제25회 한국정보처리학회 춘계학술발표대회 논문집 제13권 제1호 (2006. 5)

사용자 행동패턴을 기반으로 한 멀티 에이전트 시스템 구조

김민경*
*한국 전자통신연구원 지능형 로봇 연구단
e-mail : bluet9@etri.re.kr

Multiagent system for the Life Long Personalized Task Coordination based on the user behavior patterns

Minkyoung Kim*
*Intelligent Robot Research Division,
Electronics and Telecommunications Research Institute

요 약

유비쿼터스 컴퓨팅의 핵심은 네트워크 환경에 대한 고 가용성이라 할 수 있다. 이러한 사실은 사용자 컨텍스트(Context)가 반영된 서비스를 제공하기 위한 필수 조건이 이미 갖추어져 있다는 것을 시사한다. 지금까지 상황인지(Context-Aware) 서비스를 위한 여러 응용들이 제시되어 왔지만, 동적으로 변화하는, 즉 예측하기 어려운 환경을 충분히 반영할 만큼의 유연성을 제공하지 못했다. 왜냐하면, 응용 태스크 시나리오가 시작단계부터 이미 정해져 있었기 때문이다. 여기에, 본 고는 평생 동안 개인화된 태스크를 동적으로 생성, 제공할 수 있는 멀티 에이전트 시스템 구조를 제안하고자 한다. 평생 개인화 태스크 (Life Long Personalized Task)는 끊임없이 변화하는 사용자의 행동패턴을 반영할 수 있도록, 동적으로 생성, 제공되는 태스크를 의미한다. 이는 태스크 시나리오가 컴파일 타임에 이미 결정되지 않고, 실행 시간 중에 자동으로 생성된다는 것을 의미한다. 이러한 유연성은 평생 학습 엔진(Life Long Learning Engine)을 활용함으로써 가능하다. 이 엔진은 사용자의 행동패턴을 학습하며, 결과적으로 사용자 행동패턴 규칙들을 생성한다.

1. 서론

유비쿼터스 컴퓨팅은 스마트 스페이스(Smart Space)에서의 Ambient Intelligence 에 대한 관심을 불러 일으켰다. 언제, 어디서나, 모든 종류의 정보를 접근할 수 있는 네트워크 환경을 제공해주기 때문에 가능한 일이다. 이러한 정보 접근성을 바탕으로, 정보의 활용성을 최대화하려는 시도들이 여러 응용분야에서 나타나게 되었다. 예를 들면, 웹에서의 정보 추출[1], 멀티 에이전트를 기반으로 한 태스크간 조율 (task coordination) [2][3][5][6], 사용자 일정관리 에이전트 등 여러 응용들이 있겠다. 이러한 응용들 간의 공통된 궁극적인 목표는 사용자의 상황을 정확하게 인지함으로써, 시스템에 대한 사용자 만족도를 높이는 것이다. 본 고는 특히 인공지능 환경(Intelligent Building)에 초점을 맞추어 동적인 환경변화와 사용자의 행동패턴을 반영할 수 있는, 멀티 에이전트 시스템의 일반적 구조를 제시하고자 한다. 이러한 구조를 위해 퍼지 귀납 학습엔진을 도입하였다. 이 엔진으로부터 얻을 수 있는 중요한 요소는, 평생학습을 통해 사용자 행동패턴에 관한 동적 규칙을 얻을 수 있다는 것이다. 그렇지만, 본 고는 학습엔진 구조와 메커니즘 자체에 대한 언급은 하지 않고, 개인화된 태스크의 동적 생성과 조율을 위한 멀티 에이전트 구조에 초점을 둘 것이다.

평생 개인화 태스크를 위한 멀티 에이전트 시스템의 궁극적 목표는 컴파일 타임이 아닌 실행시간에, 사용자 상황에 적합한 개인화된 태스크를 자동적으로

생성, 제공하고, 이를 조율하는 것이다. 여기에는 몇 가지 중요한 동기들이 있다.

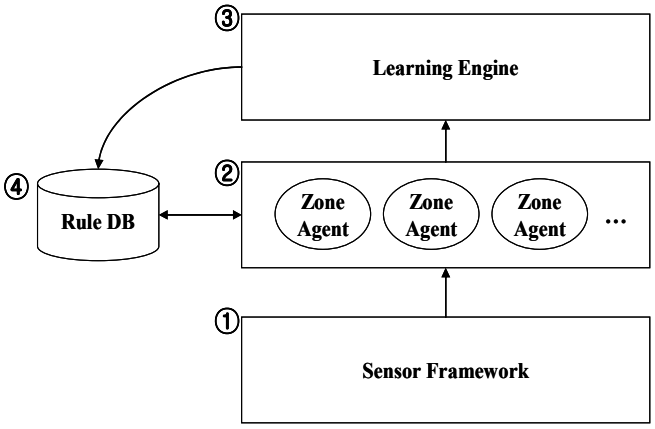
첫째로, 사용자 행동 패턴과 사용자가 처한 환경은 고정적이지 않으므로, 개발자들은 이러한 변화를 미리 예측하기 힘들고, 따라서 완벽한 시나리오를 구현하는 것은 불가능하다. 둘째로, 사용자의 선호 사항(User Preferences)은 사용자 마다 다르므로, 이를 반영시키기 위해서는 일괄적이고 획일적인 태스크가 아닌 개인화된, 즉 차별화된 태스크가 필요하다. 이는 사용자의 만족도를 높이기 위한 것뿐만 아니라 시스템을 효율적으로 관리할 수 있도록 도와주는 역할도 한다. 마지막으로, 시스템 유지보수 비용에 관한 것이다. 앞에서도 언급하였듯이, 사용자의 행동패턴이나 취향은 끊임없이 변화하므로 이런 변화가 있을 때마다 태스크 시나리오를 변경하는 것은 적지 않은 비용이며, 개발자에게 큰 부담이 될 수 있다.

본 고의 나머지 부분은 다음과 같이 구성되어 있다. Section 2 에서는 평생 개인화 태스크를 위한 멀티 에이전트 시스템의 전반적 구조에 대해 논의하고, Section 3 에서는 태스크간 조율 메커니즘 (Task Coordination Mechanism)에 대해 자세히 기술하겠다. 마지막으로 Section 4 에서는 앞으로의 연구방향과 몇 가지 주요사항으로 결론지을 것이다.

2. 평생 개인화 태스크를 위한 멀티 에이전트의 전체적 구조

우리가 흔히 주위에서 접할 수 있는 호텔을 예로 들어, 저녁 식사나 객실 예약을 위해 호텔을 방문할 때를 생각해 보자. 모든 호텔에는 각 중요한 위치마다 해당 에이전트들이 항상 대기하고 있는 것을 볼 수 있다. 예를 들면, 안내 데스크, 레스토랑, 주차장 입구에는 관련 정보를 제공하기 위한 도우미들이 대기하고 있다. 주차 도우미는 주차 요금이나 주차 공간을 지정해 주고, 안내 데스크에서는 호텔 객실, 편의시설 정보를 제공하거나 예약을 도와준다. 다시 말하자면, 실 세계는 어떤 보이지 않는 논리적인 공간에 의해 구분되어 있다. 선명하게 벽으로 구분되어 있지는 않지만, 우리는 호텔 예약을 하기 위해 주차장으로 가지 않고 안내데스크로 향한다. 각 에이전트가 위치에 따라 다른 서비스를 하고 있다는 것을 지금까지의 경험으로 알고 있기 때문이다. 이러한 사실을 착안하여, 본 고에서는 가상의 지역 에이전트 (Zone Agent)를 고안하였다. (그림 1)

실 세계의 시스템을 "평생 개인화 태스크를 위한 멀티 에이전트 시스템"에 반영하였다. 즉, 각 중요한 위치마다 지역 에이전트(Zone Agent)를 위치시켜, 각 지역 에이전트가 속한 영역과 관련된 서비스들을 제공할 수 있도록 하였다. 여기서 말하는 "위치"란 물리적인 공간이 될 수도 있고, 논리적인 위치 공간이 될 수도 있다.



(그림 1) 평생 개인화 태스크를 위한 멀티 에이전트 시스템의 주요 구조

(그림 1)은 평생 개인화 태스크를 위한 멀티 에이전트 시스템의 간략한 구조를 보여준다. 좀 더 상세한 구조는 다음 Section 에서 기술할 것이다. 평생 개인화 태스크(Life Long Personalized Tasks)를 위한 지역 에이전트는 다음과 같은 세가지 중요한 특징을 가지고 있다. (1) 유연성 (Flexibility) (2) 개인 차별화 (Personalization) (3) 태스크간 조율 (Task Coordination).

첫째로, 대부분의 태스크들은 E-C-A 규칙에 기반하여 실행된다. 즉, 어떤 특정한 이벤트(**E**vent)가 발생하였을 때, 현재 상황이 어떤 조건(**C**ondition)에 부합된다면, 어떤 일련의 행동(**A**ction)들이 전개된다. 그러나, 이미 결정되어 있는, E-C-A 규칙들에 기반한 태스크 시나리오는 끊임없이 변화하는 실 세계를 반영할 수 없다. 이러한 이유로, 시스템의 영구성을 위해 유연성 있는 구조가 필요하다.

둘째로, 일부의 공통되고 획일화된 태스크는 사용자 그룹에 공유될 수 있지만 항상 그렇지는 못하다. 왜냐하면 사용자의 취향과 행동패턴이 각기 다르기 때문이다. 여기에 태스크는 개개인 별로 차별화될 필요가 있다.

셋째로, 실 세계에서는 객체 혼자서는 살아갈 수 없다. 여러 다양한 객체들이 서로 상호작용을 하며 살아가고 있다. 여기에는 예견치 못한 충돌이 불가피하게 되는데, 이러한 충돌을 회피하고 상황을 조율하기 위한 개체들간의 조율이 필요하게 된다. 실 세계 환경 구조 마찬가지로, 여기에서도 태스크간의 조율이 필요하다.

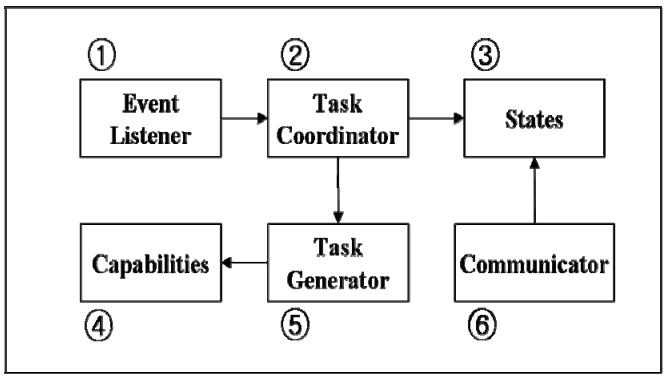
또한, 위의 주요 특성들을 반영하기 위하여, 평생 학습엔진(③Learning Engine)을 도입하였다. 이는, 사용자 일생 동안의 행동패턴 변화와 환경의 변화 등을 반영시키기 위함이다. 현재 많은 학습 방법론들이 존재하고 있지만, 여기에서 제안하고자 하는 학습엔진은 퍼지 귀납 평생 학습 시스템이다. 이 시스템은 평생 동안 변화하는 사용자의 행동패턴을 학습하여, 궁극적으로 개개인의 행동 패턴 규칙들을 생성시켜 준다[4]. 이 논문에서는 자세한 학습 엔진 구조와 메커니즘에 대해서는 더 이상 언급하지 않을 것이다.

(그림 1)에서 센서 프레임워크(①Sensor Framework)

는 센서에서 읽어드린 정보를 하나의 이벤트로 해석하고, 이 이벤트 정보를 센서 감지가 발생된 곳에 위치한 지역 에이전트 (②Zone Agent)에 전달한다. 그러면 지역 에이전트 (②Zone Agent)는 규칙 데이터베이스 (④Rule DB)로 질의(query)를 보내게 되는데, 전달받은 이벤트정보와 관련된 규칙들이 존재하는지 여부를 알아보기 위해서이다. 여기에서, 규칙 데이터베이스(④Rule DB)란 학습엔진(③Learning Engine)으로부터 얻은 사용자 행동 패턴 규칙들의 집합이라고 할 수 있다. 만일, 규칙 데이터베이스(④Rule DB)에 관련된 규칙이 존재한다면, 지역 에이전트는 이 규칙과 관련된 태스크를 동적으로 생성하게 된다. 이렇게 생성된 태스크를 통해서, 지역 에이전트(②Zone Agent)는 사용자 취향을 고려한 상황인식 서비스를 제공하게 된다. 또한, 어떤 사용자의 행동은 사용자가 위치한 환경과의 상호 작용을 통해서 인식되며, 이 행동은 학습엔진에 입력 값으로 주어져서 계속적으로 학습하게 된다. 학습 결과 변경된 규칙은 계속적으로 규칙 데이터베이스에 반영된다.

3. 평생 개인화 태스크를 위한 지역에이전트 메카니즘

평생 개인화 태스크를 가능하게 하기 위해서, 지역 에이전트 자체의 구조는 매우 중요하다. 왜냐하면, 이곳에서 결국 태스크간 조율이 이루어지고, 적절한 태스크들이 실행시간에 동적으로 생성, 제공되기 때문이다. (그림 2)에서 보는 바와 같이, 지역 에이전트는 6개의 주요 컴포넌트로 구성되어 있다.



(그림 2) 지역 에이전트 구조

지역 에이전트 내의 각 컴포넌트의 주요 기능들은 다음과 같다.

(1) **Event Listener:** Section 2 에서 언급한 바와 같이, Event Listener(①)는 (그림 1)의 센서 프레임워크(Sensor Framework)에서 인식된 이벤트 정보를 받는다. 그리고 Task Coordinator(②)에게 어떤 종류의 이벤트가 발생하였는지 알려 준다.

(2) **Task Coordinator:** Event Listener(①)로부터 이벤트 정보를 전달 받으면, Task Coordinator(②)는 이 이벤트가 센서(Sensor)로부터 발생된 이벤트인지, 액추에이터(Actuator)로부터 발생된 이벤트인지를 먼저 파악한다. 이벤트의 종류가 이처럼 센서 이벤트와 액추에이터 이벤트, 두 가지로 나뉘어지는 이유는 각각의 이벤트가 다른 방식으로 처리되어야 하기 때문이다. 액추에이터를 통한 사용자의 행동은 (예: TV 켜기, 라디오 켜기, 불 켜기 등) Actuator Event로 표현되어 학습엔진(Learning Engine)으로의 입력 값으로서 전달된다. 왜냐하면, 이런 사용자 행동들은 피드백(feed back)이나 또 다른 사용자 행동 패턴 규칙을 얻어내기 위한 데이터가 될 수 있기 때문이다. Sensor Event의 경우는 광의의 환경 변화라고 해석될 수 있다. 예를 들면, RFID Tag 인식, 사용자 음성명령 인식, 사용자 영상 인식 등이 그것이다. 이러한 이벤트들은 Task Generator(⑤)로 전달된다. Task Generator(⑤)는 (그림 1)의 규칙 데이터 베이스 (Rule DB)에 질의하여, 발생된 센서 이벤트와 관련된 규칙이 존재하는 지의 여부를 조사하게 된다.

(3) **States:** 가장 최근의 사용자 행동을 기억할 필요가 있다. 왜냐하면, 이 정보가 다른 지역 에이전트에 의해 참조될 수 있기 때문이다. 가령, 사용자가 현재의 영역에서 다른 영역으로 이동한다고 했을 때, 새로운 영역에 위치한 지역 에이전트는 최근까지 사용자가 무엇을 했는지의 정보만 알 수 있다면, 사용자에게 그와 관련된 서비스를 계속해서 제공할 수 있기 때문이다(Seamless Service). 안방에서 KBS TV 채널을 시청하고 있었다면, 다른 방에서도 계속적으로 그 채널을 시청할 수 있는 서비스를 제공하기 위함이다. 또한, 수행되고 있는 태스크들의 현재 상태정보도 가지고 있다. 이는 지역 에이전트간 태스크 조율을 하기 위함인데, 이러한 정보의 이동은 Communicator(⑥)를 통해서 가능하다.

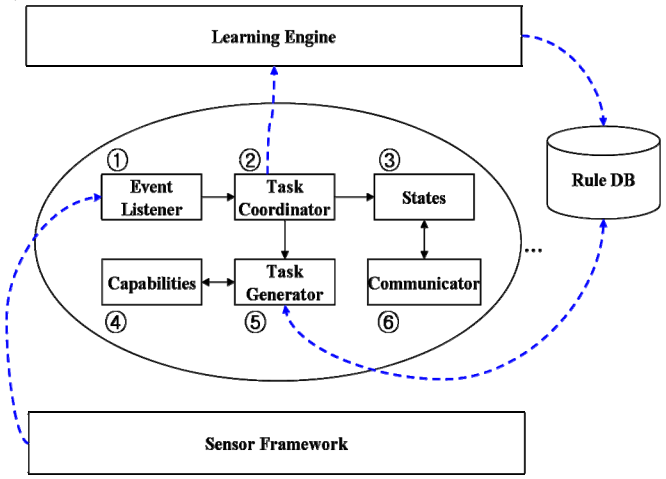
(4) **Capabilities:** 어떤 특정 영역에서 활용 가능한 서비스들이라 할 수 있다. 모든 영역은 각기 서로 다른 종류의 서비스들을 제공한다. 왜냐하면, 각 영역에 위치하고 있는 디바이스(device)들의 종류가 다르고, 또한 각 디바이스들은 수시로 바뀔 수 있기 때문이다. 따라서, Capabilities는 Task Generator(⑤)에게 가용한 서비스가 어떤 것들이 있는지에 대한 정보를 제공하는 역할을 한다.

(5) **Task Generator:** Task Coordinator(②)에 의해 이벤트 정보를 받으면, (그림 1)의 규칙 데이터베이스(Rule DB)에 이 이벤트와 관련된 규칙들이 존재하는 지의 여부를 알아보기 위한 질의를 하게 된다. 만일, 관련된 규칙들이 있다면, Task Generator(⑤)는 이 규칙들에 의거해서 동적으로 실행시간에 태스크를 생성하게 된다. 여기에서, 태스크 시나리오는 학습엔진에 기반한 행동패턴 규칙에 의거하기 때문에, 각 사용자에 특화된 시나리오가 된다. 하지만, 현재 위치한 영역(Zone)에 해당되는 서비스들이(Capabilities)이 존재하지 않는다면 해당 태스크는 생성되지 않는다.

(6) **Communicator:** 각 지역 에이전트끼리는 서로의 정보를 참조하기 위해 통신 통로가 필요하게 되는데, 이 때 Communicator가 통신 채널 역할을 하게 된다.

즉, 상태정보(States)를 공유하기 위한 일종의 통로라고 할 수 있겠다.

(그림 3)은 (그림 1)의 구조를 더 자세히 표현한 그림이다.



(그림 3) 평생 개인화 태스크를 위한 멀티 에이전트 구조

본 고에서 제시한 메커니즘이 성공적으로 동작하기 위해서는 다음과 같은 몇 가지 가정들이 필요하다.

첫째로, 한 사용자에 관련된 규칙들간 충돌이 없어야 된다는 것이다. 예를 들어, 다음과 같은 사용자 'A'에 대한 규칙들이 존재한다고 가정해 보자.

(1) 규칙_01: 사용자 'A'가 거실에 들어서면, TV 를 켠다.

(2) 규칙_02: 사용자 'A'가 거실에 들어오면, 라디오를 켠다.

위와 같은 규칙 (1), (2)는 서로 충돌하게 된다. 왜냐하면, 두 규칙이 동시에 실행된다면 TV 소리와 라디오 소리가 섞여 사용자 'A' 에게 결과적으로 소음만 안겨 줄 것이기 때문이다.

둘째로, 어떤 사용자가 한 영역에 먼저 진입해 있다면, 두 번째 다른 사용자가 그 영역에 진입할 지라도 태스크에 대한 우선권은 먼저 진입한 사용자가 갖는다(First-In First-Serve). 즉, 다른 사용자가 그 영역에 진입할 지라도, Task Generator 는 지금 들어온 사용자에 대한 태스크를 실행시키지 않고 기다리다가, 원래 있던 사용자가 그 영역을 나가면 그 때 실행시킨다. 이는, 두 사용자 태스크 간의 충돌을 피하기 위해서이다.

마지막으로, 위급한 상황에 대한 태스크는 가장 높은 우선순위를 갖는다. 가령, 화재나 무단 침입이 일어났을 때, 이와 관련된, 이미 정해진 시나리오의 태스크는 현재 실행되고 있는 어떤 태스크들보다도 가장 높은 우선순위를 갖는다.

4. 결론

많은 종류의 멀티 에이전트 시스템 구조가 연구, 개발되어 왔다. 하지만, 대부분이 미리 정의된 시나리오를 가지고 응용 태스크를 적용시키고 있다. 따라서, 장기간적인 관점으로 보았을 때, 시스템 유연성이 부족하다고 하겠다. 즉, 태스크 시나리오는 주어진 환경이 변화할 때마다 동적으로 생성되고 변화할 수 있어야 한다. 본 고는 이러한 측면에서 평생 개인화 태스크를 위한 멀티 에이전트 시스템 구조를 제안하였다. 제안한 시스템의 구조는 학습엔진을 활용하여 개 개인별로 차별화된 규칙을 적용하는 것이다. 하지만, 여기에서도 학습엔진 성능에 많은 부분 영향을 받을 수 있다. 그러나, 학습을 제외하고 개인화된 평생 태스크를 제공하는 것은 어려운 일이다. 또한 개발자 스스로가 각 개인별 행동패턴 변화를 미리 예측해서 완전한 시나리오를 결정한다는 것은 더더욱 불가능한 일이다. 또한, 사용자의 개입을 최소화 하기 위해서 사용자의 feed back 을 계속 학습하는 것도 중요한 요소라 할 수 있다.

앞으로의 연구방향은 태스크간 조율 메커니즘을 더욱 자세히 하고, 동일 공간 상의 다중 사용자간의 Win-Win Condition 에 대해 연구할 것이다. 또한, 학습엔진의 성능 또한 현재 고려하고 있는 학습 시스템에서 더욱 향상 되어야 할 것이다.

참고문헌

[1] Ashishi, Naveen, & Craig A. Knoblock. "Semi-automatic Wrapper Generation for Internet Information Sources," Information Sciences Institute and Department of Computer Science USC.

[2] Paul Scerri et. al., "Getting robots, agents and people to cooperate: An initial study," AAAI Spring Symposium 2003.

[3] Paul Scerri et. al., "A Prototype Infrastructure for Distributed Robot-Agent-Person Teams," proceedings of AAMAS'03, Melbourne, Australia, July, 2003.

[4] Hyong-Euk Lee et. al., "Fuzzy Inductive Learning System for Learning Preference of the User's Behavior pattern," Journal of Korea Fuzzy Logic and Intelligent System Society, vol. 15, number 6, December, 2005.

[5] Hyun Kim and Young-Jo Cho and Sang-Rok Oh, "CAMUS: A middleware supporting context-aware services for network-based robots," IEEE Workshop on Advanced Robotics and Social Impacts, Nagoya, Japan, 2005.

[6] Paul Davisdsoon & Magnus Boman, "A Multi-Agent System for Controlling Intelligent Buildings," MultiAgent Systems, 2000. Proceedings of Fourth International Conference, July, 2000

[7] Boman M., Davidsson P., et al., "Energy Saving and Added Customer Value in Intelligent Buildings", Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp.505-517, 1998