상황 정보를 이용한 서비스 제공 모델*

이건수[1], 김민구[2]
아주대학교 정보통신 전문 대학원[1]
lks7256@ajou.ac.kr[1]
아주대학교 정보 및 컴퓨터 공학부[2]
minkoo@ajou.ac.kr[2]

Service Selection Model using Situation in Ubiquitous Computing Environment

Keonsoo Lee[1], Minkoo Kim[2]
Graduate School of Information and Communication, Ajou University,
Suwon, Kyonggido 442-749, Republic of KoreaI[1]
College of Information & Computer Engineering, Ajou University,
Suwon, Kyonggido 442-749, Republic of Korea

요 약

본 연구는 다중 사용자가 존재하는 환경에서 각 사용자의 요구를 만족시킬 수 있는 서비스를 탐색하고, 선택된 서비스를 또 다른 사용자에게 제공되는 서비스와의 충돌 없이 제공하기 위한 서비스 선택 모델을 제안한다. 이 모델은 개별 사용자의 프로파일을 관리하는 사용자 에이전트(User Agent), 환경 정보를 인지하는 센서 매니저(Sensor Manager), 현 환경에 존재하는 기기들의 상태를 관리하는 장치 매니저(Device Manager), 그리고 개별 서비스들 사이의 충돌을 회피하고 서비스를 선택하는 서비스 브로커 (Service Broker) 의 4 가지 타입의 에이전트로 구성되어있다. 사용자 에이전트는 사용자의 과거 행동 정보로부터 사용자의 선호 패턴 및 프로파일을 추출해 낸다. 센서 매니저는 현재 환경에 존재하는 센서들로부터 전달받은 환경 정보를 조합함으로써, 현재 상황을 특징 짓는다. 사용자 에이전트와 센서 매니저로부터 현재 사용자의 특성과 현재 상황 정보를 전달 받은 서비스 브로커는 그 사용자가 현재 상황에서 어떠한 서비스를 필요로 하고 있는지 판단한다. 서비스 선택은 인지된 현재 상황에서 수행 가능한 서비스 목록 중에서, 사용자의 선호도 및 프로파일에 의해 적절한 서비스가 선택 된다. 이렇게 선택된 서비스는 그 서비스를 제공하기 위한 장치들의 작동을 제어함으로써 사용자에게 제공된다. 서비스를 위한 장치를 선택하고, 작업을 할당하기 위해 서비스 브로커는 장치 매니저를 통해 현재 사용 가능한 장치들의 상태와 제공하려는 서비스와 충돌을 일으킬 수 있는 장치들의 상태 정보를 받아와 서비스를 위한 최적의 장치들을 선택하고 동작에 대한 제어 정보를 작성한다. 이렇게 만들어진 서비스 제공 계획은 장치 매니저에게 전달되고, 장치 매니저는 계획에 따라 실제 장치들을 제어한다. 이상의 다중 에이전트 모델을 통해, 특정 상황에 존재하는 사용자 개개인에게 특성화된 서비스를 충돌 없이 제공할 수 있다.

Keyword : Ubiquitous Computing, Service Selection, Situation-aware, Multi-Agent

1. 서 론

유비쿼터스 컴퓨팅 환경에서 사용자는 네트워크에 시간, 장소 그리고 공간에 제약 없이 접근할 수 있어야 한다 [5]. 이처럼 사용자와 네트워크가 제약 없이 연계되어 있는 환경에서, 사용자는 자유롭게 네트워크 자원을 사용할 수 있을 뿐더러, 컴퓨터 시스템이 주체가 되어 특정 서비스를 사용자에게 제공할 수도 있다. 이는 곧 사용자의 명시적

* This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

인 명령 없이도, 제공 가능한 서비스를 적재 적소에서 능동적으로 제공할 수 있음을 의미한다 [3]. 즉, 자유로운 네트워크 접근이 보장되는 상황에서, 사용자의 편의를 높여주는 서비스를 자동으로 적절하게 제공해 주는 것이 오늘날 유비쿼터스 컴퓨팅 환경에 대한 기대이다. 이처럼 사용자가 필요로 하는 서비스를 적절한 상황에 자동으로 제공하기 위해서는 현재 상황을 올바르게 인식하고, 각 사용자에게 필요한 서비스가 무엇인지를 판단하고, 선택된 서비스를 오류 없이 수행하는 기능이 필요하다. 이러한 일련의 기능들이 올바르게 동작하기 위해서는 사용자의 특성, 환경 정보 그리고 사용 가능한 기기들의 상태 정보들 사이의 유기적인 관계를 파악하여, 현 상황에서 각각의 사용자에게 필요한 서비스가 무엇인지를 파악해야 할 뿐만 아니라, 서로 다른 사용자에게 동시에 제공되는 서비스들 사이의 충돌 회피방법을 고려해야 한다.

이에 본 논문에서는 현재 상황 정보를 인식해, 사용자의 욕구에 부합하는 서비스를 자동으로 제공하기 위한 유비쿼터스 서비스 모델을 제안한다. 본 모델은 사용자 에이전트(User Agent), 센서 매니저(Sensor Manager), 장치 매니저(Device Manager), 그리고 서비스 브로커(Service Broker)의 4 가지 종류의 에이전트들로 구성되어 있다. 이들 에이전트들의 협력을 통해, 특정한 사용자가 현재 처한 상황을 인지하고, 그 상황에 적합한 서비스를 선택한 뒤, 기존의 다른 서비스와 충돌이 발생하지 않도록 상황에 주어진 장치들을 사용하여 선택한 서비스를 수행할 수 있다.

2. 본 론

2-1. 관 련 연 구

상황 인식(Context-Awareness)

상황 인지의 목적은 특정 서비스를 제공하기 위해 필요한 장치들의 상황 정보를 얻고 활용하기 위함이다. 상황 정보는 각 장치(Computational Device)의 물리적, 사회적 상태를 의미한다. 가령, 극장 안에서 전화가 왔다고 가정해보자. 이러한 상황에서 핸드폰이 자동으로 진동모드로 변화한다고 할 때, 외부에서 신호가 왔다는 것, 현재 기기가 극장 안에 있다는 것, 현재 극장에서 영화가 상영 중이라는 것, 기기가 전화벨 모드라는 것이 상황정보가 되고, 자동으로 핸드폰을 진동 모드로 변화시키기 위해 이상의 정보들을 인지하는 것이 상황인식 과정이다. 일반적인 현실 환경은 너무나 복잡하기 때문에, 주어진 환경을 완벽하게 인지하기란 불가능 할 뿐만 아니라, 특정 서비스를 제공하기 위해 필요한 정보 이상을 처리하는 것은 오히려 낭비가 될 수 있다 [8].

온톨로지는 환경 모델링 과정을 위해 주로 사용된다. 온톨로지로 표현된 환경 모델에서 그 안에 존재하는 센서의 인지정보들을 관리하고, 이러한 정보를 조합하여 현재 상황을 지각할 수 있게 된다. 이 흐름은 Context Toolkit 을 비롯한 대부분의 상황 인식 어플리케이션이 사용하는 방법이다 [4]. 다만 환경을 위한 온톨로지, 인지 정보로부터 추론 가능한 상황 모델은 실제 어떤 서비스가 제공 가능한지에 따라 달라진다. 결국, 인식하려는 상황은 그 시스템이 어떠한 서비스를 제공할 수 있는가에 따라 달라지고, 어떤 상황을 인지할 것인가에 따라 환경 모델 또한 달라지게 된다. 이에 본 논문에서는 상황과 센서의 관계를 다음과 같이 정의하였다. 상황이란 제공 가능한 서비스의 시작, 종료, 유지 조건이고, 조건은 그 환경 안에 존재하는 장치의 상태, 사용자의 프로파일, 그리고 현재 동작 가능한 센서들이 인지할 수 있는 정보로 구성된다.

특성화 서비스(Personalized Service)

동일한 상황에서라도, 그 상황에 처한 모든 사용자가 동일한 서비스에 동일한 만족을 느끼지는 않는다. 현재 온도가 영상 20 도라고 할 때, 이 온도를 춥게 느끼는 사용자와 덥게 느끼는 사용자 모두를 동일한 서비스로 만족시킬 수 없다. 특성화 서비스는 이처럼 개별 사용자 각각의 특성에 기반하여 능동적으로 제공되는 서비스를 의미한다. 개

별 사용자의 특성은 그 사용자의 프로필을 기반으로 사용자 타입을 분류하거나, 그 사용자의 과거 행위로부터 사용자의 선호 형태를 추측함으로써 얻어낼 수 있다. 전자의 경우, 비교 가능한 다른 사용자 정보가 많을수록 효과적이고, 후자의 경우, 각 사용자의 과거 행적에 대한 기록이 많을수록 효과적이다.

이렇게 만들어진 특성을 통해, 현재 인식된 상황에서 어떤 서비스를 선택해야 하는지를 결정할 수 있게 된다. 이 서비스는 2 개의 계층으로 나뉘게 되는데, 상위 계층에서는 서비스의 타입을 결정짓고, 하위 계층에서는 선택된 서비스를 제공하기 위한 구체적인 작업 순서를 결정짓는다. 가령, 사용자 '갑'이 A 라는 공간에서 더워하고 있다는 것을 인지했을 때, 에어컨을 사용하여 그 공간의 온도를 낮춰주는 서비스가 자동으로 실행됐다고 가정하자. 이때, '온도를 낮춰주는 서비스'는 상위 계층의 서비스가 되고, '에어컨을 동작시키는 작업'은 하위 계층의 서비스가 된다. 이때, 인지된 상황에서 '사용자가 더워하는가?'를 판단하고, 그 사용자가 선호하는 더위를 식히는 방법이 에어컨을 사용하는 것인지, 선풍기를 사용하는 것인지, 단순히 창문을 여는 것인지를 판단하기 위해서 필요한 것이 개별 사용자의 특성이 된다.

충돌 해결(Conflict Resolution)

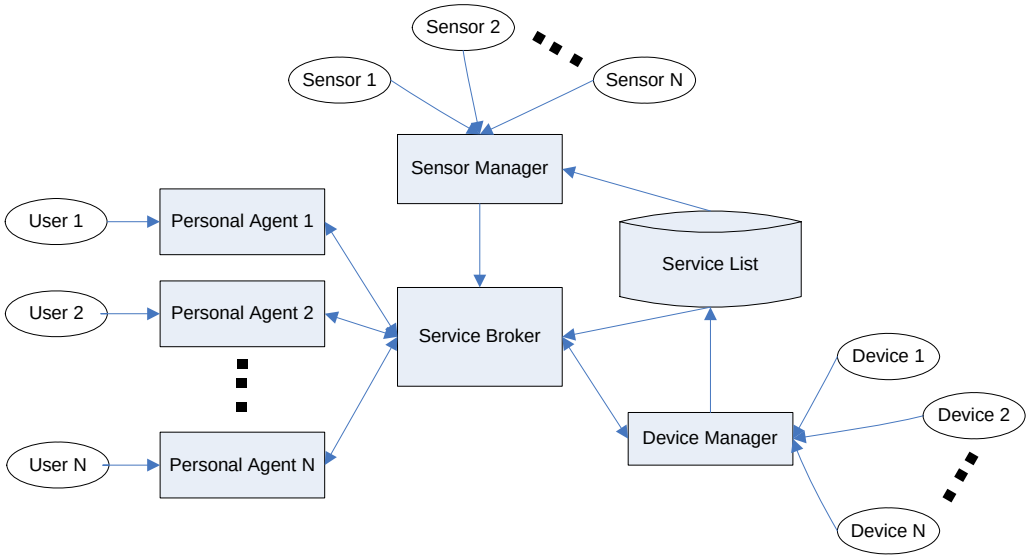
주어진 환경을 단일 사용자가 사용하지 않는 한 수행되는 서비스 사이의 충돌은 발생하기 마련이다. 불을 켜고 싶은 사람이 있으면, 불을 끄고 싶은 사람이 있고, TV 를 보고 싶은 사람이 있으면, 오디오를 듣고 싶은 사람이 있다. 이처럼, 다양한 사람들의 다양한 욕구가 존재하는 환경에서 욕구들 사이의 충돌이 발생했을 때, 할 수 있는 행동은 기다리거나, 다른 기기를 사용하거나, 다른 작업을 중지시키는 것이다. 이 3 가지 중 어느 방법을 선택할 것인가는 어느 서비스가 더 중요한 것인가에 달려있다. 현재 상황에서 우선적으로 수행해야 하는 서비스를 결정짓는 것은, 서비스 자체의 특성에 따라 결정될 수도 있고, 그 서비스를 제공 받는 사용자에 따라 결정될 수 있다. 또한 충돌이 일어나는 서비스 사이의 관계에 따라 해결 방법 또한 달라질 수 있다. 가령 응급 구조 호출 서비스는 누구의 요청인가에 상관없이 높은 중요도를 갖고 있고, 영화를 볼 것인지 드라마를 볼 것인지는 누가 영화를 보고 싶어하고, 누가 드라마를 보고 싶어하는지에 따라 결정된다. 또한 온도를 높일 것인지, 낮출 것인가에 대한 서비스의 상위 계층에서의 충돌과, 온도를 낮출 때, 어떤 기기를 어떻게 사용할 것인가에 대한 서비스의 하위 계층에서의 충돌에 따른 해결 방법 역시 달라지게 된다 [1, 2].

충돌 해결은 특정 상황에서 개별 사용자에게 제공하려는 서비스가 성공적으로 수행될 수 있는지를 보장하는 방법이 된다. 성공적인 서비스 수행이란 것은 그 서비스를 통해 대상 사용자에게는 최대의 만족감을 제공할 수 있음을 의미한다. 서비스의 우선순위는 어느 서비스가 더 큰 만족감을 제공할 수 있는가를 측정하는 기준이 되고, 이를 위해서 서비스와 서비스, 사용자와 사용자, 그리고 서비스와 사용자 사이의 유기적인 관계를 측정할 수 있어야 한다.

2-2. 제 안 모 델

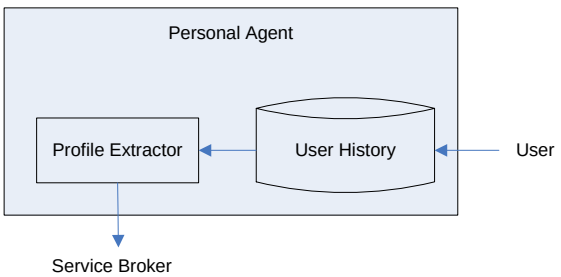
본 논문에서 제안하는 모델은 현재 상황을 인지하여 그 상황에 존재하는 개별 사용자에게 가장 적합한 서비스를 충돌 없이 제공하는 것을 목표로, 사용자 에이전트, 센서 매니저, 장치 매니저 그리고 서비스 브로커의 4 가지 타입의 에이전트들로 구성되어 있다. <그림 1>은 본 모델에서 각각의 에이전트들 사이의 관계를 보여주고 있다. 모델의 핵심 동작은 중앙의 서비스 브로커에 의해 수행된다. 서비스 브로커는 센서를 관리하고 있는 센서 매니저로부터 현재 상황에 대한 정보와 각각의 사용자 에이전트로부터 개별 사용자의 특성 정보를 얻어와 이를 토대로 어떤 서비스를 제공해야 하는지를 결정하고, 결정된 서비스를 장치 매니저를 통해 그 환경의 장치들을 사용해 선택된 서비스를 수행하게 된다. 각각의 에이전트가 수행하는 동작

은 다음과 같다.



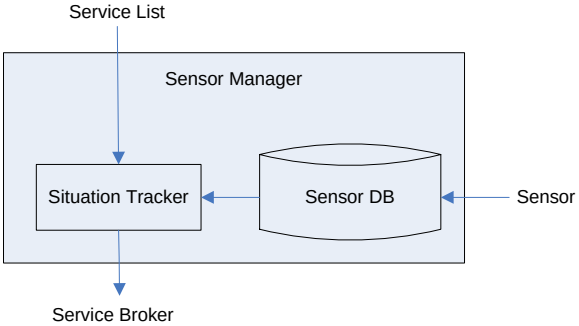
<그림 1> 제안 모델

모든 사용자는 자신의 에이전트를 통해 시스템에 접근할 수 있다. 사용자 에이전트는 기본적으로 사용자와 시스템 사이의 인터페이스 역할을 수행함과 동시에, 사용자의 프로필을 관리하는 역할을 담당한다. 사용자 프로필은 사용자의 과거 행위 기록에 기반한 행동 패턴과 에니어그램에 기반한 성격 유형, 그리고 일반적인 개인정보로 구성된다. 과거 행위 기록은 특정 상황에서 그 사용자가 어떤 서비스에 만족했는가를 패턴화 하고, 성격 유형은 제공된 서비스에 대한 피드백 및 서비스 요청 빈도에 의해 결정된다. 이러한 사용자의 특성은 추후에 서비스 충돌 발생시, 우선순위를 결정짓기 위한 요소로 사용 가능하다. 사용자 에이전트의 내부 구조는 <그림 2>에서 볼 수 있다.



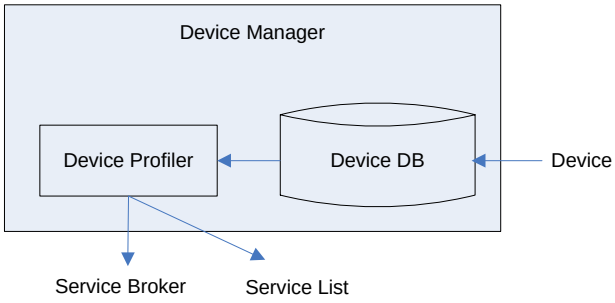
<그림 2> Personal Agent 구조

사용자 에이전트를 추출한 개별 사용자의 특성에 맞는 서비스를 제공하기 위해서, 센서 매니저는 현재 상황을 정의한다. 현재 상황은 당장 수행 가능한 서비스 목록으로 표현된다. 이를 위해 센서 매니저는 환경에 분포된 센서들로부터 환경 정보를 습득하고, 현재 인지된 정보와 등록된 서비스들의 서비스 시작 조건과 비교하여, 인지된 환경 안에서 실행 가능한 서비스 목록을 찾아준다. 이는 아무리 센서들이 정확히 환경을 인지한다고 해도, 그 환경에 적합한 서비스를 제공할 수 있는 장치가 존재하지 않는다면, 그 정보는 불필요한 낭비가 될 뿐이기 때문이다. 즉, 환경 인지의 범위는 제공 가능한 서비스가 결정하게 된다. 반면, 가능한 서비스의 실행 조건을 인지할 수 없는 경우, 그 서비스는 모든 경우에 실행 가능 서비스 목록에 첨부된다. 즉 센서 매니저는 센서들의 정보를 수집을 통해 그 상황에서 수행되어서는 안 되는 서비스를 제거하고, 서비스 브로커에게 수행 가능한 서비스 목록으로 표현되는 현재 상황을 전달해 준다. <그림 3>은 센서 매니저의 내부 구조이다.



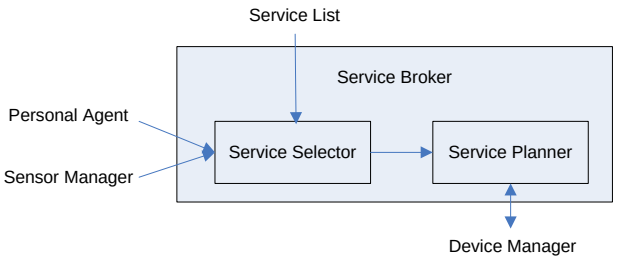
<그림 3> Sensor Manager 구조

센서 매니저가 선택하는 서비스 리스트는 장치 매니저에 의해 작성된다. 장치 매니저에 등록된 장치들의 종류에 따라, 그 환경에서 제공 가능한 서비스의 종류가 정해지기 때문이다. 장치 매니저는 자신에게 등록된 장치들의 기능에 따라, 제공 가능한 서비스 종류를 뽑아내, 서비스 리스트를 작성하고, 현재 각 장치들의 동작 상태를 관리함으로써, 특정 서비스를 수행하기 위한 장치의 사용을 조절할 수 있다. <그림 4>는 장치 매니저의 구조이다.



<그림 4> Device Manger 구조

이상의 3 가지 종류의 에이전트들은 서비스 브로커를 통해 관계를 맺게 된다. 현 상황에서 각 사용자에게 적합한 서비스를 선택하고, 선택된 서비스를 충돌 없이 실행되도록 조정하는 것이 서비스 브로커의 역할이다. 이를 위해 서비스 브로커는 센서 매니저로부터 현재 실행 가능한 서비스 목록을 받아오고, 사용자 에이전트로부터 사용자의 행동 패턴 목록과 그 사용자의 성격 유형으로 구성된 사용자 특성을 받아와, 그 사용자가 필요로 하는 서비스를 분류해 낸다. 서비스가 결정되면, 그 서비스를 장치 매니저에 등록된 장치



<그림 5> Service Broker 구조

2-3. 동 작 시 나 리 오

본 모델은 서비스 리스트에 서비스들이 등록되면서 시작된다. 서비스 리스트는 장치 매니저에 의해 채워진다. 장치 매니저는 자신에게 등록된 장치들의 기능과 그 장치의 동작 조건을 관리함으로써, 그 장치들의 사용하여 제공할 수 있는 서비스를 서비스 리스트에 등록하게 된다. 가령, 에어컨과 히터가 장치 매니저에 등록되어 있다면, 이들 장치가 제공 가능한 기능은 서비스 리스트에 각각 환경 온도를 낮추는 서비스, 온도를 높이는 서비스로 등록된다. 처음 장치가 환경에 등장할 때, 기본 동작 조건이 존재하지 않는다면, 서비스 리스트에는 시작 조건은 등재되지 않는다.

센서 매니저는 자신에게 등록된 센서들로부터 정보를 받아들인다. 센서를 통해 특정 이벤트가 발생했음을 감지하면, 센서 매니저는 서비스 리스트에 등록된 서비스들 중에 그 이벤트로 인해 동작되는 서비스가 존재하는지 판단한다. 이벤트의 발생 여부는 각각의 센서들이 감지하는 정보에 변화가 일어났을 경우, 이벤트가 발생한 것으로 본다. 이때, 서비스 리스트의 시작 조건에, 감지 가능한 환경정보 타입에 맞는 것이 없는 서비스는 항상 수행 대상 목록으로 처리된다. 이는 수행 가능한 서비스와 환경 감지 기능이 독립적으로 관리되기 때문에, 특정 서비스의 시작 조건을 감지하지 못하는 경우 그 서비스가 결코 수행되지 못하는 것을 방지하기 위함이다. 가령, 온도 조절 서비스가 존재한다고 할 때, 온도를 감지할 수 있는 센서가 존재하지 않으면, 이 서비스는 어떠한 경우에도 수행되지 못하게 된다.

센서 매니저에 의해 특정 이벤트가 발생하고, 그 이벤트에 의해 특정 목록의 서비스가 수행 가능하다는 것이 판단되면, 이 서비스 목록과, 그 서비스의 제공 대상이 되는 사용자 에이전트가 현재 상황이 된다. 이 상황 정보는 서비스 브로커에게 전달되고, 서비스 브로커는 사용자 에이전트로부터 그 사용자의 프로필을 얻어와 현재 제공 가능한 서비스 중, 필요한 서비스를 선택한다. 사용자 프로필은 그 사용자의 과거 행위 기록으로부터 추출

된 특정 행위 패턴과 그 사용자의 성격 정보로 구성되어 있다. 이 프로필에 기반하여, 행위 패턴에 부합하는 서비스가 센서 매니저로부터 받아온 목록이 존재 한다면, 그 서비스는 수행 대상이 된다. 패턴으로 등록되지 않은 서비스는 수행되지 않는다. 이는, 원치 않는 서비스가 자동으로 수행되는 것은 원하는 서비스가 자동으로 수행되는 것 이상의 불쾌감을 유발할 수 있기 때문에, 사용자의 패턴으로 등록되지 않은 서비스를 임의로 제공함에 따르는 위험 부담을 감당하지 않기 위함이다.

서비스 브로커는 수행 대상 서비스를 서비스 리스트와 비교함으로써, 그 서비스를 수행하기 위해 필요한 장치들의 목록을 검출한다. 이렇게 생성된 장치 목록은 장치 매니저를 통해 현재 상태를 확인 받고, 사용 가능한 장치가 존재한다면, 그 장치는 서비스를 받는 사용자에게 베타적으로 점유되어 사용된다. 이때, 사용 가능한 장치가 이미 다른 사용자에게 점유되어 있거나, 그 장치의 사용이 기존의 수행중인 서비스의 원활한 동작에 모순이 되는 등의 충돌이 발생하면, 충돌된 서비스들 간의 협상 과정이 수행된다.

충돌이 발생한 서비스 사이의 협상은 각 서비스의 우선순위를 비교하여 더 중요한 서비스를 먼저 수행함을 원칙으로 한다. 서비스 우선순위는 서비스 자체의 중요도와 사용자 에이전트가 갖고 있는 서비스 수혜자의 성격 및 관계에 의해 결정된다. 가령, 응급 구조 요청 서비스는 수혜자에 상관없이 높은 우선순위를 갖고, 평화주의자의 성격 유형을 갖고 있는 사용자의 서비스는 열정적인 성격 유형의 사용자의 서비스보다 높은 우선순위를 갖는다. 또한, 두 사용자의 위계 관계가 존재하는 경우, 상위 계층의 사용자에 서비스가 먼저 수행될 수 있다. 이들 우선순위를 결정짓는 요소들의 가중치를 어떻게 두고, 어떤 규칙을 통해 우선순위를 구할 것인가 하는 것은 본 모델이 적용되는 실제 시스템에 따라 달라질 수 있다.

3. 결 론

유비쿼터스 컴퓨팅 환경에서 사용자는 현재 상황에 부합하여 자신에게 특성화된 서비스가 자동으로 제공되기를 기대한다. 이러한 서비스를 제공하기 위해서는 상황을 인지하고, 그 상황에서 개별 사용자가 원하는 서비스가 무엇인지를 판단하고, 다른 사용자의 요구에 반하지 않으면서 선택된 서비스를 수행하기 위해 주위의 기기를 효율적으로 사용하고, 사용자의 기대를 충족시킨 뒤, 자동으로 종료할 수 있어야 한다. 본 논문에서 제안하는 모델은 이상의 작업을 수행하기 위하여 사용자 에이전트, 센서 매니저, 장치 매니저, 그리고 서비스 브로커의 4 가지 타입의 에이전트들을 사용한다.

Reference

1. K. Lee, W. Kim, M. Kim "Resource Allocation in Multi-Agent System for Ubiquitous Computing Service" PRIMA 2005 Malaysia pp.113-120
2. K. Lee, M. Kim "Conflict Resolution Method for Multi-Context Situation" PRIMA 2005 Malaysia pp.285-294
3. Anind K. Dey, Peter Ljundstrand and Albrecht Schmidt, "Distributed and Disappearing User Interfaces in Ubiquitous Computing" CHI 2001, Seattle, WA, March 31 - April 5, 2001.
4. Anind K. Dey and Gregory D. Abowd "The Context Toolkit: Aiding the Development of Context-Aware Applications " In the Workshop on Software Engineering for Wearable and Pervasive Computing , Limerick, Ireland, June 6, 2000.
5. Weiser, M. "The computer for the 21st century" Scientific American 265(3), 1991, pp 66-75
6. Schilit, W.N., "System architecture for context-aware mobile computing" ph.D Thesis, Columbia University, May 1995
7. Dey, A.K., Abowd, G.D, and Salber, D. A, "A Context-based Infrastructure for Smart Environments", MANSE'99 pp. 114-128