유비쿼터스 환경을 위한 서비스 에이전트의 동적 생성 기법[1)]

송창환*, 김연우*, 장현수*, 김구수**, 엄영익*
*성균관대학교 정보통신공학부
**동양대학교 정보통신공학부
e-mail:{eerien, daroobil, jhs4071, yieom}@ece.skku.ac.kr,
gusukim@dyu.ac.kr

Dynamic Generation Scheme of Service Agent in Ubiquitous Environments

Changhwan Song*, Younwoo Kim*, Hyunsu Jang*, Gu Su Kim**, Young Ik Eom*
*School of Information and Communication Eng., Sungkyunkwan University
**School of Information and Communication Eng., Dongyang University

요 약

유비쿼터스 환경에서 사용자의 추상적인 요구를 추론하고 추론 결과에 대응하는 일련의 작업들을 구성하여 수행해주는 시스템에 대한 연구가 진행되고 있다. 그러나 에이전트 기반의 유비쿼터스 환경을 고려하여 일련의 작업들을 구성하여 수행하기 위한 방법의 연구는 미흡한 상황이다. 이에 본 논문에서는 에이전트 기반의 유비쿼터스 환경에서 실행 시간에 정의되는 일련의 작업들을 수행 가능한 새로운 에이전트를 동적으로 생성하는 기법을 제안한다. 제안 기법을 이용하여 에이전트 기반의 유비쿼터스 환경에서 사용자의 요구에 대응하는 서비스 에이전트를 동적으로 자동 생성할 수 있다.

1. 서론

유비쿼터스 환경에서 사용자의 추상적인 요구를 추론하여 그에 대응하는 일련의 작업들을 구성하여 수행하는 시스템에 대한 연구가 진행되고 있다[1]. 이러한 시스템을 실현시키기 위해서는 먼저 사용자의 추상적인 요구를 시스템에서 수행 가능한 기능들과 대응되도록 추론하는 기술이 필요하다. 그리고 수행 가능한 기능들을 구성하여 실제로 사용자가 원하는 서비스를 제공해 줄 수 있는 기술이 필요하다.

그러나 에이전트 기반의 유비쿼터스 환경에서 일련의 작업들을 구성하여 수행하기 위한 방법의 연구는 미흡한 상황이다.

본 논문에서는 에이전트 기반 유비쿼터스 시스템에서 일련의 작업들을 구성하여 수행하는 새로운 에이전트를 동적으로 생성하는 기법을 제안한다. 먼저 각 에이전트가 수행할 수 있는 기능을 액션으로 정의한다. 그리고 액션에 대한 정보를 메서드 단위로 유지한다. 서비스 요청이 들어오면 보관중인 액션 정보를 참고하여 필요한 액션들을 추론한다. 그리고 추론한 액션들을 구성하여 수행할 수 있는 새로운 에이전트를 생성한다. 이렇게 생성된 에이전트는 사용자의 추상적인 요구에 대응하는 서비스를 제공할 수 있게 된다. 이 때, 각 액션을 수행하는 주체가 에이전트가 되므로 각 기능을 수행하는 에이전트의 지능성, 자율성에 따라 서비스를 제공하게 된다.

본 논문의 2장에서는 관련 연구에 대하여 소개하고, 3장에서는 본 논문에서 제안하는 기법을 설명하고, 이에 따르는 에이전트 시스템의 구조을 보인다. 4장에서는 본 논문에서 제안하는 기법의 구현에 대하여 설명하고, 5장에서는 결론 및 향후 연구 과제에 대하여 설명한다.

2. 관련연구

2.1. 에이전트

에이전트란 사전적 의미로 '대리인' 이란 뜻이다. 에이전트는 컴퓨터 분야에서 연구 분야에 따라 다양하게 정의할 수 있지만 본 논문에서는 에이전트란 사용자가 원하는 작업을 대신 해주는 소프트웨어로 정의한다. 에이전트는 일반적으로 자율성, 사회성, 이동성, 지능성이라는 특징을 갖는다. 이 중 자율성은 에이전트를 정의함에 있어서 가장 중요한 특징이

1) 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0046))

다. 자율성은 에이전트가 사용자의 조작 없이도 자율적으로 작업을 수행할 수 있음을 의미한다. 에이전트는 그 특징에 따라 다양한 종류로 분류할 수 있는데 특히, 네트워크를 통해 자율적으로 이동이 가능한 이동성을 가진 에이전트를 이동 에이전트라고 한다[2][3].

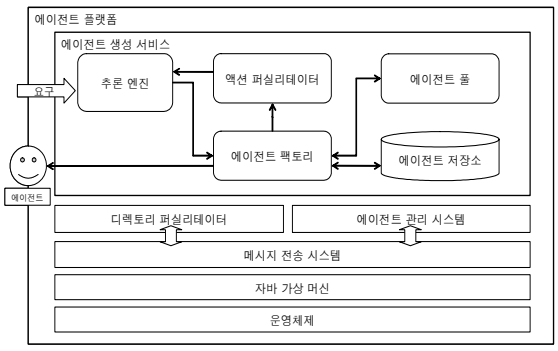
2.2. 동적 서비스 구성

동적 서비스 구성이란 서비스 선택과 프로세스 생성 모델 모두를 자동화 하는 것을 말한다[4]. 서비스 선택은 요구사항에 적합한 서비스를 찾아서 선택하는 것이다. 프로세스 생성 모델은 선택한 서비스를 구성하여 실행할 수 있게 만드는 기법을 의미한다. 서비스 선택은 추론 과정을 거쳐야 하고, 프로세스 생성 모델은 그 구현에 어려움이 있다. 따라서 동적 서비스 구성을 정확히 구현하는 것은 쉽지 않다.

3. 에이전트의 동적 생성 기법

3.1. 에이전트 시스템 구조

본 논문에서 제안하는 기법은 크게 세 부분으로 나눌 수 있다. 먼저, 가장 핵심적인 기능인 에이전트 생성, 그리고 에이전트 생성에 필요한 에이전트의 액션 정보 관리, 생성된 에이전트 객체의 저장으로 나눌 수 있다. 그림 1은 본 논문에서 제안하는 기법의 에이전트 시스템 구조을 보인다.



(그림 1) 시스템 구조

- 액션(Action): 메서드를 기반으로 에이전트가 가지고 있는 기능을 추상화한 요소
- 에이전트 액션 퍼실리테이터(AAF, Agent Action Facilitator): 에이전트의 액션 정보를 저장하고, 등록, 검색, 삭제 등의 관리 기능을 제공
- 추론 엔진(RE, Reasoning Engine): 에이전트 액션 퍼실리테이터를 참조하여 사용자의 요구에 대응하는 액션들의 구성을 추론하는 엔진
- 에이전트 저장소(AR, Agent Repository): 이주해 오거나 이미 존재하는 에이전트들의 코드 저장소
- 에이전트 팩토리(AF, Agent Factory): 에이전트 저장소와 에이전트 액션 퍼실리테이터를 참조하여 새로운 에이전트를 생성
- 에이전트 풀(AP, Agent Pool): 한번 생성된 에이전트를 동일한 요구에 대해서 재사용하기 위해 생성된 에이전트를 저장해두는 풀

에이전트 팩토리는 새로운 에이전트를 생성하는 공장의 역할을 한다. 즉, 메서드 단위로 액션을 구성하여 실행시킬 수 있는 에이전트를 생성한다.

에이전트 액션 퍼실리테이터는 에이전트가 가지고 있는 액션을 등록, 검색하고 관리한다. 이동 에이전트 환경에서는 에이전트가 새로운 에이전트 플랫폼으로 이동할 수 있다. 따라서 본 논문에서 제안하는 동적 에이전트 생성을 위해서는 이주해 온 에이전트의 액션을 등록하거나, 이주해 간 에이전트의 액션을 삭제하는 등의 액션 관리 기능이 필요하다. 이러한 관리 기능을 에이전트 액션 퍼실리테이터에서 제공한다. 에이전트 액션 퍼실리테이터는 에이전트가 가지고 있는 액션을 관리하기 위한 데이터베이스를 포함한다. 이 데이터베이스에 저장되는 액션 정보의 레코드는 표 1과 같은 형태로 저장된다.

<표 1> 에이전트 액션 퍼실리테이터에 저장되는 액션 레코드 형태

번호	클래스	메서드
1	kr.ac.skku.dclab.agent.DoorAgent	lock
2	kr.ac.skku.dclab.agent.BoilerAgent	boilingWater
3	kr.ac.skku.dclab.agent.BathtubAgent	fillingBathtub
⋮	⋮	⋮
n	kr.ac.skku.dclab.agent.PhoneAgent	toggleARS

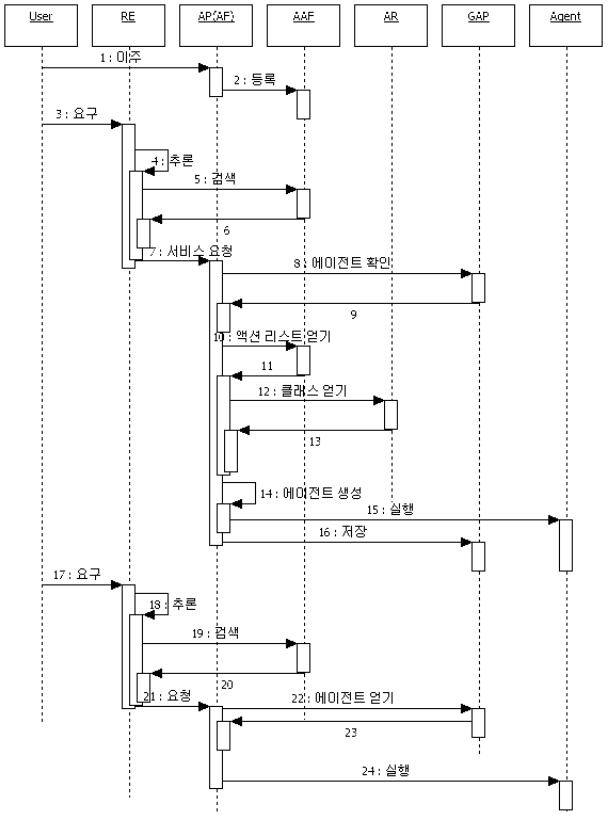
에이전트 풀은 한번 생성된 에이전트를 저장해 두는 역할을 한다. 이를 통해 이미 생성된 에이전트에 대응하는 요청이 들어왔을 경우, 에이전트를 새로 생성하지 않더라도 저장된 에이전트 객체를 불러와 곧바로 서비스를 제공할 수 있게 해 준다.

그리고 이동 에이전트 시스템을 지원하기 위한 요소로써 에이전트 관리를 위한 에이전트 관리 시스템, 에이전트의 서비스 정보를 관리하는 디렉토리 퍼실리테이터, 메시지 통신을 위한 메시지 전송 시스템이 에이전트 플랫폼에 포함된다[5].

3.2. 시나리오

이번 장에서는 본 논문에서 제안하는 기법의 서비스 시나리오를 통해 본 기법의 시스템이 어떻게 동작하여 서비스가 제공되는지를 설명한다.

그림 3의 시나리오는 직장인 독신 남성 사용자가 퇴근 후 귀가하여 목욕을 하고, 다음날 역시 귀가하여 목욕을 하는 상황을 기준으로 한다.



(그림 2) 사용자의 귀가 후 목욕 시나리오

1~2. 사용자가 귀가하여 집에 들어서자 사용자의 PDA에서 동작하던 전화기 에이전트가 자신을 복제하여 집의 에이전트 플랫폼으로 이주한다. 그리고 자신이 가진 기능인 전화기를 자동 응답 모드로 전환하는 기능을 에이전트 액션 퍼실리테이터에 등록한다.

3. 사용자가 옷을 벗고 “목욕”을 하고 싶다는 의사를 표현한다.

4~6. 추론 엔진은 “목욕” 이라는 요청을 받아 에이전트 액션 퍼실리테이터를 참고하여 물 데우기, 문 잠그기, 욕조에 물 채우기, 전화기를 자동 응답 모드로 전환하기의 4가지 액션의 구성으로 추론한다.

7. 결정된 액션의 구성을 바탕으로 에이전트 플랫폼(에이전트 팩토리)에게 서비스를 요청한다.

8~9. 에이전트 풀에 동일한 기능을 수행하는 에이전트가 존재하는지 확인해 보니, 존재하지 않는다는 응답을 받았다.

10~11. 에이전트 액션 퍼실리테이터에서 각 액션의 클래스와 메서드 정보를 받아온다.

12~13. 에이전트 액션 퍼실리테이터로부터 받은 클래스 정보를 바탕으로 에이전트 저장소로부터 클래스를 불러온다.

14. 액션 정보와 클래스 정보를 조합하여 새로운 에이전트를 생성한다.

15. 생성된 에이전트로 서비스를 제공한다.

16. 생성된 에이전트를 에이전트 풀에 저장한다.

17. 다음날이 되어, 사용자가 다시 귀가하여 “목욕”을 하고 싶다는 의사를 표현한다.

18~20. 추론 엔진은 “목욕” 이라는 요청을 받아 에이전트 액션 퍼실리테이터를 참고하여 물 데우기, 문 잠그기, 욕조에 물 채우기, 전화기를 자동 응답 모드로 전환하기의 4가지 액션의 구성으로 추론한다.

21. 결정된 액션의 구성을 바탕으로 에이전트 팩토리에게 에이전트 생성을 요청한다.

22~23. 에이전트 풀에 동일한 기능을 수행하는 에이전트가 존재하는 것을 확인하여 이를 불러온다.

24. 불러온 에이전트로 서비스를 제공한다.

위의 시나리오에서 각 에이전트가 가지고 있는 기능은 각 에이전트가 독립적으로 기능을 수행했을 때와 동일하게 작동한다. 따라서 물을 데우는 에이전트가 현재 상황을 고려하여 물의 온도를 조절하는 적응성을 지니고 있다면 위의 시나리오에서도 역시 동일한 적응성을 나타낸다.

4. 구현

자바 언어가 제공하는 기능인 리플렉션을 이용하여 본 논문에서 제안하는 기법을 구현해 보았다. 자바 리플렉션은 불러온 클래스 정보를 바탕으로 메서드를 호출을 가능하게 해 준다[6]. 여기서는 구현한 코드 중에서 중요한 부분인 서비스를 요청하는 메서드, 에이전트를 생성하는 메서드, 생성된 에이전트에 구성된 기능을 자바 리플렉션을 이용하여 실행시키는 메서드를 보이고 설명한다.

그림 3은 에이전트 플랫폼의 service 메서드 코드

를 보인다. service 메서드는 요청 정보를 인자로 받아 에이전트 팩토리로부터 에이전트를 생성하여 이를 통해 서비스를 제공하는 역할을 한다.

```
public void service(Request request) throws Exception {
  Agent agent = null

  if(pool.containsKey(request.getKey())) {
    agent = pool.get(request.getKey());
    agent.execute();
  } else {
    agent = AgentFactory.getInstance().createAgent(request);
    agent.execute();
    pool.put(request.getKey(), agent);
  }
}
```

(그림 3) 에이전트 플랫폼의 service 메서드

그림 4는 에이전트 팩토리에서 에이전트를 생성하는 createAgent 메서드의 코드를 보인다. createAgent 메서드는 사용자의 요청에 대응하는 일련의 액션 정보를 에이전트 액션 퍼실리테이터로부터 받아온다. 그리고 받아온 액션 정보를 포함하는 새로운 에이전트를 생성하여 반환한다.

```
public Agent createAgent(ArrayList<Integer> actionSeqList) {
  Agent agent = null;
  AgentActionFacilitator aaf = AgentActionFacilitator.getInstance();
  ArrayList<Action> actionList = aaf.getActionList(actionSeqList);
  agent = new Agent(actionList);
  return agent;
}
```

(그림 4) 에이전트 팩토리의 createAgent 메서드

그림 5는 생성된 에이전트에 할당된 액션을 실제로 실행하는 execute 메서드의 코드를 보인다. execute 메서드는 에이전트가 가지고 있는 액션 정보를 바탕으로 자바 리플렉션을 이용해 할당된 기능을 수행한다.

```
public void execute() {
    for(int i=0; i<actionList.size(); i++) {
      Class<?> cls =
Class.forName(actionList.get(i).getClassName());
      Method method =
cls.getMethod(actionList.get(i).getMethodName());
      Object targetObject =
Class.forName(actionList.get(i).getClassName()).newInstance();
      method.invoke(targetObject);
    }
}
```

(그림 5) 에이전트의 execute 메서드

5. 결론 및 향후 연구 과제

본 논문에서는 에이전트 기반 유비쿼터스 환경에서 사용자의 추상적인 요구에 대응하는 서비스 에이전트를 동적으로 생성하여 서비스를 제공하는 기법을 제안하고, 그 구현을 보였다. 본 논문에서 제안하는 기법을 사용하면 에이전트 기반의 유비쿼터스 환경에서 동적으로 새로운 에이전트를 생성할 수 있다. 그리고 이렇게 생성된 에이전트는 사용자의 추상적인 요구에 대응하는 서비스를 동적으로 제공할 수 있다.

향후에는 각 액션간의 의존성이나 결합관계를 고려하여 에이전트를 생성하는 방법에 대한 연구가 필요하다.

참고문헌

[1] Ángel Jiménez Molina, Hyung-Min Koo and In-Young Ko, “A Template-Based Mechanism for Dynamic Service Composition Based on Context Prediction in Ubicomp Applications,” IWBT-07, Vol. 302, pp. 1-8, 2007

[2] A. Aneiba and J. S. Rees, “Mobile Agent Technology and Mobility,” Proc. of the 5th Annual Post graduate Symposium on the Convergence of Telecommunications, Net-working and Broadcasting, 2004.

[3] V. A. Pham and A. Karmouch, “Mobile Software Agents: An Overview,” IEEE Communications Magazine, Vol. 36, Issue 7, pp. 26-37, July, 1998.

[4] J. Rao and X. Su. “A Survey of Automated Web Service Composition Methods,” Proc. of the First International Workshop on Semantic Web Services and Web Process Composition, San Diego, California, USA, July 6th, 2004.

[5] FIPA Abstract Architecture Specification, SC00001L, SC00023, http://www.fipa.org

[6] Sun, http://java.sun.com