지능공간을 위한 사용자주도형 상황인식서비스 개발

박정규*, 이긍해*
*한국항공대학교 컴퓨터공학과
e-mail: fcopark@kau.ac.kr, khlee@kau.ac.kr

User-driven Context-aware Service Development for Smart Space

Jeongkyu Park*, Keung Hae Lee*
*Dept of Computer Science, Korea Aerospace University

요 약

상황인식은 사용자의 상황을 분석하고 이에 적합한 서비스를 자동으로 제공하는 컴퓨팅 기술이다. 상황인식에 관한 대부분의 기존 연구들은 개발자가 만든 서비스를 사용자에게 공급하는 전통적인 개발 모델을 따르고 있다. 이런 일방적인 모델은 원하는 서비스가 서로 다른 다양한 사용자들의 요구를 만족시키기 어려울 뿐만 아니라 상황인식서비스가 설치되는 각 지능공간의 특징을 반영하기도 어렵다. 사용자와 대상 컴퓨팅 환경을 고려하지 않고 서비스를 개발하고 배포하는 것은 어려운 일이다. 본 논문에서는 사용자가 자신의 지능공간을 구성하는 기기들의 기능을 활용해 직접 원하는 서비스를 개발할 수 있도록 하는 새로운 상황인식서비스 개발 모델을 제안한다. 기존에도 이와 유사한 방법을 제안한 연구들이 몇몇 있었으나 기기의 일부 기능 간 연동만을 지원해 확장성이 낮았다. 본 연구는 독립된 다수 기기의 모든 기능을 사용자가 자유롭게 연동할 수 있게 한다는 점에서 이런 연구들 문제를 개선하였다. 본 논문에서는 지능공간을 위한 사용자주도형 상황인식서비스 개발을 지원하는 도비(Dobby) 시스템을 소개하고 이를 구현한 결과를 설명한다.

1. 서론

정보통신기술의 발달로 초소형 고성능의 컴퓨팅 기기들이 많이 등장함에 따라 이들을 이용해 인간의 삶을 더욱 편리하고 안전하게 만들고자하는 상황인식기술에 대한 관심이 높아졌다. 상황인식을 통해 제공되는 상황인식서비스는 누구에게나 유용한 범용 서비스와 일부 사용자에게만 유용한 개인화된 서비스로 구분할 수 있다. 안전과 관련된 일부를 제외한 많은 서비스들은 사용자에 따라 선호도가 서로 다른 경우가 흔히 있다. 한 사람에게 유용한 서비스도 다른 사람에게는 불편한 서비스가 될 수 있다. 상황인식에 관한 대부분의 기존 연구들은 개발자가 만든 서비스를 사용자가 일방적으로 이용하는 개발자주도형 모델을 따르고 있다. 이런 방식은 개개인의 서로 다른 요구를 만족시키기 어렵다. 이뿐만 아니라 이런 모델은 사용자마다 서로 다른 컴퓨팅 환경을 지원하기 어렵다. 사용자 각각은 생활하는 공간과 소유한 기기들이 서로 다르다. 이는 상황인식서비스가 실행될 플랫폼이 사용자마다 서로 다르다는 것을 의미한다. 이처럼 사용자와 실행플랫폼에 대한 이해 없이 서비스를 개발하고 배포하는 것은 어려운 일이다. 이를 해결하는 방법으로 사용자가 주변 기기를 이용하여 원하는 서비스를 직접 정의하도록 지원하는 몇몇 연구가 제안된 바 있다. 이런 방법을 본 논문에서는 사용자주도형 모델이라 부른다. 기존에 제안된 사용자주도형 모델들은 일부 정해진 기능만을 연동할 수 있어 그 확장성이 낮다는 한계가 있다.

본 논문에서는 사용자가 자신이 원하는 서비스를 주변 기기의 기능들을 연동하여 스스로 만들 수 있게 하는 도비(Dobby)시스템을 소개한다. 우리 주변에는 다양한 기능을 제공하는 여러 기기들이 있다. 사용자가 이런 기기들의 기능을 간단한 방법으로 자유롭게 연동할 수 있게 한다면 사용자가 원하는 상황인식서비스를 직접 정의하는 것이 가능하다. 예를 들어 어떤 사용자가 TV와 전화기를 가지고 있다고 하자. 그 사용자는 이 두 기기에서 제공하는 기능들을 모두 소유하고 있다. 하지만 “전화기가 울리면, TV 소리를 소거한다.”와 같이 두 기기의 기능을 연결하여 새로운 서비스를 만들지는 못한다. 본 논문에서는 사용자가 자신이 소유한 주변 기기들의 기능을 연동하여 원하는 서비스를 직접 정의할 수 있도록 하는 도비(Dobby)를 분석한 결과와 간단한 테스트베드를 소개한다.

2. 기존 연구

개발자주도형 모델의 대표적인 연구로 OSGi[1], RCSM[2], SOCAM[3], Aura[4] 등을 들 수 있다. OSGi는 자바 프로그래밍 언어를 사용하여 서비스를 개발하고, RCSM은 C, C++, 자바 등 다양한 프로그래밍 언어를 통해 서비스를 정의한다. SOCAM과 Aura는 자신들이 개발한 고유의 프로그래밍 언어를 통해 서비스를 정의하는 방법을 제안하고 있다. 이런 연구들은 개발자가 서비스를 만들고 사용자가 이를 이용하는 일방적인 모델을 따르고 있

어 사용자의 다양한 요구사항을 반영하기 힘들다. 사용자 주도형 모델의 대표적 관련 연구로는 JIGSAW[5]와 a CAPpella[6]가 있다. JIGSAW는 이해하기 쉬운 그래픽언어로, 서비스도메인에서 발생될 수 있는 상황을 함축된 그림으로 표현한 퍼즐조각들을 나열함으로써 원하는 상황인식서비스를 정의한다. a CAPpella는 “Program by demonstration”을 통한 상황인식서비스 정의를 제안하고 있다. 이 방법에서는 사용자가 새로운 서비스를 정의하기 위하여 센서들이 설치된 개발시스템 앞에서 정의하고자 하는 상황을 반복적으로 시연한 후, 그 기록을 보고 서비스와 관련된 상태들을 선택하고 이에 대응되는 액션을 정의함으로써 서비스를 만든다. JIGSAW와 a CAPpella는 정해진 기기의 일부 기능만을 사용자가 연동할 수 있도록 한다는 점에서 사용자가 원하는 임의의 서비스를 정의할 수 있도록 하는 확장석이 낮다는 한계가 있다.

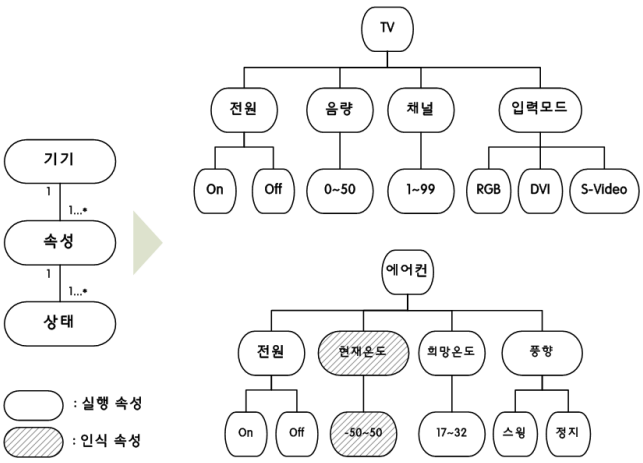
3. 도비

3.1 상황인식서비스

상황인식서비스는 하나 이상의 이벤트 x가 발생하면 이에 대응하여 하나 이상의 액션 y를 수행하는 것으로써 <x, y>로 표현할 수 있다. 이때 하나의 이벤트는 한 기기의 상태 변화로 정의된다. 예를 들어, “TV가 켜짐”이라는 이벤트는 TV의 상태가 꺼짐에서 켜짐으로 변경되었다는 것을 의미한다. 이처럼 기기가 이전의 어떤 상태에서 현재의 상태로 바뀌는 것을 이벤트라 한다. 이벤트에 대응해서 실행되는 액션은 한 기기의 현재의 상태를 사용자가 원하는 다음의 상태로 변경하는 것으로 정의할 수 있다. 이처럼 상황인식서비스를 구성하는 이벤트와 액션은 상태로 표현할 수 있다. 이는 사용자가 기기의 상태를 인식하고 이들을 조합하여 이벤트와 액션을 명세할 수 있게 하면, 사용자가 상황인식서비스를 직접 정의하는 것이 가능해진다는 것을 의미한다. 본 논문은 이를 기반으로 사용자가 필요한 서비스를 스스로 정의하는 방법을 제안한다.

3.2 기기 인식과 서비스 정의

앞서 상황인식서비스를 기기 상태의 조합으로 정의할 수 있다는 점을 살펴보았다. 본 논문은 이를 지원하기 위해 SEAD (State-based Event and Action Description)를 제안한다. SEAD는 상태를 기반으로 기기의 기능을 사용자가 인식할 수 있고 이를 통해 이벤트와 액션을 명세함으로써 상황인식서비스를 정의하도록 지원하기 위한 방법이다. 그림1은 SEAD와 그 사용 예를 보여준다. SEAD는 하나의 기기를 하나 이상의 속성의 집합으로 표현한다. 그림1의 예에서 TV라는 기기는 "전원", "볼륨", "채널", "입력모드"라는 속성을 가지고 기기 에어컨은 "전원", "현재온도", "희망온도", "풍향" 이라는 속성을 가진다. 하나의 속성은 한 개 이상의 상태를 가진다. TV의 전원속성은 "On"과 "Off"라는 두 상태를 가지고 있고, 볼륨속성 "0"부터 "50"까지의 정수로 51가지 상태를 가지고 있다. 속성은 실행속성과 인식속성으로 구분된다. 실행속성은 사용자가 그 상태를 직접 제어할 수 있는 것이고, 인식속성은 사용자가 그 상태를 직접 제어할 수 없는 것이다. 예를 들어, 에어컨의 속성인 현재온도는 정보를 읽어 오는 것은 가능하지만 이를 직접 제어할 수는 없다.



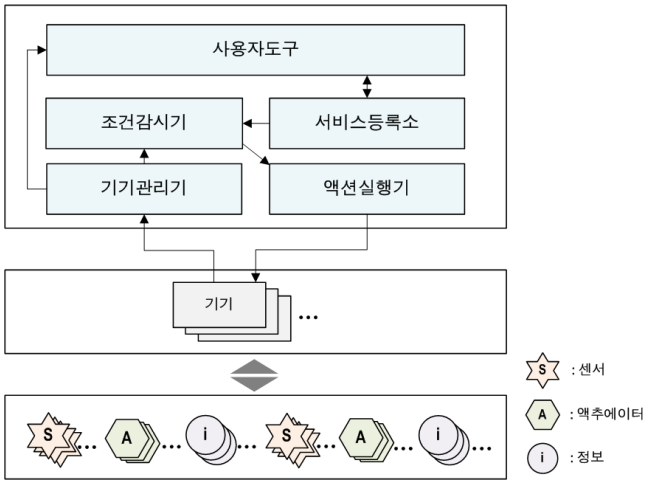
(그림 1) SEAD

SEAD를 이용해 이벤트나 액션을 명세할 수 있다. 예를 들어 에어컨의 기능이 그림1에서와 같이 정의될 때 실내온도가 30도가 된다는 이벤트는 "에어컨-현재온도-30"이라는 기기, 속성, 상태의 조합으로 표현할 수 있다. 또한, 에어컨의 켜고 희망온도를 24도로 설정하라는 액션은 각각 “에어컨-전원-On", "에어컨-희망온도-24”으로 표현할 수 있다. 이처럼 사용자는 SEAD를 통해 “현재 실내온도가 30도가 되면 에어컨을 켜고 희망온도를 24도로 설정하라는 상황인식서비스를 쉽게 정의할 수 있게 된다. SEAD는 그림1에서 볼 수 있듯이 한 기기의 기능을 직관적으로 쉽게 이해할 수 있기 때문에 API문서와 같이 해당 기능을 호출하기 위한 별도의 설명이 필요하지 않다는 장점이 있다.

3.3 아키텍처

우리 주변에는 많은 수의 센서와 액추에이터가 설치되어 있으며 그 수는 점점 증가하고 있다. 이런 장치들은 설치, 이용, 관리가 용이한 기기의 단위로 배치되고 있다. 이처럼 기기의 기능이 확장됨에 따라 이를 조합하여 상황인식서비스를 개발하는 것이 가능해진다. 우리는 이런 점을 발견하고 이를 지원하는 시스템으로 도비를 개발하였다. 그림2는 도비의 전체 아키텍처를 보여준다. 상황인식서비스는 두 개 이상의 독립된 기기들의 협력을 통해 서비스를 제공한다. 이런 이유로 상황인식서비스를 지원하는 시스템들은 대부분 그림2와 유사한 아키텍처 구조를 가진다. 상황인식서비스를 제공하기 위해서는 우선 서로 다른 기기들을 일관되게 인식하고 관리하는 방법이 필요하다. 특정 지능공간 내에 가용한 모든 기기들과 인터넷 상의 정

보기기들은 도비의 기기관리기에 등록된다. 등록이란 기기들이 자신이 어떤 기능이나 정보를 제공하는지 도비에게 알려주는 것을 의미한다.



(그림 2) 도비 아키텍처

등록된 이후에 기기는 특정 기능이 실행되거나 제공되는 정보가 변경되면 이를 즉시 보고한다. 기기를 발견하고 등록하는 과정은 UPnP[7], Jini[8] 등 다양한 서비스발견 기술로 구현 가능하기 때문에 본 논문에서 상세히 다루지는 않기로 한다. 사용자는 도비에 등록된 기기에서 제공하는 기능을 조합함으로써 상황인식서비스를 정의한다. 사용자도구는 사용자가 기기를 확인하고 이를 조합하여 서비스를 정의할 수 있게 하는 인터페이스를 제공한다. 정의된 서비스는 서비스등록소에 저장된다. 조건감시기는 기기관리기로부터 실시간으로 기기들의 상태를 전달받고 이를 토대로 서비스등록소에 저장된 서비스 중 실행조건을 만족하는 것을 선별한다. 만약 실행조건이 만족된 서비스가 있다면 액션실행기에게 이에 대응되는 액션을 실행하라고 해당 기기에 지시함으로써 사용자에게 서비스를 제공한다.

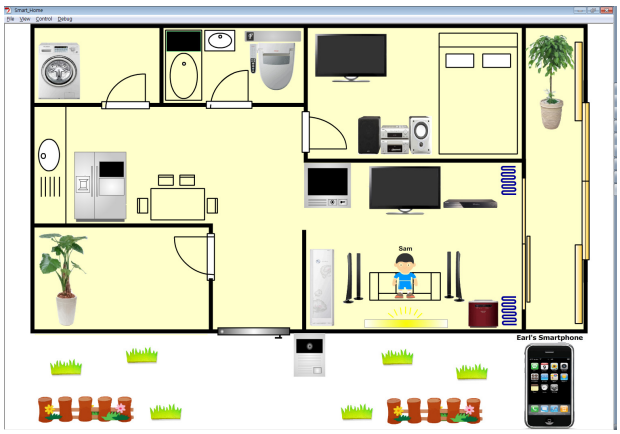
3.4 후보선택방식의 서비스정의

도비의 핵심은 일반 사용자도 쉽게 상황인식서비스를 정의할 수 있도록 하는 것이다. 일반적으로 서비스는 프로그래밍을 통해 개발된다. 프로그래밍은 개발하고자 하는 기능을 코드로 서술하는 작성하는 방법이다. 이런 서술적 방법은 데이터타입, 연산자, 타입캐스팅 등과 같이 프로그래밍 언어에 대한 배경지식을 필요로 하기 때문에 일반인에게는 너무 어렵다. 본 논문에서는 사용자가 서비스를 보다 쉽게 정의하는 방법으로 후보선택(candidate selection) 방식을 제안한다. 후보선택방식의 서비스정의는 서비스정의 과정을 여러 단계로 나누어 각 단계에서 다음 단계에 올 수 있는 후보들 중 하나를 선택해 가는 방식이다. 예를 들어, 앞서 살펴본 “실내온도가 30도가 되면 에어컨을 켜고 희망온도를 24도로 설정하라.”라는 서비스를 정의한다고 가정하자. 이를 정의하기 위해 사용자가 도비의 사용자 도구에서 에어컨을 선택하면 도비는 에어컨의 모든 속성인 전원, 현재온도, 희망온도, 풍향을 출력하고 사용자가 이 중 원하는 속성인 현재온도를 선택하면 도비는 가능한 현재온도의 상태인 -50부터 50사이의 값을 출력하여 이 중 하나를 사용자에게 선택하도록 한다. 이처럼 선택의 연속으로 이벤트와 이에 대응하는 액션을 정의할 수 있도록 하는 것이 후보선택방식이다. 후보선택방식은 데이터타입과 타입 간 호환, 데이터타입별 가능 연산 등과 같은 전문지식 없이도 시스템의 리드에 따라 원하는 서비스를 정의할 수 있다. 또한, 구두점을 찍지 않거나 오타를 치는 것과 같이 사소하지만 자주 발생하는 실수를 예방한다는 장점이 있다.

4. 구현

4.1 테스트베드

그림 3은 도비의 테스트베드인 가상 스마트홈의 모습을 보여준다.



(그림 3) 가장 지능형홈

가상 스마트홈은 플래시로 구현되었다. 가상의 스마트홈에는 TV, DVD플레이어, 전동커튼, 오디오, 냉장고, 공기청정기 등 여러 객체들이 설치되어 있다. 이 모든 객체들은 기본적인 기능조작이 가능하도록 구현되었고 모두 도비미들웨어에 등록되어 있다. 이뿐만 아니라 이벤트가 발생하면 즉시 도비미들웨어에게 전달하고, 도비미들웨어의 지시에 따라 객체의 기능을 실행할 수 있도록 설계되었다.

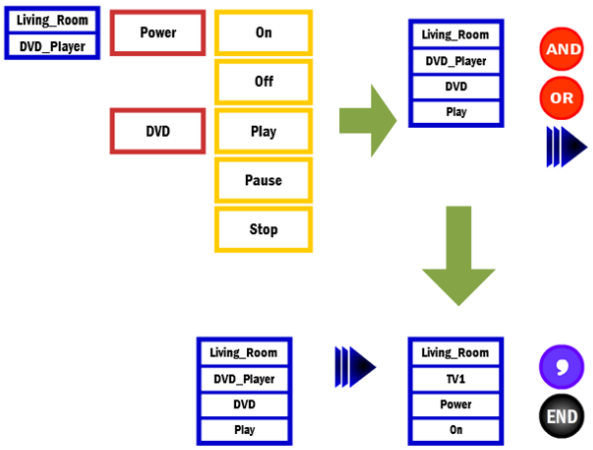
4.2 도비미들웨어

도비미들웨어는 지능형홈의 객체와 그 기능을 등록받고 객체의 상태변화를 보고 받아 사용자가 정의한 시너지 서비스를 언제 실행할지 결정한다. 도비미들웨어는 명세된 시너지서비스를 실행하기 위한 토근분석기, 구문분석기, 의미분석기를 포함하고 있고, Eclipse IDE for Java를 사용하여 구현하였으며, 네트워크를 통해 가상 지능형홈과

통신한다.

4.3 도비 사용자도구

도비 사용자도구는 후보선택방식을 통해 사용자가 보다 쉽게 서비스를 정의할 수 있도록 한다. 그림4는 사용자도구를 통해 상황인식서비스를 정의하는 예를 보여준다.



(그림 4) 사용자도구와 후보선택방식의 예

그림4에서 사용자는 거실에 설치된 DVD플레이어가 재생되면 이와 연결된 TV1을 자동으로 켜주는 서비스를 정의하고자 한다. 사용자가 거실과 DVD플레이어를 나타내는 아이콘을 차례로 이어 그림1과 같이 "DVD Play"를 선택하면 사용자도구는 그 다음으로 올 수 있는 가능 후보인 논리연산자와 액션정의연산자를 출력한다. 이벤트정의를 마치고 액션을 정의하고자 하는 사용자는 액션정의연산자 심볼을 선택하고 거실의 TV의 전원이 켜지는 액션을 후보선택방식에 의해 정의한다. 지면의 제약으로 모든 단계를 하나하나씩 보여줄 수는 없으나 도비는 이런 과정을 통해 전문지식이 없는 사용자도 도비의 리드를 통해 쉽게 원하는 서비스를 정의하도록 지원한다.

5. 결론 및 향후연구

기존 상황인식서비스 개발 모델은 크게 개발자주도형과 사용자주도형으로 나눌 수 있다. 기존에 제안된 두 모델의 방법들은 여러 사용자들의 서로 다른 요구를 만족시키기 어렵다는 공통된 한계점이 있다. 또한 개발자주도형 모델은 사용자마다 서로 다른 컴퓨팅환경에 서비스를 설치하기 어렵다는 한계를 가지고, 사용자주도형 모델은 제안된 범위에서만 서비스를 정의할 수 있다는 한계를 가진다. 본 논문에서는 새로운 사용자주도형 상황인식서비스 개발 모델을 소개하고 이를 지원하는 도비를 설명하였다. 도비는 기존 연구들과 비교하여 다음과 같은 장점을 가진다. 첫째, 사용자의 다양한 요구에 유연하게 대처할 수 있다는 점이고, 둘째는 사용자 각자의 컴퓨팅 환경에 맞출 수 있는 확장성 있는 모델이라는 점이다. 향후 우리는 다양한 상황인식서비스 시나리오를 분석하여 이들을 지원할 수 있도록 도비를 발전시킬 계획이다.

참고문헌

[1] OSGi Alliance, www.osgi.org

[2] Stephen S. Yau, Fariaz Karim, Yu Wang , Bin Wang, and Sandeep K.S. Gupta, "Reconfigurable Context-aware-Sensitive Middleware for Pervasive Computing," IEEE Pervasive Computing, Vol. 1, No.3, pp. 33-40, 2002.

[3] Tao Gu, Hung Keng Pung, and Da Qing Zhang, "A service-oriented middleware for building context-awareservices," In Journal of Network and Computer Applications, vol.28, Issue 1, Jan. 2005.

[4] Glenn Judd and Peter Steenkiste , "Providing Contextual Information to Pervasive Computing Applications," Proceedings of the First IEEE International Conference on Pervasive Computing and Communications , March 2003.

[5] Tom Rodden, Andy Crabtree, Terry Hemmings, Boriana Koleva, Jan Humble, Karl-Petter AKESSON, and Par HANSSON, "Configuring the Ubiquitous Home," The 6th International Conference on Designing Cooperative Systems, pp.11-14, May, 2004.

[6] Anind Dey, Raffay Hamid, Chris Beckmann, Ian Li, Daniel Hsu, "a CAPpella: programming by demonstration of context-aware applications," Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2004.

[7] Web page, "Understanding of universal plug and play," http://www.upnp.org

[8] Rahul Gupta, Sumeet, Talwar, Dharma P. Agrawal, "Jini Home Networking: A Step toward Pervasive Computing," IEEE Computer, Volume 35, Issue 8, pp. 34-40 Aug. 2002.