ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

사용자주도형 상황인식서비스☆

User-driven Context-aware Service

박 정 규[1] 이 긍 해[2*]
Jeongkyu Park Keung Hae Lee

요 약

상황인식은 컴퓨터가 사용자의 상황을 인식하고 자동으로 유용한 서비스를 제공하는 컴퓨팅기술이다. 상황인식에 관한 대부분의 기존 연구는 일방적으로 개발자가 제공하는 상황인식서비스를 사용자가 사용하는 개발자주도형 모델을 따르고 있다. 사용자가 원하는 서비스를 스스로 정의할 수 있는 방법이 일부 제안되기도 했으나 대체로 사용하기 어렵거나 매우 단순한 형태의 서비스 구현에 머물러 있다. 본 논문은 개인의 필요에 맞춘 상황인식 서비스 구현을 목표로, 스마트홈 환경에서 사용자가 직접 실세계의 장치나 인터넷의 정보를 연동하여 다양한 서비스를 구현할 수 있도록 개발된 도비(Dobby) 모델에 대하여 설명한다. 본문에서는 도비 모델을 상세히 설명하고 일반사용자를 위한 서비스정의 방법으로 Program by selection을 제안한다. 가상 스마트홈 기반의 테스트베드에서의 실험을 통하여 도비의 서비스 개발방법이 기존의 사용자주도형 방법 연구에서 제안된 방법보다 표현력과 사용의 편의성 면에서 우수함을 주장한다.

주제어 : 상황인식, 사용자주도형 개발, 지능형공간, 시너지서비스

ABSTRACT

Context-awareness is a computing technology that automatically delivers useful services to users based on their situation. Most previous studies on context-awareness adopted the view that the user simply is a consumer of what the developer creates. Few studies addressed catering to the need of personalized services for the user. They are either too complex for the user to grasp or unable to express many useful services due to their weak expressive power. To address these issues, we propose Dobby as a new model and architecture for user-driven context-aware service development. Dobby enables the user to create services that are more suited to his personal preferences. We argue that Dobby offers an enhanced method for defining personalized context-aware services over existing methods.

☞ keyword : Context-awareness, User-driven Development, Smart space, Synergy Service

1. 서 론

상황인식컴퓨팅은 사용자의 상태나 상황을 인식하고 분석하여 자동으로 유용한 서비스를 제공해주는 기술이다. 상황인식컴퓨팅을 이용하면 기존에 없었던 편리한 서비스들을 제공할 수 있을 뿐만 아니라 서비스를 인식하고 제어하는데 드는 사용자의 노력을 줄여줄 수 있는 장점이 있다. 상황인식컴퓨팅을 통해 사용자에게 제공되는 컴퓨팅 서비스를 상황인식서비스라고 부른다. 상황인식에 관한 대부분의 기존 연구들은 개발자가 서비스를 만들어 이를 사용자에게 제공하는 개발자주도형 모델을 따르고 있다. 개발자주도형의 상황인식서비스는 모든 사용자에게 공통적으로 유용한 서비스를 개발하는 것을 목표로 한다. 개발자주도형 서비스 모델에는 다음과 같은 두 가지 취약점이 존재한다. 첫 번째는 사용자들의 다양한 요구를 만족시키기 어렵다는 점이다. 한 사람에게 유용한 서비스가 때로는 다른 사람에게 불필요하거나 오히려 성가신 서비스가 될 수도 있다. 이처럼 개발자주도형 모델은 사용자의 개인적 필요나 선호를 만족시키기에는 한계가 있다. 두 번째로는, 사용자에 따라 큰 차이가 있는 사용자들의 다양한 컴퓨팅 환경을 서비스 구현에 밀접하게 반영하기 힘들다는 점이다. 사용자마다 자신이 소유한 장치의 종류, 제조자, 기능 등이 다를 수 있기 때

1 R&D Center, LIG System CO., Seoul, 121-888, Korea
2 Department of Computer Engineering, Korea Aerospace University, Goyang-City, 412-791, Korea
* Corresponding author (jk.park@ligcorp.com)
[Recived 18 October 2012, Reviewed 23 October 2012(R2 4 March 2013), Accepted 17 June 2013]
☆ 본 논문은 2011년 8월에 발표된 박사학위논문 "사용자주도형 상황인식서비스 개발방법 연구"의 일부를 요약한 것임.
☆ 본 논문은 한국항공대학교의 교비연구지원을 받아서 수행된 연구의 결과임.

http://dx.doi.org/10.7472/jksii.2013.14.4.01

문에 개발자가 처음부터 일괄적으로 사용자의 컴퓨팅 환경에 맞는 상황인식서비스를 제공하는 것은 어려운 일이다. 우리는 이런 문제들을 개선하기 위해서는 사용자가 자신의 기기들을 이용하여 직접 원하는 서비스를 개발할 수 있도록 해줄 수 있는 방법이 필요하다고 본다. 본 논문에서는 이와 같은 상황인식 서비스 개발 방법을 사용자주도형 방법이라 부르기로 한다. 지금까지 사용자주도형 상황인식서비스 모델이 일부 연구된 적이 있으나 기존 연구에서 제안된 사용자주도형 모델은 대체로 프로그래밍능력을 가진 컴퓨터 전문가를 사용자로 고려하고 있다. 따라서, 프로그래밍지식이 없는 일반 사용자에게는 사용하기가 너무 어려울 뿐만 아니라 구현할 수 있는 서비스의 종류도 매우 단순한 형태에 머물러 있다.

본 논문은 다음과 같은 구조로 구성된다. 2장에서는 관련연구를 살펴볼 것이고 3장에서는 본 논문에서 제안하는 사용자주도형 상황인식서비스 모델인 도비를 소개한다. 4장과 5장에서는 각각 도비의 구현과 평가를 보여주고, 마지막 6장에서는 결론 및 향후과제를 살펴본다.

2. 관련연구

상황인식서비스의 개발 방법은 크게 세가지로 분류해 볼 수 있다. 앞에서 설명한 개발자주도형 모델과 사용자주도형 모델 외에도, 컴퓨터 시스템이 사용자의 행동 데이터를 분석함으로써 반복되는 행동패턴을 자동으로 서비스화 하는 데이터기반형 모델로 나눌 수 있다. 이 세가지 개발 모델에 대한 기존 연구를 살펴보기로 한다.

첫째, 데이터기반형 상황인식서비스 모델의 대표적인 연구에는 CASPER[1], Episode[2], MavHome[3] 등이 있다. CASPER는 인간 행동 기록을 이벤트열(event sequence)로 추상화하고 인과관계가 있는 패턴을 인식하여 이를 상황인식서비스로 정의한다. Episode는 데이터마이닝 기법을 이용하여 반복되는 인간의 생활패턴을 인식함으로써 상황인식서비스를 제공할 수 있다. MavHome은 Episode에 기반하여 상황인식서비스를 제공할 수 있는 아키텍처를 제안하였다. 이런 데이터기반형 방법은 서비스를 일일이 정의할 필요가 없다는 장점이 있다. 그러나, 자주 반복되지 않거나 복잡한 행동패턴을 인식하기 어렵다는 문제점이 있다. 또한, 사용자가 자신이 구상한 서비스를 능동적으로 구현하기 어렵다는 단점이 있다.

둘째, 개발자주도형 상황인식서비스 모델의 대표적인 연구로는 OSGi[4], RCSM[5], SOCAM[6], Aura[7] 등을 들 수 있다. OSGi는 자바언어를 사용한 서비스정의를 지원하며, RCSM은 C, C++, 자바 등 다양한 언어를 통한 서비스정의를 지원한다. SOCAM은 고유의 언어를 제공함으로써 서비스를 정의할 수 있는 방법을 제공하며, Aura는 역시 SQL과 유사한 형태의 고유 언어를 사용하여 상황인식서비스를 정의한다. 이와 같은 개발자주도형 모델은 모두 범용 프로그래밍 언어를 이용하거나 독자적으로 개발한 프로그래밍 언어를 사용하여 서비스를 구현한다. 소프트웨어 개발 전문가들을 통해 다양한 상황인식 서비스를 구현한다는 장점이 있지만 사용자의 개인적 필요나 선호를 반영하기 어렵다는 한계가 있다.

본 논문에서 제안하는 사용자주도형 모델로 분류할 수 있는 연구에는 JIGSAW[8]와a CAPpella[9] 등이 있다. JIGSAW는 서비스 도메인에서 발생될 수 있는 상황을 그림으로 표현한 퍼즐 조각들을 나열함으로써 원하는 상황인식 서비스를 정의한다. 예를 들면, JIGSAW 사용자는 거실전등이 꺼진 모양을 묘사한 그림 조각과 거실창문이 닫히는 모양을 묘사한 그림 퍼즐 조각을 나란히 배치함으로써, “거실전등이 꺼지면, 거실창문이 닫힌다.”라는 서비스를 정의할 수 있다. JIGSAW는 다음과 같은 몇 가지 단점이 있다. 첫째, JIGSAW는 객체의 제한된 일부 기능만 연동하도록 지원하기 때문에 사용자가 원하는 서비스를 정의하지 못하는 경우가 흔히 발생할 수 있다. 둘째, JIGSAW는 아주 단순한 형태의 서비스만 정의할 수 있다. JIGSAW는 관계연산과 산술연산을 지원하지 않기 때문에 “실내온도가 실외온도 보다 5도 이상 낮으면, 냉방병 예방을 위해 에어컨 가동을 중단한다.” 등과 같은 서비스를 정의할 수 없다. 셋째,JIGSAW의 서비스 개발 모델은 간단한 서비스의 경우에는 단순하고 직관적이지만, 확장성에서 한계가 있다. 예를 들면, 10개의 객체가 각각 10개 기능을 가진다고 할 때 이 기능들은 JIGSAW에서 100개의 퍼즐조각으로 표현된다. 장치와 기능이 더욱 다양해지고 있는 우리의 컴퓨팅 환경을 고려할 때 퍼즐조각은 더욱 늘어나게 될 것이다. 두 번째 방법인 a CAPpella는 “Program by demonstration”을 통한 상황인식 서비스 정의를 제안하고 있다[9]. 이 방법에서는 새로운 서비스를 정의할 때, 먼저 사용자가 센서들이 설치된 개발시스템 앞에서 정의하고자 하는 서비스 상황을 반복적으로 시연한다. 시연이 끝나면 시연 기록을 보면서 서비스의 시작조건이 될 상태를 선택하고 대응되는 액션을 지정함으로써 서비스를 정의한다. 그러나, a CAPpella는 시연하기 어려운 상황을 서비스의 정의에 포함하기 어렵다는 문제점이 있다. 예를 들면, ″실내온도가 30도가 되

면"이나 "사용자 A의 혈압이 200이 넘으면" 등과 같은 관념적인 상황은 표현하기 곤란하다.

3. 도 비

우리의 생활공간에는 다양한 센서와 장치를 탑재한 여러 종류의 전자제품, 기기들이 있고 인터넷에는 다양한 정보를 제공하는 많은 수많은 서비스들이 있다. 스마트홈 환경에서 상황인식 서비스의 구현에 이용될 수 있는 우리 주변의 기기나 인터넷 상의 서비스를 callable interface를 가진 객체로 추상화할 수 있는데, 우리는 이를 서비스객체(Service Object) 또는 간단히 객체라 부르기로 한다. 우리 주변에는 다양한 서비스객체들이 있고 그 수와 기능은 정보통신 기술의 발달과 함께 계속 증가하게 될 것이다. 사용자가 주변에 있는 객체들의 서비스를 연동할 수 있다면 필요한 서비스를 사용자가 직접 만드는 것이 가능하다. 이 절에서는 사용자가 여러 객체들의 서비스를 통합하여 개인화된 상황인식 서비스를 구현할 수 있는 방법인 도비 모델에 대하여 설명한다.

사용자가 자신에게 가용한 객체의 기능을 연동하여 구현하는 개인화된 상황인식서비스를 우리는 **시너지 서비스**라고 부르기로 한다. 시너지 서비스는 전문 개발자가 아닌 가정의 일반 사용자를 대상으로 한다는 점과 다수 객체가 제공하는 기능을 신축적으로 연동하여 구현할 수 있다는 점에서 앞에서 살펴본기존의 상황인식서비스 구현 모델과 구분되는 목적을 가지고 있다. 시너지 서비스의 한 예로는, 예를 들어, 집에 TV와 전화기가 있는 경우를 생각해보자. 사용자는 보통 이두 기기에서 제공하는 기능들을 모두 소유하고 있다. 하지만 "전화기가 울리면, TV 소리를 소거한다."와 같이 두 기능이 통합된 서비스는 요소 서비스들이 존재함에도 불구하고 아직 사용자가 직접 구현하기 어렵다. 우리는 머지 않아 여러 장치에서 제공되는 요소 서비스들을 결합함으로써 자신에게 유용한 새로운 시너지서비스를 사용자가 직접 개발할 수 있게 해주는 것이 필요하게 될 것으로 생각한다. 본 논문에서 제안하는 도비 모델은 이런 목적을 가지고 설계되었다.

도비의 시너지서비스 개발 방법은기존의 상황인식서비스와 비교해 다음과 같은 장점들이 있다. 첫째, 사용자가 자신이 원하는 서비스를 스스로 정의할 수 있다. 둘째, 주어진 서비스도메인에서 가용한 객체들을 이용하기 때문에 사용자의 다양한 컴퓨팅 환경을 적극 활용할 수 있다. 셋째, 다양한 상황을 묘사할 수 있기 때문에 기존 사용자주도형 모델로 정의할 수 없었던 복잡한 형태의 상황인식서비스도 정의할 수 있다. 이제 도비 모델에 대해 상세히 설명하기로 한다.

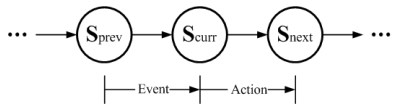
3.1 시너지서비스

시너지서비스는 예측된 상황발생에 대응하여 하나 이상의 액션을 실행하는 서비스이다. 시너지서비스는 다음과 같이 하나의 실행규칙으로 정의된다.

firing_condition => actions

여기서 *firing-condition*은 이 실행규칙의 실행조건으로서 특정 상황이 발생하였음을 의미한다. *actions*는 좌변에 정의된 상황 발생에 대응하여 시스템이 실행할 조치로서, 하나 또는 다수의 액션을 의미한다.

상황을 감시하고 서비스를 실행하는 상황인식 시스템을 우리는 그 시스템을 구성하고 있는 개별 서비스 객체들의 상태를 결합한 상태기계로 모델링 할 수 있다. 우리는 객체의 상태가 한 상태에서 다른 상태로 바뀌는 것을 이벤트라고 부른다. 예를 들어 "TV On"이라는 이벤트는 TV가 "Off"의 상태에서 "On"의 상태로 바뀌었다는 것을 뜻한다. 액션(action)은 시스템의 상태에변화를 가져오는 이벤트이다. 예를 들어 "TV Volume Up"이라는 액션은 TV Volume의 현재 상태를 1 증가시킨 값으로 바꾸라는 뜻이다. 어떤 객체의 상태가 아래 그림과 같이 과거에 Sprev에서, 현재 Scurr를 거쳐, 그 다음에 Snext로 변한다고 하자.



이 때 이벤트와 액션은 각각 다음과 같이 표현할 수 있다.

Event : Sprev → Scurr
Action : Scurr → Snext

시너지 서비스의 실행조건은하나 또는 다수의 이벤트 발생으로 구성된다. 본 논문에서는 사용자들이 실행조건을 다수 이벤트의 혼합으로 구성할 수 있도록 함으로써 기존 사용자주도형의 연구에서 제안된 명세방법보다 더 다양한 상황을 표현할 수 있도록 한다.

3.2 시너지 서비스 개발

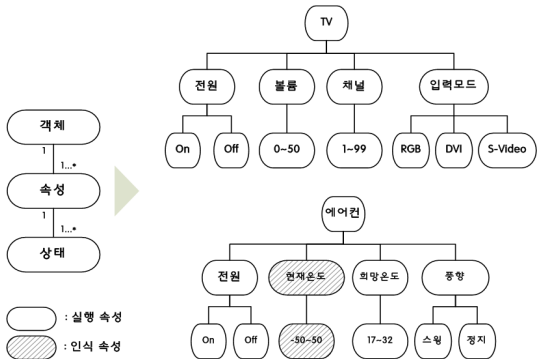
시너지 서비스를 개발하기 위해서는 사용자가 자신의 서비스객체들을 인식할 수 있어야 하고, 서비스객체들을 조합해 상황을 명세할 수 있어야한다. 그리고, 상황인식 시스템은 명세된 상황이 발생할 경우 그에 대응해 지정된 액션을 실행해야 한다.

3.2.1 객체인식

시너지 서비스를 개발하려면 먼저 사용자가 자신이 원하는 서비스의 구현에 필요한 기능들이 서비스도메인에서 제공되는지 확인할 수 있어야 한다. “객체인식”은 이처럼 자신에게 접근 가능한 객체를 확인하는 단계를 의미한다. 우리 주변에는 장치, 제조사, 기능, 서비스의 종류, 사용방법 등이 상이한 다양한 객체가 있다. 이들을 인식할 수 있으려면 장치나 서비스와 같은 자원의 등록, 삭제를 지원하는 객체 관리기술이 필요하다. 또한 원하는 객체를 검색할 수 있는 서비스가 필요하다. 이러한 목적으로 제안된 표준들인 웹서비스[10], OSGi[4], Jini[11], UPnP[12], Bluetooth[13] 등은 동적으로 장치를 관리할 수 있는 기술들로써, 서비스 도메인에 있는 장치나 서비스를 동적으로 검색하는 기능을 핵심으로 하고 있다.

본 논문에서는 시스템 수준에서의 객체 관리와 객체 검색 기능의 구현에 이러한 기술이 사용되는 것으로 가정한다. 하지만 이런 기술들은 전문적 지식을 가진 개발자들을 위하여 개발된 표준으로서 해당 기술의 아키텍처와 프로토콜 및 프로그래밍 등에 대한 깊은 이해를 필요로 하여, 본 연구에서 목표로 하는 일반 사용자를 위한 시너지서비스 구현에 그대로 이용하기는 어렵다. 도비에서는 이러한 서비스발견 기술 계층 위에 위치하면서 사용자가 편리하게서비스 객체의 기능을 인식할 수 있게 해줄 수 있는 방법이 필요하다.

우리는 사용자가 객체의 기능을 이해하기 쉽도록 객체를 그 객체가 취할 수 있는 상태의 집합으로 표현한다. 이를 위해 우리는 SEAD (State-based Event and Action Description) 라는 표기법을 만들었다. SEAD는 객체(Object)를 외부에서 관찰할 수 있는 속성(Attribute)의 집합으로 모델링한다. 객체의 속성은 속성값 즉, 그 속성이 취할 수 있는 상태(State)의 집합으로 표현한다. (그림 1)은 SEAD와 그 사용 예를 보여준다. (그림 1)에서 TV 객체는 ″전원″, ″볼륨″, ″채널″, ″입력모드″라는 속성을 가지고 있으며, 에어컨 객체는 ″전원″, ″현재온도″, ″희망온도″, ″풍향″ 이라는 속성을 가진다. 하나의 속성은 취할 수



(그림 1) SEAD 예

(Figure 1) SEAD Example

있는 값의 범위, 즉, 상태 변화를 의미하는 한 개 이상의 상태를 가진다. TV의 전원 속성은 ″On″과 ″Off″라는 두 상태가 가능하고, 볼륨 속성은 ″0″부터 ″50″까지의 정수로 51가지 구분되는 상태가 가능하다. 이러한 SEAD의 표기 방법은 장치의 기능을 외부의 사용자들이 경험하는 바와 같이 직관적으로 표현함으로써 객체의 외부 기능을 쉽게 이해할 수 있도록 해준다.

속성은 사용자가 그 상태를 제어할 수 있는 실행속성과 장치의 상태를 관찰하기 위한 인식속성으로 구분한다. (그림 1)에서 빗금 친 부분은 인식속성을 의미한다. 예를 들면, 에어컨의 현재온도는 하나의 인식속성으로써, 현재의 온도를 얻을 수는 있지만 이 속성을 통해 원하는 온도를 설정할 수는 없다. 하지만 실행속성인 희망온도를 이용하면 새로운 온도를 설정할 수 있다. 인식속성은 센서를 통해 얻어진 정보를 제공하는 객체의 기능이다.

사용자가 자신이 사용 가능한 서비스 객체와 그 기능을 인식하였다면 그 다음으로 이를 이용하여 firing condition과 그에 대응되는 액션을 명세한다.

3.2.2 이벤트, 액션

이벤트와 액션은 SEAD로 표현된 객체정보를 이용해 명세할 수 있다. 예를 들어, Telephone.Bell=″Ringing″ 이라는 표현은 ″전화벨이 울리면″이라는 이벤트를 의미한다. 또한, ″TV 소리를 소거한다″라는 액션은 TV.Volume := 0 으로 표현할 수 있다. 전화벨이 울리면 TV 소리를 소거한다″라는 실행규칙은 Telephone.Bell=″Ringing″=> TV.Volume:=0 과 같이 명세된다.

도비에서는 프로그래밍 지식이 없는 일반 사용자들을 위하여 이러한 명세를 쉽게 할 수 있도록 이벤트와 액션

을 하나의 구문형식으로 표현한다. 이벤트의 경우는 그 이벤트의 결과로 발생된 객체의 새로운 상태, 즉, 현재 상태가 중요하고, 액션의 경우는 변경하고자 하는 객체의 목표 상태, 즉, 다음 상태가 중요하다. 따라서, 우리는 이벤트와 액션을 모두 객체의 상태로 표현할 수 있다. 상태를 표현하는 SEAD의 object.attribute.value 형식을 이용하면, 앞에서 설명한 실행규칙은 다음과 같이 정의할 수 있다.

Telephone.Bell.Ringing => TV.Volume.0

이러한 통합된 표현식은 사용자가 문자열과 정수라는 타입의 개념을 이해하지 않고도 원하는 실행규칙을 표현할 수 있게 해주는 장점이 있다. 도비 시스템은 이러한 규칙을 정의할 수 있게 해주고 그 의미에 맞는 연산을 수행한다.

3.2.3. 상황 명세

어떤 시너지서비스는 이벤트와 액션의 단순한 연결을 넘어 다수 객체의 상태를 결합한 보다 복잡한 상황 표현 능력을 요구한다. 사용자주도형 상황인식 서비스개발 기술이 확산되려면 사용자가 생각하는 다양한 서비스를 스스로 개발할 수 있게 해주는 것이 매우 중요하다. 상황의 표현에는 일반적으로 프로그래밍과 컴퓨터에 대한 깊은 지식이 필요하다. 기존의 사용자주도형 연구에서는 여러 가지 유용한 상황인식 서비스들을 명세하기 어려웠음을 앞에서 살펴보았다. 사용자의 편의를 위하여 명세시스템의 표현력을 과도하게 축소한 결과인 것이다.

다양한 상황을 표현할 수 있는 표현력은 명세시스템이 지원하는 연산과 직접적인 관련이 있다. 본 절에서는 여러 가지 유용한 상황인식 서비스 시나리오를 분석해봄으로써 우리가 목표로 하는 시너지서비스의 구현을 위해 어떤 연산이 지원되어야 하는지 살펴본다. 논문의 뒷부분에서는 이러한 연산들이 기존 연구에서 어떻게 지원되고 있는지도 살펴본다.

시나리오 S1과 같은 상황인식 서비스를 정의하기 위해서는 비교 연산과 AND 연산이 필요함을 알 수 있다.

S1: 철수는 만성 천식 때문에 공기에 매우 민감하다. 그는 실내 미세먼지농도가 150$\mu g/m^3$ 이상이면 환기를 한다. 이때 실외 미세먼지농도가 100$\mu g/m^3$ 이하이면 창문을 열어 환기하고, 그렇지 않으면 공기청정기를 실행해 환기한다. 그러나 실외 미세먼지농도가 100$\mu g/m^3$ 이하더라도 실내외 온도차가 10도 이상이면 보온을 위해 공기청정기를 실행한다.

실내 미세먼지농도가 150$\mu g/m^3$ 이상"이라는 조건과 "실외 미세먼지농도가 100$\mu g/m^3$ 이하"라는 두 조건은 동시에 만족되어야 하는 논리 AND 관계에 있다.

S2: 에어컨이나 보일러가 켜지면 전동창문을 닫는 그린냉난방서비스

S2에서는 "에어컨 켜짐"과 "보일러 켜짐" 이라는 두 상태 중 하나라도 만족되면 이에 대응되는 기능이 실행된다. 이 서비스의 경우에는,

에어컨 켜짐 => 전동창문 닫음
보일러 켜짐 => 전동창문 닫음

이라는 두 개의 실행규칙으로 정의할 수 있다. 그러나, OR 연산을 이용하면 이 두 규칙을 하나의 규칙으로 간단히 정의할 수 있다. OR 연산이 반드시 필요한 것은 아니지만 지원된다면 상황을 보다 간단히 표현할 수 있다. 또한 AND와 OR가 결합한 다양한 조건을 표현할 수 있어서 표현력도 높일 수 있다.

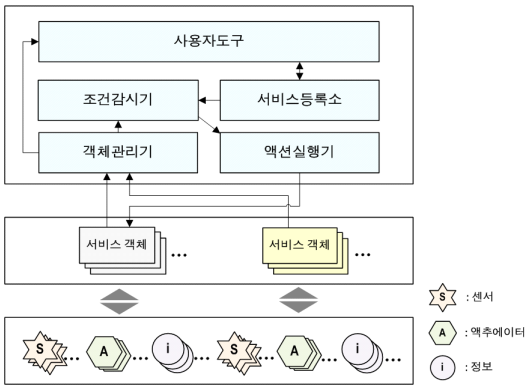
앞에서 살펴본 바와 같이 시나리오 S1을 정의하기 위해서는 논리연산과 더불어 비교를 위한 관계연산이 필요하다. 관계연산을 통해 "실내 공기오염도 150$\mu g/m^3$ 이상", "실외 공기오염도 100$\mu g/m^3$ 이하" 등의 조건을 표현할 수 있다. 다음 서비스시나리오 S3에서도 관계연산의 필요성을 볼 수 있다.

S3: 사용자의 혈압이 200을 초과하면 구조메시지를 긴급구조센터로 전송하는 스마트헬스케어서비스

"혈압이 200을 초과"라는 조건 역시 관계연산을 이용해서 표현할 수 있다.이와 같이 다양한 서비스정의를 위해서 관계연산은 논리연산과 마찬가지로 중요함을 알 수 있다.

시나리오 S1에서 "실내온도의 차가 10도 이상"이라는 구절을 살펴보자. 실내와 실외의 온도차를 계산하기 위해서는 산술연산이 필요하다. 산술연산의 다른 예로 아래의 시나리오 S4를 살펴보자.

S4: 에어컨이 켜져 있으면 희망온도설정을 실외온도보다 6도 낮게 하는 에너지절약 및 냉방병예방 서비스



(그림 2) 도비 아키텍처
(Figure 2) Dobby Architecture

S4에서 “희망온도를 실외온도보다 6도 낮게”라는 상태는 산술연산과 비교연산으로 표현할 수 있다. 이처럼 상황인식 서비스의 정의에는 산술연산이 필요한 경우가 자주 발생한다.

지면의 제약으로, 일부의 상황인식 시나리오를 살펴보았지만, 지금까지 살펴본 바에 의하면 산술연산 +, -, 비교연산 >, =, <, 논리연산 AND, OR 등은 시너지 서비스를 정의하기 위해 필요함을 알 수 있었다. 표현능력을 높이기 위해서는 범용 프로그래밍 언어에서 제공하는 연산들을 모두 제공한다면 보다 많은 상황을 명세할 수 있을 것이다. 그러나, 앞에서 설명한 바와 같이 프로그래밍 언어 전문적 지식을 요구하기 때문에 프로그래밍 지식이 없는 사용자들을 위한 방법으로는 부적합하다.

도비에서는, 앞에서 분석한 산술연산, 관계연산, 논리연산을 지원함으로써 표현력을 높이되 과도한 프로그래밍 경험을 요구하지 않고도 사용자들이 상황을 표현할 수 있는 방법을 연구하고자 노력하였다. 참고로, 아래의 표2에 보인 것처럼, 기존의 사용자주도형 상황인식서비스 정의 방법에서는 이러한 연산들이 모두 지원되지 않거나 일부만 지원되고 있다.

3.3 도비 아키텍처

(그림 2)는 도비시스템의 아키텍처이다. 시너지서비스의 구현에 참여하는 모든 서비스 객체들은 객체관리기에 등록된다. 등록을 통해 객체들이 SEAD로 명세된 자신의 기능을 도비시스템에 알려주게 된다. 등록된 객체는 특정 기능이 실행되거나 제공할 새로운 정보가 있으면 이를 도비시스템에게 보고한다. 도비시스템은 이런 이벤트들을 수집하고 분석하여 실행조건이 만족되는 시너지서비스가 있는지 검사한 후만약 있다면 해당 규칙에 명세된 액션을 실행함으로써 서비스를 제공한다. (그림 2)는 이런 동작이 일어나는 과정을 보여준다. 아키텍처를 구성하는 각 요소를 좀 더 상세히 설명하기로 한다.

객체관리기는 객체들의 라이프사이클을 관리하고 객체들이 보내오는 이벤트를 수집하는 역할을 한다. 사용자도구는 사용자가 객체의 기능을 확인하고 이들을 조합하여 서비스를 정의할 수 있게 해주는 프로그래밍 환경과 사용자 인터페이스를 제공한다. 정의된 서비스는 서비스등록소에 저장된다.

조건감시기는 실행조건이 만족된 서비스 규칙을 찾아내어 액션을 실행하는 작업을 반복한다. 먼저 객체관리기로부터 실시간으로 객체들의 상태를 전달받고 이를 토대로 서비스등록소에 저장된 서비스 중 실행조건을 만족하는 서비스 규칙을 선별한다. 만약 실행조건이 만족된 서비스가 있다면 액션실행기에게 이에 대응되는 액션을 실행하라고 지시하고 액션실행기는 해당되는 서비스 객체에 이를 전달한다. 해당 객체는 도비의 지시에 따라 자신의 기능을 실행함으로써 사용자에게 서비스를 제공한다.

3.4 Program by selection

도비는 전문지식이 없는 사람도 시너지 서비스를 손쉽게 명세할 수 있게 해주는 것을 목표로 한다. 일반적으로 컴퓨팅 서비스를 개발하기 위해 가장 널리 쓰이는 방법은 프로그래밍이다. 프로그래밍은 개발하고자 하는 기능을 문법에 맞게 프로그램 텍스트로 서술하는 방법이다. 프로그래밍은 변수, 타입, 연산자, 자료구조, 문법 등을 비롯하여 일반 사용자가 쉽게 이해할 수 있는 범위를 넘는 전문 지식을 필요로 하기 때문에 우리가 목표로 하는 시너지서비스 명세에 적합하지 않다.

본 논문에서는 사용자가 서비스를 보다 쉽게 정의하는 방법으로 ***Program by selection*** (후보선택에 의한 명세)을제안한다. 이 방법은 서비스 정의 과정을 여러 단계로 나누어 사용자가 매 단계마다 다음 단계에 올 수 있는 입력의 후보군들 중에서 하나를 선택해 가는 방법으로 서비스를 정의하는 문법지시적(Syntax-directed) 실행규칙 명세 방법이다. 이 방식은 데이터 타입과 타입 간의 호환, 데이터 타입별 가능 연산 타입체킹 등의 전문 지식이 없어도 시스템의 안내에 따라 원하는 서비스를 정의할 수 있도록 해준다. 또한, 구두점과 같은 문장부호를 빠뜨리거나 오타를 치는 것과 같이 사소하지만 자주 발생하

는 실수를 예방해주는 장점도 있다.

실행규칙 정의를 위한 언어는 3.2절에서 설명한 바와 같이 SEAD의 상태 표현식과 산술연산, 관계연산, 논리연산을 지원하며 다음의 EBNF 문법으로 정의된다.

```
rule ::= fc -> actions
fc ::= cond | fc (AND | OR) fc | (fc)
cond::= object.attribute.value
        | object.attribute [(+ | -) Number]
(= | > | < | >=) Number
```

이 문법에서 짙은 색 기호는 메타심볼이다. rule은 실행규칙을 의미하고 fc는 firing condition 이다. cond는 SEAD에서 설명한 객체의 상태이거나 상태에 대한 명제로 bool 값을 갖는다. object는 도비에 등록된 객체의 이름이다. 타입 규칙은 도비의 사용자도구에 구현된 인터프리터를 통하여 자동으로 검사되어 선택 가능한 후보들이 사용자의 화면에 제시되도록 구현된다.

(그림 3)은 후보선택에 의한 명세방법을 요약한 것이다. 예를 들어, 사용자가 다음 서비스시나리오 S5를 정의한다고 하자.

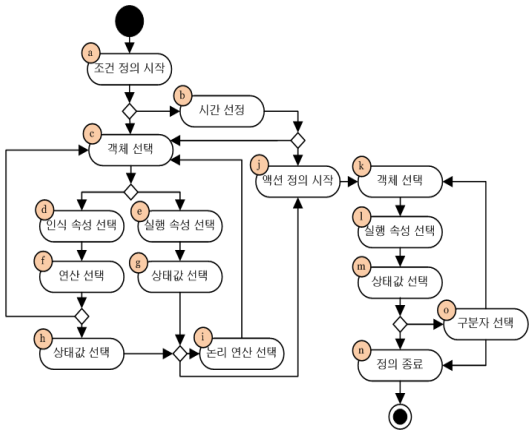
> S5: DVD플레이어가 재생되면 TV를 켜고, 거실전등을 끄며, 커튼을 닫는 홈시어터서비스

S5는 SEAD를 사용하여 D1과 같이 명세할 수 있다.

> D1: DVD_Player.DVD.Play => TV.Power.On, Living_Room.Light.Off, Living_Room.Curtain.Close;

이런 서비스를 정의하기 위해 사용자는 도비시스템의 안내에 따라 (그림 3)의 각 단계를 (표 1)과 같이 순회한다.

먼저 이벤트 정의를 시작한다(a). 이벤트정의를 시작하면 도비는 서비스도메인에서 현재 가용한 모든 객체의 목록을 보여준다. 사용자는 이 중 객체 "DVD Player"를 선택한다(c). 사용자가 객체 "DVD Player"를 선택하면 도비는 다음 선택 가능한 후보들로 "DVD Player"의 모든 속성들을 보여주고 사용자는 이들 중 실행속성 "DVD"를 선택한다(e). 그 다음으로 도비는 객체 "DVD Player"의 "DVD" 속성에 속한 모든 상태들을 표시하고 사용자는 이 중 "Play"를 선택한다(g). 그러면 도비는 추가 조건을



(그림 3) 후보선택을 통한 시너지서비스 정의 방법
(Figure 3) Programming by Selection Example

(표 1) 후보선택방식의 서비스정의 예
(Table 1) Service Definition Example

순서	단계	선택	설명
1	a	이벤트정의 시작	서비스 정의 시작
2	c	객체 선택	가용 객체 중"DVD Player" 선택
3	e	실행속성 선택	"DVD Player"를 속성 중 "DVD"선택
4	g	상태 선택	"DVD Player"의 "DVD" 속성의 상태 중 "Play"를 선택
5	j	액션 정의 시작	컨디션 정의 완료 후 액션 정의 시작
6	k	객체 선택	가용 객체 중 "TV" 선택
7	l	실행속성 선택	"TV" 속성 중 "Power"선택
8	m	상태 선택	"TV"의 "Power" 속성의 상태 중 "On"을 선택
9	o	구분자 선택	다른 액션 정의를 위해 ","선택
10	k	객체 선택	가용 객체 중"Living Room" 선택
11	l	실행속성 선택	"Living Room" 속성 중 "Light"선택
12	m	상태 선택	"Living Room"의 "Light" 속성의 상태 중 "Off"을 선택
13	o	구분자 선택	다른 액션 정의를 위해 ","선택
14	k	객체 선택	가용 객체 중"Living Room" 선택
15	l	실행속성 선택	"Living Room" 속성 중 "Curtain"선택
16	m	상태 선택	"Living Room"의 "Curtain" 속성의 상태 중"Close"를 선택
17	n	정의종료	서비스 정의 완료

정의할 수 있는 연산자들('&&'와 '||')과 함께 연산자(Action Definition Operator '=>')를 출력한다. 조건 정의를 마친 경우, 사용자는 '=>'를 선택함으로써 액션 정의를 시작한다(j). 논리연산으로 결합된 조건을 추가로 정의하려면 '&&' 또는 '||' 연산자를 선택하면 된다. 우리는 조건정의가 완료되었다고 하고 그 다음으로 수행할 내용에 대해 설명하기로 한다.

액션정의를 시작하면 도비는 다시 가용객체의 목록을

보여준다. 사용자는 이 후보객체 중 "TV"를 선택하고(k), TV의 속성 "Power"와 그 상태 "On"을 차례로 선택함으로써 하나의 액션을 정의한다(l, m).

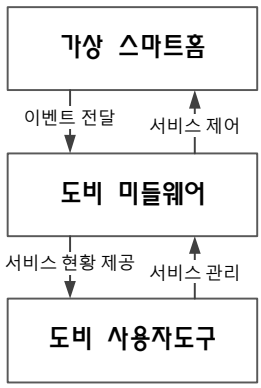
이 후 도비는 더 정의할 액션이 있는지 서비스정의를 종료할지 결정하기 위해 구분자(',')와 함께 서비스종료심볼("End"')을 출력한다. 추가 액션을 더 정의하기 위해 사용자는 구분자를 선택한다(o). 이 후 사용자는 단계 k, l, m, o를 반복하면서 "Living_Room.Light.Off"와 "Living_Room.Curtain.Close" 정의한 후 마지막으로 사용자는 서비스정의종료심볼을 선택함으로써 서비스 정의를 마친다(n).

이러한 후보선택방법을 통해 사용자는 보다 쉽게 서비스를 정의할 수 있다. 도비는 매 단계마다 문법, 등록된 객체 및 사용자의 선택 내용에 따라 그에 적합한 후보들을 사용자에게 제시한다. 예를 들어 "실내온도가 20 보다 작다"라는 것을 표현하기 위해 사용자가 인식속성 "현재온도"를 선택하는 경우 도비는 그 속성의 타입이 정수이기 때문에 그 후 관계연산이나 산술연산이 가능하다는 것을 판단하고 해당 연산자를 선택 가능한 후보로 출력한다. 같은 인식속성이지만 "사용자의 위치"와 같은 경우에는 타입이 문자열이므로 산술연산과 같이 문자열에 수행할 수 없는 연산자는 출력하지 않는다. 이러한 방법으로 도비는 언어에 대한 전문지식 없이 사용자를 안내함으로써 원하는 시너지서비스를 명세할 수 있도록 해준다.

4. 구 현

도비 시스템은 크게 두 부분으로 구성된다. 하나는 객체들을 등록하고 이벤트를 수집하며 서비스를 실행하는 역할을 하는 미들웨어이고, 다른 하나는 사용자가 시너지서비스를 명세할 수 있도록 지원하는 사용자도구이다. 도비를 실제로 구현하고 실험하기 위해서는 다양한 객체들이 존재하는 서비스도메인을 구축해야 할 것이다. 그러나, 그러한 환경의 구축에는 큰 비용이 요구되어 아직 이러한 시험 환경을 구축하지 못하고 있다.

본 연구에서는 위에서 설명한 논문의 아이디어를 구현하고 실험하기 위한 환경으로 먼저 가상 스마트홈 환경을 테스트베드로 구현한 다음, 이 환경에서 도비를 실험하였다. 가상 스마트홈은 센서와 액추에이터를 갖춘 실제 환경에서의 실험과 비교하여 객체와의 인터페이스, 액션의 실행에 의한 시스템의 상태 변화 등에서 사실적인 관찰을 할 수 없다는 한계가 있다. 그러나, 비행 시뮬레이터에서의 훈련이 실제 비행기 조종하는 것과 같이 인정받을 수 있는 것처럼, 실험의 목적에 따라서는 가상



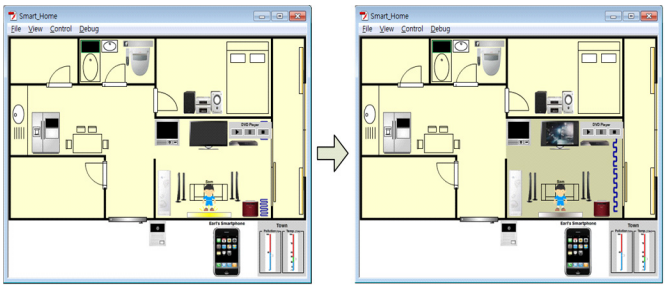
(그림 4) 도비 실험환경 구성
(Figure 4) Dobby Experimental Environment

스마트홈도 실험을 위한 충분히 유용한 실험 모델이 될 수 있다. 특히, 본 논문은 사용자에 의한 시너지서비스 정의와 그 실행 방법의 유효성을 확인하는 것을 실험의 목표로 하기 때문에 가상 환경에서의 실험이 라고 해도 그 실험을 통해 목적하는 효과를 확인할 수 있을 것이다. (그림 4)는 실험환경구성을 보여준다.

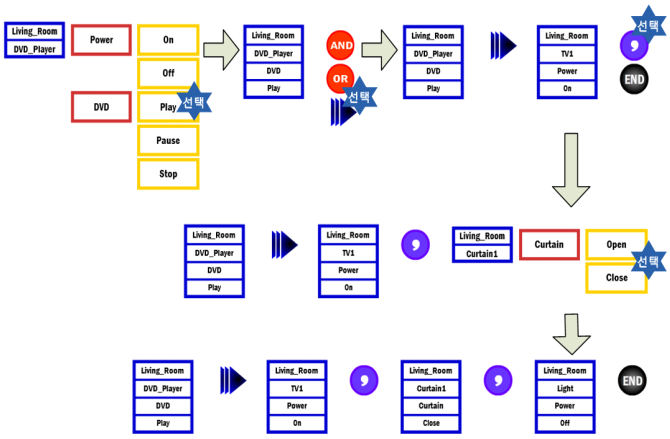
가상 스마트홈은 가정의 IT 환경을 구현한 프로그램이다. UI를 가진 이 프로그램은 실제 가전기기들을 제어할 수 있고, 이에 따라 발생하는 이벤트를 도비미들웨어에 제공한다. 도비미들웨어는 이벤트를 확인하고 관계된 상황인식서비스를 실행하거나 중지한다. 사용자는 도비미들웨어와 연결된 사용자도구를 통해 정의된 서비스를 확인하고 필요에 따라 기존 서비스를 수정, 삭제하거나 새로운 서비스를 정의한다. 각각의 도비 실험환경에 대해 보다 상세히 설명하면 다음과 같다.

4.1 가상 스마트홈 테스트베드

(그림 4)는 도비의 테스트베드인 가상 스마트홈의 모습이다. 가상 스마트홈은 우리가 흔히 볼 수 있는 전자제품들을 가진 스마트홈 환경으로서, 장치(appliance)들을 대표하는 서비스 객체들과 이 객체들의 상태 변화를 시각적으로 관찰할 수 있게 해주는 Flash 인터페이스로 구현되었다. 이 가상의 스마트홈에는 TV, DVD플레이어, 전동커튼, 오디오, 냉장고, 공기청정기 등 여러 객체들이 설치되어 있으며 이 객체들은 모두 기본적인 기능을 조작할 수 있도록 설계 구현되었고 도비 미들웨어에 등록되어 있다. 또한 한 장치에서 이벤트가 발생하면 이를 즉시 도비 미들웨어에게 전달하고, 도비미들웨어의 지시에



(그림 5) 가상 지능형홈
(Figure 5) Virtual Smart Home



(그림 6) 사용자 툴을 통한 서비스정의 예
(Figure 6) Service Definition Tool Example

따라 객체의 기능을 실행할 수 있도록 구현되었다.

4.2 도비미들웨어

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

4.3 도비 사용자도구

도비의 사용자도구는 사용자가 도비미들웨어에 등록된 객체와 그 기능을 확인하고 이를 조합하여 시너지서비스를 명세할 수 있도록 지원한다. 사용자도구는 SEAD를 구현한 것이다. (그림 6)은 도비 사용자도구를 사용해 3.4절에서 설명한 시너지서비스 D1을 정의하는 과정을 보여준다. 도비 사용자도구는 그림에서 볼 수 있듯이 사용자가 후보 중 하나를 선택하는 과정을 반복함으로써 최종적으로 서비스 D1을 정의할 수 있게 해준다.

이렇게 정의된 이렇게 정의된 서비스는 (그림 5)의 왼쪽과 같이 사용자가 DVD플레이어를 재생하면 (그림 5)의 오른쪽처럼 자동으로 TV를 켜고, 거실전등을 끄며, 거실커튼을 닫는다. (그림 6)에서 볼 수 있듯이 거실에 있는 객체인DVD플레이어는 “Power”와 “DVD”라는 속성을 가지고, 속성 “Power”는 “On”과 “Off”를 상태로 갖고 속성 “DVD”는 “Play”, “Pause”, “Stop”을 상태로 갖는다. 사용자는 이런 것들을 적절히 조합하고 연동함으로써 원하는 상황인식서비스를 정의할 수 있다.

(표 2) 상황인식서비스 개발 방법의 비교 (서비스 예 미지원 표기)

(Table 2) Context-aware Service Definition Method Comparison

분류	연구	표현력				사용의 복잡도
		산술 연산	관계 연산	논리 연산	시간 명세	
개발자주도형	RCSM	○	○	○	○	상
	AURA	○	○	○	○	상
	SOCAM	×	×	○	○	상
	OSGi	○	○	○	○	상
	ECA-DL	×	○	○	○	상
데이터기반형	MavHome	×	×	○	×	하
	Casper	×	×	○	○	하
사용자주도형	Jigsaw	×	×	○	×	중
	a CAPpella	×	×	○	×	중
	Dobby	○	○	○	○	중

5. 평 가

(표 2)는 상황인식에 관한 기존연구와 도비를 비교 분석한 표이다. 개발자주도형 방식인 RCSM, AURA, OSGi는 다양한 연산을 지원해 표현력은 매우 높지만 전문지식을 갖춘 사람만 서비스를 정의할 수 있고 그 절차도 매우 복잡하다. SOCAM은 이들에 비해 서비스정의의 복잡도는 낮추었으나 S1, S3, S4와 같이 산술연산이나 관계연산이 필요한 서비스를 지원하지 않아 표현력이 낮다. ECA-DL은 산술연산을 제외한 대부분의 연산을 지원해 표현력이 높은 편이나 전문지식을 필요로 하는 복잡한 방법이다. 데이터기반형 방식인 MavHome과 Casper는 시스템이 사용자의 행동을 분석해 상황인식서비스를 자동으로 만들어주지만 단순한 형태의 서비스만 가능하다는 한계가 있다. 사용자주도형 방식인 JIGSAW[3], a CAPpella[5]는 전문지식이 없는 사용자도 서비스를 정의할 수 있게 사용의 복잡도가 높지 않으나 전반적으로 표현력이 너무 낮아 서비스 S1, S2, S3, S4, S5를 정의할 수 없다.

이에 비해 도비는 전문지식이 없는 사용자가 사용할 수 있으며 산술, 관계, 논리연산은 물론이고 본 논문에서는 자세히 설명하지 않았지만 시간제약을 명세할 수 있어 본 논문에서 제안한 상황인식서비스 예를 모두 지원한다. 또한 Program by selection을 도입함으로써 사용의 복잡도가 JIGSAW나 a CAPella에 비해 상대적으로 낮다는 장점이 있다. 이 중 도비가 기존 사용자주도형 방식인 JIGSAW에 비해 왜 복잡도가 낮은지 상세히 분석해본다. 관련 연구에서 언급했듯이 JIGSAW는 기기의 상태나 동작을 하나의 조각으로 표현한다. 어떤 서비스도메인에 10개의 객체가 있고 각 객체는 5개의 속성을 가지며 각 속성은 다시 10개의 상태로 구성되어있다고 하자. 이런 환경에서 사용자가 <A, B>라는 매우 단순한 형태의 시너지서비스를 명세한다고 할 때 JIGSAW는 총 500(10*5*10)개의 조각으로 객체의 기능이 표현된다. x개의 후보 중 y개를 선택하는데 소요되는 노력을 x/y라고 정의하면, 500개의 후보 중 하나를 선택하는데 소모되는 노력을 500(500/1)이다. JIGSAW는 서비스 <A, B>를 명세하기 위해 이벤트 A를 선택하는데 500의 노력이 소요되고, 액션 B를 선택하는데 다시 500의 노력이 소요되어 총 1000이라는 노력이 소요된다.

이번엔 도비의 경우를 살펴보자. 도비는 객체, 속성, 조건을 단계적으로 탐색하므로 이벤트 A를 정의하는데 약 25(10/1 + 5/1 + 10/1)라는 노력이 소요된다. 액션 B를 명세 하는데 역시 같은 노력이 필요하므로 총 50의 노력이 필요하다. 즉, 도비는 JIGSAW보다 20배 더 쉽게 서비스 <A, B>를 명세할 수 있다고 할 수 있다. 도비와 JIGSAW의 차이는 단순히 선택의 복잡성 문제가 아니다. 도비는 산술, 관계연산을 지원하고 시간에 따른 상황인식서비스를 정의를 지원하는 기능을 더 갖추고 있는 등 그 표현력이 더 높다고 할 수 있다.

6. 결론 및 향후과제

본 논문에서는 상황인식서비스 개발방법을 세가지로 분류하고 사용자주도형 상황인식서비스 정의방법에 속하는 시너지서비스의 개발 방법에 대하여 설명하였다. 시너지서비스의 개념을 소개하였고 이를 구현하기 위한 기반기술로서 도비와 Program by Selection 방법에 대하여 상세히 설명하였다. 어떤 서비스를 필요로 하는지는 사용자마다 서로 다르기 때문에 사용자의 필요에 주목하는 사용자 중심의 서비스가 필요하다. 상황인식에 관한 연구의대부분을 차지하는 개발자주도형 개발모델로는 이 문제를 해결하기 어렵다는 점과 기존의 사용자주도형 상황인식 서비스 정의 방법은 사용하기가 어렵고 표현력도 낮다는 점을 설명하였다. 도비는 가용한 객체를 조합하여 사용자가 직접 필요한 시너지서비스를 만들어 쓸 수 있게 함으로써, 첫째 사용자 개개인의 다양한 요구를 만족시킬 수 있고, 둘째 각자의 컴퓨팅 환경에 맞는 서비스를 개발할 수 있으며, 셋째 가용한 객체의 기능을 자유롭게 연동할 수 있어 사용자가 다양한 서비스를 만들 수 있도록 지원한다는 장점이 있다.

향후 연구로는 객체 간 멀티미디어 연동이 가능하도록 도비를 발전시킬 계획이다. 최근 많은 멀티미디어 기기들이 등장하고 있다. 이런 멀티미디어 기기들간의 상호연동을 이용하면 여러 유용한 서비스들을 개발할 수 있다. 예를 들어, 거실에서 TV를 보다가 물을 마시러 부엌으로 잠시 이동하더라도, 부엌 냉장고나 싱크대에 설치된 화면을 통해 시청 중이던 방송을 계속 볼 수 있도록 하는 멀티미디어 시너지서비스를 사용자가 개발할 수 있도록 지금의 도비를 확장할 계획이다.

참 고 문 헌(Reference)

[1] Jeongkyu Park and Keung Hae Lee, "CASPER: Mining Personalized Services," Ubiquitous Intelligence and Computing, LNCS 6406, pp. 337-351, Oct. 2010.

[2] Heikki Mannila, Hannu Toivonen and A. Inkeri Verkamo, "Discovery of Frequent Episodes in Event Sequences," Data Mining and Knowledge Discovery, Vol. 1, Num. 3, pp. 259-289, 1997.

[3] Sajal K. Das, Diane J. Cook, Amiya Bhattacharya, Edwin O. Heierman III,and Tze-Yun Lin, "The role of prediction algorithms in the MavHome smart home architecture,": Wireless Communications, Vol. 9, Issue 6, pp. 77-84, Dec. 2002.

[4] OSGi Alliance, www.osgi.org

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Web Services, http://www.webservices.org

[11] 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.

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

[13] Bluetooth, http://www.bluetooth.com/English/Pages/default.aspx

◐ 저 자 소 개 ◑

박 정 규 (Jeongkyu Park)

2004년 한국항공대학교 컴퓨터공학과 학사
2006년 한국항공대학교 컴퓨터공학과 석사
2011년 한국항공대학교 컴퓨터공학과 박사
2012년~현재 LIG시스템㈜
관심분야 : 상황인식서비스, 모바일컴퓨팅, 임베디드시스템, 유비쿼터스센서네트워크

이 긍 해 (Keung Hae Lee)

1980년 서강대학교 전자공학과 학사
1986년 Virginia Tech 전산학과 석사
1990년 Virginia Tech 전산학과 박사
1990년~1993년 IBM
1999년~2000년 IBM Almaden Research Center 방문과학자
1993년~현재 한국항공대학교 컴퓨터공학과 교수.
관심분야 : 소프트웨어공학, 프로그래밍언어, 상황인식서비스