유비쿼터스 환경에서의 UPnP 장치 사용자 인터페이스의 자동 생성 및 재구성 기법

김병오, 김성일, 이경민, 이동만
한국정보통신대학교
{cmossetup, adreamer, kmlee, dlee}@icu.ac.kr

An Automatic Generation and Reconfiguration Scheme for UPnP Device User Interface in a Ubiquitous Computing Environment

Byoungoh Kim, Sungil Kim, Kyungmin Lee, Dongman Lee
Information and Communication University

요 약

유비쿼터스 환경에서 사용자는 언제 어디서나 주변에 편재하는 컴퓨팅 장치나 서비스를 활용하여 자신이 필요로 하는 작업을 수행할 수 있다. 이러한 컴퓨팅 장치와 서비스를 제어하기 위한 표준으로서 UPnP(Universal Plug and Play)가 각광을 받고 있다. 하지만, 기존에 개발된 UPnP 장치의 원격 제어 프로그램은 사용자의 상황 정보, 선호도 및 접근 권한에 관계없이 동일한 사용자 인터페이스를 제공하기 때문에 사용자에게 편의를 제공하는데 있어 한계가 있다. 본 논문에서는 이러한 단점을 극복하기 위하여 개발된 개인화 및 상황 인지를 지원하는 UPnP 장치 원격 제어 프로그램의 설계 및 구현에 대해 기술한다. 제안된 원격 제어 프로그램은 사용자의 상황, 선호도 및 접근 권한에 따라 사용자 인터페이스(User Interface, UI)가 동적으로 생성되고 재구성된다. 또한 이러한 UI 개발의 편의성을 위하여 UPnP 장치 기술(description)에 따라 UI 를 자동으로 생성하는 방법에 대해서도 기술한다.

Keyword : Automatic User Interface Generation & Reconfiguration, Personalization, Context-Awareness, UPnP, Remote Control

1. 서 론

유비쿼터스 컴퓨팅 환경에는 다양한 컴퓨팅 장치(device) 및 서비스(service)가 편재하며 네트워크를 통하여 서로 연결되어 있다. 사용자는 언제 어디서나 주변에 존재하는 컴퓨팅 장치나 서비스를 활용하여 자신이 필요로 하는 작업을 수행할 수 있다. 이를 위해서는 이러한 장치 및 서비스를 편리하게 제어(control)할 수 있는 방법이 필요하다. UPnP (Universal Plug and Play) [1]는 정보가전, PC 관련 장비 등 여러 장소에 분산되어 있는 장치 및 서비스간의 쉽고 편리한 통신 방법을 제공해주는 표준으로서 홈네트워크 등 유비쿼터스 환경에서의 제어 미들웨어로 각광을 받고 있다.

UPnP 장치를 제어할 수 있는 원격 제어(remote control) 프로그램으로써 Intel UPnP SDK의 Device Spy [2]와 CyberLink 의 Control Point [3] 등이 있다. 하지만, 이들은 사용자의 상황 정보, 선호도 및 접근 권한에 관계없이 동일한 사용자 인터페이스(user interface, UI)를 제공하기 때문에 사용자에게 편의를 제공하는데 있어 한계가 있다. 이러한 단점을 극복하기 위해서는 개개인의 현재 상황, 선호도 및 장치에 대한 접근 권한에 따라 UPnP 기기의 제어를 위한 UI 가 동적으로 생성 및 재구성되어야 한다. 예를 들어, 사용자의 현재 위치에 따라 사용할 가능성이 큰 장치 또는 사용자가 자주 사용하던 장치를 원격 제어 프로그램의 장치 목록 앞 부분에 배치하거나 사용자의 접근 권한에 따라 장치 목록이 다르게 구성되어야 한다.

본 논문에서는 사용자의 상황, 선호도 및 접근 권한에 따라 UI 가 동적으로 생성되고 재구성되는 UPnP 장치 원격 제어 프로그램의 설계 및 구현에 대해 기술한다. 본 논문에서 제안한 UPnP 원격 제어 프로그램은 다음과 같은 특징을 가진다. 첫째, 가장 중요한 상황 정보 중의 하나인 사용자 위치에 따라 UI 가 동적으로 재구성된다. 즉, 사용자에게 가까이 있는 장치일수록 제어 장치 목록의 앞부분에 배치된다. 둘째, 사용자의 제어 프로그램 사용 이력(usage history)를 데이터베이스화하여 각 장치에서 가장 자주 사용하는 기능을 장치의 제어 기능 목록의 앞부분에 배치한다. 셋째, 역할 기반 접근 제어(roll-based access control) 방식을 사용하여 사용자의 접근 권한에 따라 UI 가 구성되도록 하였다. 또한, 이러한 UI 개발의 편의성을 위하여 UPnP 장치 기술(description)에 따라 UI 를 자동으로 생성하는 방법을 제공하고 있다.

본 논문은 다음과 같이 구성된다. 2 절에서는 UPnP 에 대한 간략한 설명과 함께 기존에 개발된 UPnP 장치 제어 프로그램에 대해 분석한다. 3 절과 4 절에서는 각각 우리가 제안하는 개인화 및 상황 인지 UPnP 원격 제어 프로그램의 설계와 구현에 대해 서술한다. 마지막으로 5 절에서 결론과 함께 향후 연구 방향에 대해 논한다.

2. 관 련 연 구

UPnP 는 Universal Plug & Play 의 약자로 기존의 PC 상에서 제공되던 Peripheral Plug & Play 의 확장된 개념이다. 기존의 PnP 는 장치 드라이버를 사용하여 PC 와 직접적으로 연결된 프린터나 모니터와 같은 기기들을 검색하고 제어할 수 있도록 도와준다. UPnP 는 이와 같은 PnP 기능을 네트워크에 연결된 다양한 기기들에 대하여 지원해준다. 하지만 기존의 PnP 에서 장치 드라이버를 사용하는 반면에, UPnP 에서는 네트워크 상에 통일된 프로토콜을 적용시킴으로써 미디어에 독립적인 특성을 갖게 되었다 [1].

이러한 UPnP 장치의 제어를 위해 구현된 프로그램으로써 다음의 두 가지가 있다.

2-1 Intel UPnP SDK


(a) Device Spy **(b) Network Light**

그림 1. (a) Intel UPnP SDK의 Device Spy와 (b) 가상 조명 장치

그림 1 은 Intel UPnP SDK 에서 제공하는 통합 원격 제어 서비스 역할을 하는 Device Spy 와 그에 의해 제어되는 가상 UPnP 조명 장치이다 [2]. UPnP 장치의 기능을 제어하는 데는 문제가 없는 애플리케이션이지만, 사용자가 애플리케이션을 사용하는 데 있어서 불편함을 느낄 수 있다. 제어를 위한 인터페이스가 같은 컴포넌트 그룹들을 나열한 것에 불과하기 때문에, 그에 따른 혼동이 가능하기 때문이다. 또한, 사용자의 현재 상황, 선호도 및 접근 권한에 대한 고려가 전혀 되지 않아 있기 때문에, 사용자에 최적화된 UI 를 제공해주지 못한다.

2-2 CyberLink


(a) Control Point **(b) 가상 UPnP 시계**

그림 2. (a) CyberLink Control Point와 (b) 가상 UPnP 시계

CyberLink 는 Java 언어를 기반으로 제작된 UPnP 개발 라이브러리의 하나로, 자체적으로 UPnP 장치들을 통제하기 위한 API 를 제공하고 있다 [3]. 이 API 에 따라 UPnP 장치를 통제하기 위한 Control Point 와 다양한 가상 UPnP 장치들을 제공하고 있다. 기존의 UPnP 의 XML 서비스 정의나 장치 정의의 규칙에 충실하게 구현되어 있기 때문에, Intel UPnP SDK 에서와 같은 문제점이 여전히 존재하고 있다.

3. 설 계

3-1 전 체 구 조

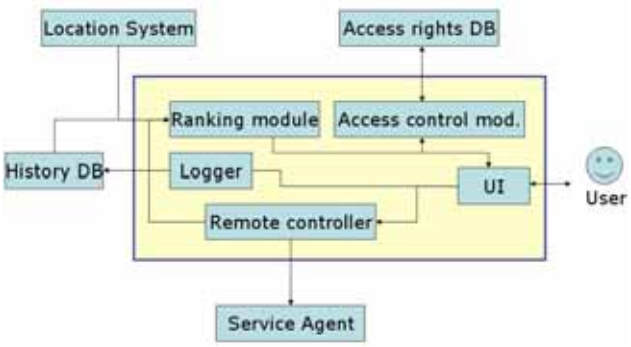


그림 3. 원격 제어 프로그램 구조도

그림 3 은 원격 제어 프로그램의 구조도로, 상황 정보를 이용하여 사용자의 권한을 제어하기 위한 랭킹 모듈, 접근 제어 모듈, 위치 정보 시스템, 그리고 사용 기록 데이터베이스 사이의 관계를 나타낸다. 먼저 각 장치에 대한 장치 목록과 명령 목록이 Service Agent 로부터 Ranking Module 로 보내진다. Ranking Module 은 사용자의 위치와 사용자의 사용 기록을 받아와 목록을 정렬한다. (미들웨어에서 사용자의 위치를 파악해 정렬된 상태로 목록으로 보내주는 걸로 변경됨) 사용권한에 따라서 가능한 명령만이 UI 에 나타나게 된다. 그럼 사용자는 원하는 명령을 실행하게 되고 이 명령은 Remote Controller 로 보내져 Service Agent 로 명령을 실행하게 된다.

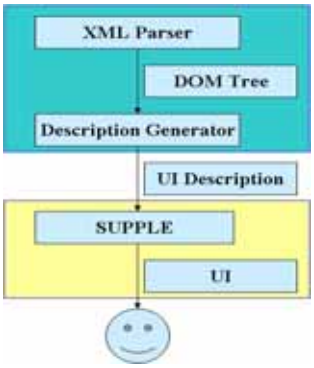


그림 4. 사용자 인터페이스 자동 생성 과정

그림 4 는 Service Agent 와 Remote Controller 사이의 통신에서 일어나는 사용자 인터페이스의 생성 과정을 보여주는 순서도이다. 위의 녹색 상자 내부의 객체들은 Service Agent 내부에서 SUPPLE 을 위한 사용자 인터페이스 정의를 생성시켜 주는 파트이고, 아래의 노란 상자는 Remote Controller 에서 사용자 인터페이스를 생성하여 사용자에게 보여주는 부분이다.

먼저, UPnP 의 XML 정의 생성 규칙에 따라 정의된 XML 정의가 XML Parser 에 의하여 DOM Tree 형태로 Description Generator 에게 전달된다. 다음에는 그 정보를 바탕으로 Description Generator 가 SUPPLE 에서 이해 가능한 UI Description 을 생성시켜 준다. 생성된 UI Description 은 SUPPLE 에 의해서 실제 사용자 인터페이스를 생성하여 사용자에게 보여주게 된다.

3-2 상황에 따른 재구성

우리가 인터페이스 구성시에 고려한 상황 정보는 위치 정보이다. 가까운 기기 및 서비스들이 목록의 위쪽에 위치하게 된다. 이는 UWB 기반의 위치 인식 시스템인 Ubisense [8]을 사용하였고, UPnP 기기 및 서비스 목록들을 원격 서버로부터 받아서 랭킹 모듈으로 보낸다.

3-3 선호도에 따른 재구성

우리는 사람들이 자주 사용되는 기능들을 선호하는 기기 및 기능이라고 가정하였다. 이에 SUPPLE [4][5][6] 에서 구현되어있는 event tracing module 을 사용하려 했지만 SUPPLE 에서 개발된 것은 UI 상에서의 component 빈도만을 기록했기 때문에 기존에 존재하던 SUPPLE 의 것을 변경하여 사용하였다. 즉, 각 사용자 인터페이스 컴포넌트들이 자신들의 사용 기록 정보를 저장하게 하였다. 이 정보들은 랭킹 모듈로 보내져, 상황과 선호도에 따라 목록을 재구성하고, 인터페이스를 구성하게 된다.

3-4 접근권한에 따른 재구성

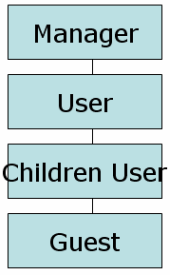


그림 5. 접근 권한 계층 구조

우리는 역할 기반 접근 제어 방식을 사용하기 위하여 그림 5 와 같이 네 가지의 역할을 계층적으로 정의하였다. Manager 는 모든 기기와 기능들을 제어할 수 있도록 하였다. Guest 의 경우에는 로그인을 통하여 신원이 확인되지 않은 사용자이기 때문에, Manager 에 의해 허용된 일부분의 기기 및 기능들을 제어 가능하도록 하였다. 각각의 유저는 한 가지의 역할을 가지게 되고, 그 역할에 따라 그들이 제어할 수 있는 기기와 기능들이 정해진다. 또한, 몇몇 기능들은 어떤 상태에서 특정한 규칙을 가지게 된다.

3-5 UI 자동 생성

우리는 기존의 UPnP 에서 사용하는 기기 정의와 서비스 정의들을 이용하여 사용자 인터페이스를 자동으로 생성할 수 있도록 구현하였다[7]. 이를 통해서, 우리는 같은 컴포넌트 그룹들을 제어 인터페이스 상에 단순 나열하는 것이 아닌, 각각의 Action 과 StateVariable 의 특성에 맞는 알맞은 컴포넌트들을 배치할 수 있도록 하였다.

아래의 표 1 과 2 는 Description Generator 가 UI Description 을 생성할 때에 사용하는 컴포넌트 생성 규칙들이다.

Modifiability	Length of Allowed Value(N)	Mapping GUI Component
Modifiable	N<=6	RadioButton
Modifiable	N>7	Listbox
Non-modifiable		Label

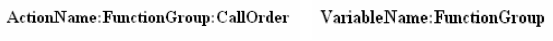
표 1. String Type 전환 규칙

Modifiabillity	Range	Mapping GUI Component
Modifiable	1 <= R <= 99	Listbox
Modifiable	100 <= R	Spinner
Non-modifiable		Label

표 2. Integer Type 전환 규칙

두 데이터 형태에서 공통적으로 사용되는 Modifiability 는 기기를 제어할 때에, 인터페이스 상에서의 변수 변경 필요성에 따라 달라진다. StateVariable 의 경우에는 기기의 현 상태를 보여주기 위해, 사용자가 조작할 수 없도록 제약을 두고, argument 로 들어가게 되는 변수는 사용자가 인터페이스 상에서 변경할 수 있도록 하였다.

또한, UPnP 서비스 정의에서는 각각의 Action 과 StateVariable 들이 하는 기능에 대한 정보와 하나의 기능을 위해 사용되는 Action 들의 일반적인 호출 순서에 대한 정보가 제공되어지지 않으므로, StateVariable 과 Action 을 위한 컴포넌트들의 단순 나열만이 가능하게 되었다. 여기서 우리는 UPnP 서비스 정의를 바꾸지 않고, 정보를 추가하는 방법으로 각각의 StateVaraible 과 Action 의 이름을 특정 구조로 표현하여, 정보를 추가하는 방식을 고안하였다. 다음의 그림 6 은 우리가 고안한 정보 표현 방식이다.



(a) (b)

그림 6. (a) Action 을 위한 이름 구조, (b) StateVariable 을 위한 이름구조

이와 같이 Action 과 StateVariable 의 이름에 우리가 원하는 정보를 추가할 수 있게 되어, 인터페이스 상의 컴포넌트들을 기능에 따라 그룹화하고, 또한, Action 들을 호출 순서에 따라 재정렬함으로써, Action 들을 올바른 순서에 따라 사용할 수 있도록 사용자들을 도왔다.

4. 프로토타입 구현

4-1 구현 환경

우리는 Java 를 기반으로 원격 제어 시스템을 구현하였다. 이는 SUPPLE 과 위치 시스템이 Java 로 만들어져 있었고, Java 가 가진 플랫폼 독립적인 특성을 이용하여 어떤 장치에서도 원격 제어 서비스를 제공할 수 있도록 하기 위해서이다

또한, 우리는 SUPPLE 이라는 UI 툴킷을 사용하여 UI 를 생성할 수 있도록 하였다. SUPPLE 은 프레임 사이즈에 맞추어 인터페이스를 자동적으로 재구성해 주는 특성을 가지고 있다.

그리고 위치 인식 시스템으로 UWB 기반의 Ubisense [8] 를 사용하였다. 이 시스템은 사용자의 위치에 따라서 기기 목록을 재구성하여 준다.

구현 시에는 프로젝트 관리 및 컴파일 능력을

갖추고 있는 Java Builder 로 Eclipse SDK 를 사용하였다.

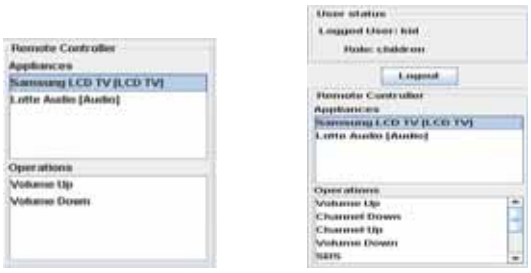
4-2 구현 결과

우리는 이미 위치시스템으로부터 정렬된 목록을 받아와서, 이를 랭킹 모듈에서 선호도 정보를 이용하여 재정렬하는 방식을 사용하였다.

(a) (b)

그림 7. (a) 상태에서 "Volume Up" 기능을 여러번 사용한 후에 (b)의 상태로 변화

사용된 기기와 기능의 이름들을 배열 목록에 추가하고, 사용된 횟수를 기록함으로 해서, 사용자가 선호하는 기기 및 기능들에 대한 정보를 적용시켰다. 하지만, 이러한 정보들이 적용될 때, 목록의 상태가 자주 변화되면, 사용자의 입장에서 불편함을 느낄 수 있기 때문에, 매 5 회의 기능 호출 이후에 목록을 업데이트하도록 하였다. 아래의 그림 7 은 사용자의 기능 호출에 따른, UI 상의 변화이다.



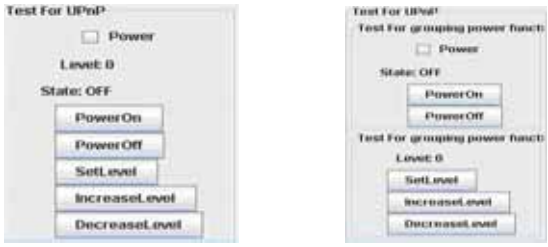
(a) (b)

그림 8. (a) guest 상태의 원격 제어 서비스의 모습, (b) kid 로 로그인된 사용자의 원격 제어 서비스 상태

그리고 우리는 유저의 역할과 접근 권한에 대한 정보를 데이터베이스에 저장해 두고, 사용자가 로그인하였을 경우에, 그 정보를 데이터베이스로부터 읽어서, 신원 확인을 할 수 있도록 하였다. 다음의 그림 8 은 로그인 전과 후의 원격 제어 서비스의 상태 변화를 보여준다.

그림 9. '12'번 채널을 사용하지 못하는 사용자에 대한 제한

그리고 이러한 역할 기반 접근 제어 방식의 예로, 12 번 채널을 성인 채널이라 가정하고, Children 이란 역할을 가진 사용자에 대해서 12 번 채널을 사용할 수 없도록 제한하였다. 그림 9 는 Children 역할의 사용자가 12 번 채널에 접근하려 했을 경우의 에러 메시지를 보여주는 그림이다.



(a) (b)

그림 10. (a) 컴포넌트 생성 규칙에 따라 생성된 UI (b) 그룹화와 Action 정렬 이후의 UI

그림 10. (a)는 그룹화와 호출 순서에 따른 Action 의 정렬이 되지 않은 인터페이스의 모습이다. 이전의 표 1 과 2 에 나타난 규칙에 따라 각각의 StateVariable 과 Action 들을 알맞은 컴포넌트로 변환시킬 수 있도록 하였다. XML 정의를 읽을 때에, 기존의 자바에서 제공되는 XML parser 를 이용하였고, 이에 따라 DOM 트리를 구성하였다. 이 트리로부터 재귀적 DFS 알고리즘을 이용하여 하나하나의 노드 정보를 받아들여 그들 정보를 데이터 형태에 따라, Boolean, Integer, String 으로 구분하였고, 쓰임새에 따라서, Argument, StateVariable 로 구분하여 각각 알맞은 UI 컴포넌트로 변환하였다.

또한, 각각의 문자열을 토큰화하여 그들 중에 그룹화와 Action 정렬을 위해 필요한 정보들을 추출하였다. 이에 따라, 컴포넌트 그룹화와 Action 정렬을 거친 후의 상태가 그림 10. (b)와 같다.

5. 결 론

우리는 사용자의 다양한 상황 정보를 이용하여 네트워크 상에 존재하는 기기들을 제어할 수 있는 통합 원격 제어 서비스를 개발하였다. 역할 기반 접근 제한 방식을 통한 제어 기능 허용과 각각의 사용자들의 사용 기록, 위치에 따른 목록의 정렬을 통해 유비쿼터스 상황에 적합한 통합 원격 제어 서비스를 개발해 보았다.

또한, 기존의 기기 제작자에 의한 인터페이스 제공이 아니라, XML 서비스 정의로부터 직접 인터페이스를 생성할 수 있게 함으로써, 서비스 정의 변경만을 통해 UI 를 변경할 수 있도록 하였다. 따라서, 기기 제작자들이 해야 할 일을 줄이고, 그들이 인터페이스를 개발하는 비용을 줄일 수 있게 된 것이다. 그리고 기능에 따른 컴포넌트들의 그룹화와 Action 들의 호출 순서에 따른 정렬을 통하여, 유저가 기기를 작동함에 있어, 보다 쉽게 제어할 수 있도록 하였다.

하지만 현재 SUPPLE 의 UI 최적화의 경우, 화면을 최적화시킬 뿐이지, 사용자의 편의를 생각하지 않았기 때문에, 가장 중요한 원격 제어 서비스 파트는 찾기 힘들어지고, ID 파트가 부각되는 등의 불편함이 나타났다. 따라서, 사용자의 편의에 맞게 인터페이스를 최적화시켜 주는 UI 툴킷에 대한 연구가 필요할 것이다.

또한, 이 원격 제어 서비스가 사용 기록을 사용함에 있어, 처음에는 그 기록의 양이 적기 때문에 목록의 정확도가 떨어질 가능성이 있다. 이는 장기간 사용시에는 보다 높은 정확도를 가질 수 있게 된다.

그리고 UPnP 서비스 정의에 더 많은 정보들을 추가함으로써, 보다 더 많은 사용자의 상황 정보를 이용하여 인터페이스 생성이 가능해지므로, 보다 더 나은 원격 제어 서비스 인터페이스를 생성시킬 수 있게 될 것이다.

6. Acknowledgement

본 연구는 21 세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅및네트워크원천기반기술개발사업과 정보통신부 및 정보통신연구진흥원의 디지털미디어연구소 지원사업의 연구결과로 수행되었음

7. 참 고 문 헌

[1] UPnP Forum, “UPnP Device Architecture 1.0,” December 2003.

[2] Intel, “Intel Software for UPnP Technology,” http://www.intel.com/cd/ids/developer/asmo-na/eng/downloads/upnp/index.htm.

[3] S. Konno, “CyberLink for Java”, available at http://www.cybergarage.org/net/upnp/java/.

[4] SUPPLE Homepage, http://www.cs.washington.edu/ai/supple/.

[5] K. Gajos and D. S. Weld, “SUPPLE: Automatically Generating User Interfaces,” in *Proceedings of IUI’04,* 2004.

[6] K. Gajos, R. Hoffmann, and D. S. Weld, “*Improving User Interface Personalization,*” in *Supplementary Proceedings of UIST’04*, 2004.

[7] M. Jeronimo, “UPnP Design by Example,” Intel Press, 2003.

[8] Ubisense, http://www.ubisense.net.