, 어떠한 에이전트가 해당 메시지를 요구할지 알 수 없으므로 N-1개의 에이전트에 데이터를 전송할 필요가 있으며, 하나의 메시지를 전송할 때의 전송비용이 t_{mt}라 할 때, 총 전송비용은

$$T_{1:1} = \sum_{i=1}^{N} M_i (N-1) t_{mt}$$

에 달한다. 하지만 블랙보드 아키텍처의 Subscribe 방식을 이용하여 데이터를 공유하게 되면, 모든 데이터는 클라우드보드 컴포넌트가 관리하게 되고, 자신이 가지고 있는 Subscription 리스트를 통해 전송해야 하는 에이전트를 구분할 수 있다. 따라서 임의의 에이전트 i가 공유하기 위한 메시지 개수 M_i를 클라우드보드에 전송하게 되면, 클라우드보드는 해당 메시지에 포함된 정보를 필요로 하는 에이전트를 리스트에서 검색하여 개별적으로 m_i개의 메시지를 전송하게 되므로 총 전송비용은

$$T_{cb} = \sum_{i=1}^{N} M_i (1+m_i) t_{mt}$$

가 된다. 이때 m_i는 N에 비해서 항상 작거나 같으므로, 총 데이터 전송량은 언제나 1:1 방식의 메시지 전송방식에 비해서 개선된 성능을 가져오게 된다.

블랙보드 아키텍처를 기반으로 하고 있는 클라우드보드 시스템은 데이터의 전송량뿐만 아니라, 각 에이전트가 가지는 리소스 부담을 줄이는 효과도 함께 가져오게 된다. 각 에이전트는 자신이 데이터를 공유하기 위한 주소의 리스트를 가지고 있는 부담을 클라우드 환경에 위임하고 스스로는 클라우드보드의 주소만을 소유하게 되는데, 모바일 환경과 같이 제한된 리소스를 활용하는 시스템 하에 교육 커리큘럼이나 사용자 작업계획과 같이 다양한 데이터를 보유하여야 하는 상황에서 이와 같은 개선은 매우 유용하다고 할 수 있다.

5. 실험에 의한 성능분석

본 논문에서는 클라우드 기반 통합 프레임워크 체계로서 클라우드보드를 제안하고, 이를 통해 개선된 정보 공유방법을 제시하였다. 이 항목에서는 클라우드보드 시스템이 가시적으로 성능의 개선점을 확인할 수 있는 메시지 전송량을 검증하고, 이를 통해 다자간 정보 공유상황에서의 클라우드보드 시스템의 개선점을 확인한다.

우리는 교육상황에서의 다자간 정보 공유를 가정하기 위하여 하나의 학급에 20대의 로봇으로 구성된 학급 교육장 10곳에서 동시에 교육을 진행하는 것을 시나리오로 가정하였다. 하나의 로봇은 학급 내의 다른 로봇과 상호 연동된 학습효과를 제공하기 위하여 학급 내부에 있는 19대의 로봇과 데이터를 주고받아야 하며, 이러한 행위는 10곳의 교육장에서 동시에 수행되게 된다.

여기서는 이와 같이 활발하게 데이터를 포함한 메시지를 전송해야 하는 상황에서, 1:1 방식의 메시징 시스템에 비해

클라우드보드 시스템의 데이터 전송량 및 데이터 전송에 걸리는 평균시간을 측정하여, 데이터 비용의 절감효과를 확인하고자 하였다.

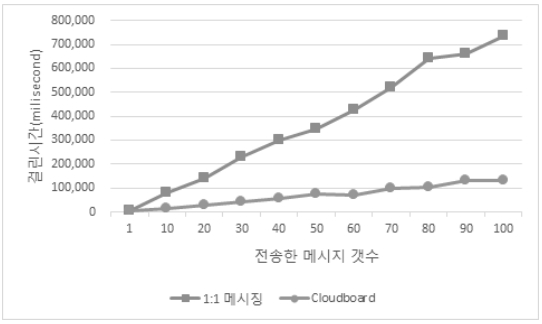

Fig. 6. Messaging Performance of Cloudboard

위의 그래프와 같이 클라우드보드 시스템은 기존의 1:1 방식 메시징 시스템에 비하여 획기적으로 절감된 전송 비용을 보이는 것을 확인하였다. 논문에서 가정하고 있는 교육 상황에 대한 로봇 활용을 위해서는 위의 가정과 같이 적게는 수십에서 많게는 수천 대까지의 로봇이 동시에 운용되는 상황을 쉽게 가정할 수 있는 것을 고려할 때, 각각의 로봇이 서로에게 정보를 제공하고 이를 통한 작업을 수행하는 상황에서의 데이터 전송비용의 절감효과는 중요한 요소로 작용할 것이다.

본 실험을 통하여 이와 같은 문제를 해결하고자 하는 클라우드 기반 통합 프레임워크의 지식 공유 요구 조건을 만족함을 확인하고, 교육상황뿐만 아니라 에이전트가 서로 지식을 공유할 필요가 있는 환경에서도 손쉽게 응용될 수 있는 가능성을 파악하였다.

6. 요약 및 결론

본 논문에서는 클라우드 환경에서 모바일 기기와 로봇을 이용하여 손쉽게 소프트웨어를 개발하고 프로그래밍 교육을 할 수 있는 클라우드 기반 지식 공유 및 제어 시스템인 클라우드보드(Cloudboard)를 제안하였다. 클라우드보드는 기존의 대표적 아키텍처의 패턴을 비교 분석한 결과와 로봇 활용 프로그래밍 교육의 요구사항을 반영하여 개발되었으며 이에 따라 다수의 로봇이 개별 로봇의 센서 정보를 서로 공유하여 공동 임무를 수행하는 것과 같은 복잡한 프로그램도 클라우드보드의 인터페이스를 활용하면 비교적 손쉽게 개발할 수 있게 된다.

클라우드보드 아키텍처는 블랙보드 기반의 인터페이스를 통하여 클라우드 환경에서 로봇이 지식을 공유하여 협업이 가능하도록 한다. 이에 따라 로봇의 성능이나 용량의 제약을 넘어 정보의 공유와 이를 통한 클라우드 기반의 상황 관리 및 제어가 가능하도록 한다. 특히 로봇을 프로그래밍 교육에 활용하는 경우에 다수의 로봇이 협력하는 고난도의 프로그래밍도 클라우드보드에서 제공하는 효율적인 인터페이스를 통하여 손쉽게 실현할 수 있게 되어 교육의 효과가 증대할 것으로 기대된다. 향후 로봇의 행동을 더욱 다양하게 프로그래밍 할 수 있는 작업계획과 다양한 미디어와의 상호작용 부분이 보완되면, 모바일 환경과 로봇을 이용한 프로그래밍 교육 환경의 개선에도 기여할 것으로 기대된다.

References

[1] 백인수, “창조경제 구현을 위한 컴퓨터 프로그래밍 교육 정책의 바람직한 방향,” *IT & Future Strategy*, 제7호, pp.1-29, 2013.
[2] 이영준, “로봇의 교육적 활용 방안 및 적정 기능 연구,” *연구보고 KR 2007-26*, 2007.
[3] Corkill, Daniel D, “Blackboard Systems,” *AI Expert*, Vol.6, No.9, pp.40-47, 1991.
[4] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sonnerlad, and Michael Stal, “Pattern-Oriented Software Architecture Volume 1: A System of Patterns,” 1996.
[5] Kai Qian, Xiang Fu, Lixin Tao, Chong-Wei Xu, Jorge L Diaz-Herrera, “Software Architecture and Design Illuminated,” p.200, 2009.
[6] J. Liedtke. “Toward Real Microkernels,” *Communications of the ACM*, Vol.39, No.9, pp.70-77, Sep., 1996.

이 재 호
e-mail : jaeho@uos.ac.kr
1985년 서울대학교 계산통계학과(학사)
1987년 서울대학교 계산통계학과(석사)
1997년 University of Michigan(박사)
현 재 서울시립대학교 전자전기컴퓨터공학부 교수
관심분야 : 인공지능, 지능 로봇

최 병 기
e-mail : byunggi.choi@gmail.com
2010년 서울시립대학교 전자전기컴퓨터공학부(학사)
2012년 서울시립대학교 전자전기컴퓨터공학과(석사)
현 재 서울시립대학교 전자전기컴퓨터공학과 박사과정
관심분야 : 인공지능, 지능형 시스템

배 재 형
e-mail : gaoroter@gmail.com
2010년 서울시립대학교 전자전기컴퓨터공학부(학사)
현 재 서울시립대학교 전자전기컴퓨터공학과 석사과정
관심분야 : 인공지능, 지능형 시스템