다중기기 제어를 위한 태스크 기반 사용자 인터페이스

임성수° 조성배
연세대학교 컴퓨터과학과
lss°@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

A Task-based User Interface for Manipulating Multiply Connected Appliances

Sungsoo Lim° and Sung-Bae Cho
Dept. of Computer Science, Yonsei University

요 약

최근 TV, DVD, AV 등의 각종 가전기기가 보급되면서 사용자는 다수의 리모컨을 조작하고 기기 사이의 관계를 이해해야 다양한 서비스를 제공받을 수 있게 되었다. 하지만 사용자는 보통 기기 단위의 기능을 일일이 제어하기 보다는 간단한 조작으로 다수 기기의 기능이 복합적으로 함축된 상위수준의 서비스를 제공받기 원한다. 기존에는 다수의 기기를 조작하기 위해 각 기기의 리모컨을 물리적으로 통합한 통합 리모컨 등이 제안되었으나 고수준의 서비스를 위해서는 사용자가 각 기기를 직접 조작해야 했고, 복수의 작업을 하나의 버튼으로 수행하는 매크로 기능 등은 사용자가 직접 설계해야 하거나 직관적이지 못하였다. 본 논문에서는 사용자에게 제공되는 서비스 작업 단위를 태스크로 정의하고 서비스 제공을 위해 필요한 기기 사이의 연결성과 각종 값을 내부적으로 설정하여 사용자에게 태스크 기반의 직관적인 인터페이스를 제공하는 방법을 제안한다. 기기를 일일이 조작해야 하는 기존의 통합 리모컨과 달리, 기기의 연결 상태 등을 분석하여 사용자가 이해하기 쉬운 태스크 단위로 사용자 인터페이스를 설계하고, 복수의 기기를 제어하는 작업은 기기의 상태에 따라 동작하여 사용자는 기기의 상태를 신경 쓰지 않고 서비스를 제공받는다. XML로 설계된 각 기기의 기능과 인터페이스 및 태스크의 기능과 인터페이스를 실시간으로 통합하여 현재 상태에 적적한 태스크 기반 사용자 인터페이스를 동적으로 생성한다. 본 논문에서는 사용성 평가를 통해 제안하는 방법의 유용성을 검증하였다.

1. 서 론

네트워크 기술이 발전함에 따라 가정내의 가전기기들도 유/무선을 통해 연결되고 있다. 무선 키보드/마우스, 이어폰, 마이크 등과 같은 컴퓨터 주변기기들뿐만 아니라, 홈 씨어터 시스템과 같이 TV 셋탑박스, DVD 플레이어, 오디오, 비디오 등과 같은 다수의 장치들도 서로 연결되어 서비스를 제공한다. 또한, 유비쿼터스 컴퓨팅이 실현되면, TV, 에어컨 등과 같이 기존에 리모컨으로 조작하던 가전기기들 외에도 조명, 창문, 보일러, 냉장고 등의 가전기기들도 하나의 네트워크로 연결되어 조작이 가능할 것이다.

다양한 기기들이 서로 연결되어 서비스를 제공하는 반면에, 각 기기를 제어하기 위한 인터페이스는 통합되지 않고 각각 존재한다[1]. 이러한 환경에서 사용자는 원하는 서비스를 받기 위해, 기기 제어 방법뿐만 아니라 기기간의 연결 관계도 숙지하고 있어야 한다. 현재 가정에 보급되고 있는 많은 가전기기들은 제조업체에서 제공되는 리모컨을 통해서 제어가 가능하다. 하지만 가전기기의 수가 늘어나고 기기간 연결이 복잡해 짐에 따라서 사용자가 기존의 리모컨을 통하여 원하는 서비스를 제공받기에는 불편함과 어려움이 있다. 가전기기 제조사는 보통 제품이 가지고 있는 모든 기능을 제어할 수 있도록 리모컨을 만들기 때문에 많은 수의 버튼을 만들지만, 그 사용성은 1/3에 그치며[2], 오히려 사용자에게 혼란을 초래한다. 또한, 홈 씨어터 시스템을 통한 영화관람의 경우, 케이블이나 위성 TV 셋탑박스, DVD 플레이어, 오디오, 비디오 등과 같이 다수의 장치를 동시에 조작해야 하고, 원하는 기능을 실행하기 위해서 적절한 리모컨을 선택하여야 한다. 다른 기기를 위한 리모컨은 유사해 보이지만, 기기나 제조사에 따라 디자인이 다르므로 많은 장치를 위한 리모컨을 모두 익숙하게 사용하는 것은 생각보다 어려운 일이며[3], 특정 기능을 수행하기 위해 다수 리모컨의 동시 조작이 필요한 경우도 있어 사용자를 더욱 어렵게 한다.

본 논문에서는 사용자에게 제공되는 상위수준의 서비스 작업 단위를 태스크로 정의하고, 이러한 태스크 단위의 제어를 생성하기 위한 인터페이스 설계 언어인 HDML(Home Devices Markup Language)을 제안하며, 이를 이용하여 태스크 기반 사용자 인터페이스를 구축한다.

2. 다중기기 제어를 위한 사용자 인터페이스

가정환경에서 다양한 기기들을 동시에 조작하도록 다수의 리모컨을 통합하려는 시도가 몇몇 업체들을 중심으로 진행되고 있다. 최근 TV 리모컨의 경우, 모드를 바꾸어 TV 뿐만 아니라 DVD 플레이어, 비디오, 오디오 등의 기기를 동시에 조작하도록 돕기도 하며, 소니는 각종 기기의 적외선 신호를 기억하여 하나의 리모컨으로 제어하는 통합 리모컨을 개발하기도 하였다. 뿐만 아니라 가정환경에서의 다양한 기기들에 접근하기 위해 마이크로

소프트와 인텔을 중심으로 진행 중인 UPnP (Universal Plug and Play), 소니와 필립스 등이 제정한 HAVi (Home Audio Video interoperability), 썬 마이크로 시스템이 중심이 된 Jini, 그리고 에체렌의 제어 네트워크 기반기술인 Lonworks 등의 기술이 개발되었으며, ConvergeX 사의 경우, 이들 기반 기술을 이용하여 가정환경을 조작하는 통합 사용자 인터페이스를 구현하였고, HomeLogic 사는 이미 700개 이상의 가정환경 제어 시스템을 공급하였다. 하지만 이들 기술과 제품은 사용자가 가정기기의 각종 기능을 사용하도록 리모컨과 기기를 연결시켜 줄 뿐이어서 원하는 상위수준의 작업을 하기 위해서는 다수의 장치를 제어하기 위한 복잡한 조작이 필요하다.

앞에서 언급한 것과 같이 다양한 기기의 리모컨을 하나로 통합하려는 시도는 사전에 정의된 환경에서만 적용이 가능하며, 사용자는 원하는 서비스를 제공받기 위해 여전히 많은 조작을 해야 한다는 단점이 있다. 이를 극복하기 위해 다중 기기를 효과적으로 제어하는 사용자 인터페이스에 관한 연구가 활발히 진행되고 있다. Nichols와 Myers는 가정과 사무실의 각종 기기를 XML 형태로 기술하고 스마트 폰으로 통합된 인터페이스를 제공하여 기기들을 제어하는 기술을 제안하였다[4]. 다양한 기기를 스마트 폰이라는 하나의 매체를 통해 제어할 수 있다는 장점이 있지만, 여러 기기가 복합적으로 연동하여 제공하는 서비스나 기기 제어는 불가능하고 사전에 모든 기기의 인터페이스가 미리 설계되어야 한다.

Isbell 등은 사용자의 기기에 대한 사용 버튼을 분석하여 주요 태스크를 선정하고 사용자별로 개인화된 사용자 인터페이스를 생성하여 제공하였다[5]. 이는 다양한 기기를 스마트 폰이라는 하나의 매체를 통해 제어할 수 있다는 장점이 있지만, 여러 기기가 복합적으로 연동하여 제공하는 서비스나 기기 제어는 불가능하고 사전에 모든 기기의 인터페이스가 미리 설계되어야 한다.

CMU의 Nichols 그룹은 사용자가 보다 편하게 복수의 기기를 제어하도록 태스크 중심의 사용자 인터페이스를 자동으로 생성하는 시스템을 개발하였다[3]. 태스크 별로 각종 기기의 연결관계를 사용자가 명시하면 기기별로 사전에 설계된 인터페이스를 중심으로 해당 태스크를 위한 인터페이스가 자동으로 생성된다. 태스크 기반의 사용자 인터페이스를 제공하기 때문에, 사용자는 복수의 기기를 일일이 조작할 필요가 없고 상위 수준의 명령을 내리면 내부적으로 다수의 기기가 제어된다. 하지만 기기 사이의 연결 관계를 명시해야 하고 생성되는 인터페이스가 단순한 기기 인터페이스의 조합 수준에 머물러있다.

3. 태스크 기반 사용자 인터페이스

사용자가 홈 씨어터 환경과 같이 다양한 기기들이 연결되어서 하나의 서비스를 제공하는 환경에서, 각 기기의 리모컨을 통해 원하는 태스크를 수행하기 위해서는 첫째, 주변에 태스크와 연관된 기기들이 있는지 확인하고, 둘째, 각 기기의 상태를 확인한 후, 셋째, 원하는 태스크 수행을 위해 적절한 기기에 적절한 명령의 전달이 필요하다. 본 논문에서는 HDML(Home Device Markup Language)의 설계와 이를 용한 태스크기반 인터페이스를 통해서, 사용자가 의미정보를 기반으로 손쉽게 태스크를 수행 할 수 있도록 하는 통합리모컨을 제안한다.

```
device ::= "<device>" dev_id dev_name dev_concept (dev_src) (dev_des) variable {variable} functions "</device>"
dev_id ::= "<dev_id>" $string "</dev_id>"
dev_name ::= "<dev_name >" $string "</dev_name>"
dev_src ::= "<dev_src >" src_concept { src_concept } "</dev_src >"
dev_des ::= "<dev_des >" des_concept { des_concept } "</dev_des >"
variable ::= digit_variable | enum_variable
digit_variable ::= "<variable type = digit mode = " ("parallel" | "harmonic" | "src" | "des") step " = " $float ">" var_id var_concept (src_concept) min max "</variable>"
enum_variable ::= "<variable type = enum mode = " ("parallel" | "harmonic" | "src" | "des") ">" var_id var_concept (src_concept) value {value} "</variable>"
var_id ::= "<var_id >" $string "</var_id>"
min ::= "<min>" $float "</min>"
max ::= "<max>" $float "</max>"
value ::= "<value>" $string value_concept "</value>"
functions ::= "<functions>" protocol button {button} "</functions >"
protocol ::= "<protocol format = " $string "/>"
button ::= toggle_button | set_button | inc_button | dec_button
toggle_button ::= "<button type = toggle var_id = " $string ">" btn_id btn_name btn_concept ir_code "</button>"
set_button ::= "<button type = set var_id = " $string " value = " $string ">" btn_id btn_name btn_concept ir_code_value "</button>"
inc_button ::= "<button type = inc var_id = " $string ">" btn_id btn_name btn_concept ir_code "</button>"
dec_button ::= "<button type = inc var_id = " $string ">" btn_id btn_name btn_concept ir_code "</button>"
btn_id ::= "< btn_id >" $string "</btn_id>"
btn_name ::= "< btn_name >" $string "</ btn_name>"
ir_code ::= "<ir_code delay = " $int ">" ( 0 | 1 | # | ? ) { 0 | 1 | # | ? } "</ir_code>"
```

그림 1. 기기 기술언어를 위한 BNF 형식

3.1 HDML

HDML은 크게 기기 기술 부분, 태스크 기술 부분, 동의어 기술 부분의 3가지로 구성된다. 기기 기술 부분은 통합리모컨으로 하여금 기기의 내부 상태 알게 하고 기기 조작을 가능하도록 하는 정보를 포함하고, 태스크 기술 부분에서는 사용자가 원하는 태스크 조작을 위한 정보와 인터페이스 구성을 위한 버튼들 간의 관계 정보가

기술된다. 마지막으로 동의어 기술 언어는 태스크 기술 부분에 기술된 기기 및 기기 기능과 실제 기기 및 기기 기능간의 연결 관계 설정을 위해 사용된다.

기기 기술 언어는 통합리모컨으로 하여금 주변 기기 정보를 인식할 수 있도록 기기에 대한 정보를 기술 하는 언어이다. 그림 1의 기기 기술언어를 위한 BNF 형식에서 보는 바와 같이, 기기정보는 통합리모컨이 기기들을 구분하도록 하는 아이디(*dev_id*)와 사용자에게 보여질 기기명(*dev_name*), 영상, 음성 등과 같이 각 기기가 생성하는 신호(*dev_src*)와 기기 최종적으로 수신할 수 있는 신호(*dev_des*)가 명시된다. 그리고 기기의 내부 상태를 표현하기위한 변수(*variable*)는 음량, 채널과 같이 실수 값을 상태 값으로 갖는 변수들을 표현하기 위한 *digit_variable*과, 채널(on/off), 외부입력(외부입력1, 외부입력 2, RGB 등) 등과 같은 상태 값을 갖는 변수들을 표현하기 위한 *enum_variable*로 구성된다. 마지막으로 *function*은 리모컨의 통신 방법을 나타내는 *protocol*과 리모컨의 각 버튼을 나타내는 *button*들로 구성되며, *ir_code*를 통해서 버튼이 눌러졌을 때에 보내지는 적외선 신호 값이 인코딩된다.

태스크 기술 언어는 만능리모컨이 태스크 기반 인터페이스를 생성하기 위한, 태스크 기본 정보, 태스크 수행을 위한 선행 조건과 태스크 수행을 위해 필요한 버튼에 대해 기술한다. 그림 2는 태스크 기술 언어를 위한 BNF 형식을 보여준다.

```
task ::= "<task>" tsk_id tsk_name tsk_concept {tsk_src}
 {tsk_des} tsk_function "</task>"
tsk_id ::= "<tsk_id >" $string "</tsk_id>"
tsk_name ::= "<tsk_name>" $string "</tsk_name>"
tsk_src ::= "<tsk_src >" (dev_concept | dev_concept
 src_concept) "</tsk_src>"
tsk_des ::= "<tsk_des>" (des_concept | dev_concept
 des_concept) "</tsk_des>"
tsk_function ::= "<functions>" tsk_button {tsk_button}
 group {group} "</functions>"
tsk_button ::= "<button>" tsk_btn_id tsk_btn_name
 (dev_concept var_concept value_concept | var_concept
 value_concept | dev_concept btn_concept | btn_concept)
 "</button>"
tsk_btn_id ::= "<btn_id >" $string "</btn_id>"
tsk_btn_name ::= "<btn_name>" $string "</btn_name>"
group ::= "<group>" grp_concept grp_btn {grp_btn}
 relation {relation} "</group>"
grp_btn ::= "<button btn_id = '$string'/>"
relation ::= t2b_relation | l2r_relation | hor_relation
t2b_relation ::= "<relation type = top2bottom>" button
 button {button} "</relation>"
l2r_relation ::= "<relation type = left2right>" button
 button {button} "</relation>"
hor_relation ::= "<relation type = horizontal>" button
 button {button} "</relation>"
```

그림 2. 태스크 기술언어를 위한 BNF 형식

통합리모컨이 태스크를 수행하기 위해서는 먼저 태스크 수행에 필요한 환경이 갖추어 졌는지를 확인해야 한다. 본 논문에서는 태스크 수행을 위한 신호 생성 기기가 무엇이며 최종적으로 이 신호를 수신하는 기기가 무엇인지에 대해서 명시하고, 기기의 내부 상태 값을 보고 각 기기가 어떤 기기와 연결되었는지를 따라가며 연결된 기기들을 찾는다. *tsk_src*와 *tsk_des*는 각각 태스크 수행을 위한 신호 생성 기기와 최종 신호 수신 기기를 명시한다. *tsk_src*를 통해서 태스크 수행의 소스 기기를 기술하거나 특정 기기의 신호를 기술한다. 비슷하게 *tsk_dev*를 통해서는 최종적으로 처리할 신호나 신호 처리를 위한 기기를 기술한다

*tsk_function*에서의 *group*은 유사한 속성을 갖는 여러 버튼의 그룹을 의미한다. 모든 그룹은 하나 이상의 *button*으로 이루어지며, 그룹을 구성하는 버튼들은 그룹의 특성에 따라 서로 관계를 갖는다. 이 관계는 기기 기술자를 설계할 때 함께 정의되며 *relation*을 사용해 기술된다. 버튼들이 가질 수 있는 가능한 관계는 *top2bottom* (상하관계), *left2right* (좌우관계), *horizontal* (병렬관계)의 세 가지가 있다. 이 관계 정보는 나중에 사용자 인터페이스 구성 시, 그룹 내의 버튼 배치를 위한 레이아웃을 결정하는 데에 사용된다. 그림 3은 *relation type*에 따른 레이아웃 배치를 보여준다. (a)는 소리 올림과 소리 내림 버튼을 상하 관계로 배치하고자 할 때, (b)는 재생, 일시 정지, 정지 버튼을 병렬 관계로 배치하고자 할 때, (c)는 번호선택 1, 2, 3을 좌우 관계로 배치하고자 할 때의 태그 및 실제 버튼 배치를 보여준다. 버튼의 모양은 임의로 적절한 것을 사용했으며, 그림에서 'O'는 바로 배치된 것을 의미하고, 'X'는 잘못 배치된 것을 의미한다. 병렬 관계(b)에서는 순서가 중요하지 않으며, 상하 관계(a)와 좌우 관계(c)에서는 순서가 중요하므로 버튼 배치 시에 이를 고려해 주어야 한다.

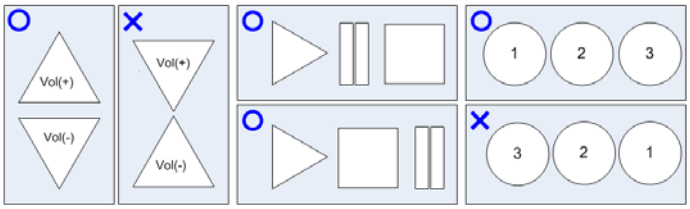

그림 3. 버튼 간의 관계 정보

마지막으로 동의어 기술 언어는 기기 기술 언어와 태스크 기술 언어 간의 의미 정보 기반 연결을 위해서 사용된다. 기기는 시간 공간에 따라서 얼마든지 변할 수 있으므로 기기 기술 언어와 태스크 기술 언어 간의 정적 연결은, 환경의 변화에 있어 기기 기술 언어뿐만 아니라 태스크 기술 언어도 변경이 필요하다. 각 기기에 대한 정보는 기기 제조사에서 생성해준다고 가정하면, A사에서는 전원 기능을 'power'으로 기술하고, 동일 가전제품을 취급하는 B사에서는 'pw'로 기술하는 것과 같이 같은 기능을 다르게 표기할 수 있는데, 의미 정보 기반 매핑을 통해 'power'와 'pw'가 동일한 기능을 가짐을 명시하면 이러한 문제는 간단히 해결될 수 있다. 따라서 본

과제에서는 동의어 정보를 기술하기 위해 그림 4와 같은 동의어 기술 언어를 정의한다.

```
representation ::= '<representation>' $string '</representation>'
synonym ::= '<synonym>' $string '</synonym>'
semantic ::= '<semantic>' representation synonym {synonym} '</semantic>'
semantic_info ::= '<semantic_info>' semantic {semantic} '</semantic_info>'
```

그림 4. 동의어 기술 언어를 위한 BNF

본 논문에서는 두 의미 정보의 의미적 유사도를 계산하기 위해 F-measure를 적용한다. F-measure는 정확률과 재현율을 함께 고려하는 성능을 측정하는 도구로 다음과 같이 평가 값을 얻는다. 본 논문에서는 정확률과 재현율을 동등하게 고려하기 위해 F-measure의 α값을 1로 설정하였다.

$$F-measure = \frac{(a+1)\times precision \times recall}{a\times precision + recall}$$

$$precision = \frac{A}{A+B},\ recall = \frac{A}{A+C}$$

기기 기술 언어 / 태스크 기술 언어	포함	미포함
포함	A	B
미포함	C	D

A, B, C, D : 빈도 수

3.2 태스크 기반 사용자 인터페이스 생성 및 동작 과정

본 절에서는 통합리모컨이 위의 기기 기술 언어, 태스크 기술 언어, 동의어 기술 언어로부터 어떻게 태스크기반 인터페이스를 구성하고 사용자에게 서비스를 제공하는지에 대해서 알아본다. 만일 어떤 사람이 새로운 환경에서 어떤 태스크를 수행하고자 한다면, 이 사람은 첫째, 자신이 수행하고자 하는 바를 인식하고, 둘째, 주변 환경에 어떠한 기기들이 어떻게 연결되어 있는지 파악한 후, 셋째, 태스크 수행에 필요한 기기들의 전원을 키고 기기들 간의 연결 관계를 세팅하고, 마지막으로 태스크 제어에 필요한 기능을 적절히 사용하여 태스크를 수행할 수 있다.

본 논문에서는 통합리모컨으로 하여금 위와 같은 사용자의 수고를 덜어, 주변 환경에 상관없이 사용자가 원하는 태스크를 수행할 수 있도록 태스크 기반 사용자 인터페이스를 구성한다. 태스크 기반 인터페이스 위해서 통합리모컨은 위의 사용자 예와 같이 첫째, HDML을 이용하여 기술된 주변 기기 및 사용자 태스크 정보를 얻어오고, 둘째, 현재 환경에서 수행 가능한 태스크 목록을 작성하고, 셋째, 사용자가 태스크를 선택하면 태스크 수행을 위한 UID를 생성하고, 넷째, 생성된 UID를 바탕으로 태스크 기반 사용자 인터페이스를 제공한다.

그림 4는 제안하는 방법의 시스템 동작과정을 보여준다. HDML 파서는 기기 HDML과 태스크 HDML, 그리고 동의어 사전에 있는 정보를 파싱하여 각 객체를 메모리에 로드하고, 기기와 태스크 간의 연결관계를 동의어를 기반으로 매핑한다. 태스크 검사 모듈은 현재의 기기 상태를 바탕으로 수행 가능한 태스크를 추출한다. 사용자는 태스크 선택 모듈을 통해서 수행하고자 하는 태스크를 선택하고, UID 생성기는 선택된 태스크와 기기 정보 및 기기 상태 정보를 바탕으로 태스크 기반 사용자 인터페이스 생성을 위한 기술 언어인 UID를 생성하고, 이렇게 생성된 UID는 UI 렌더러를 통해서 태스크 기반 사용자 인터페이스를 구축하게 된다. 그림 5는 시스템의 동작 과정을 보여준다.

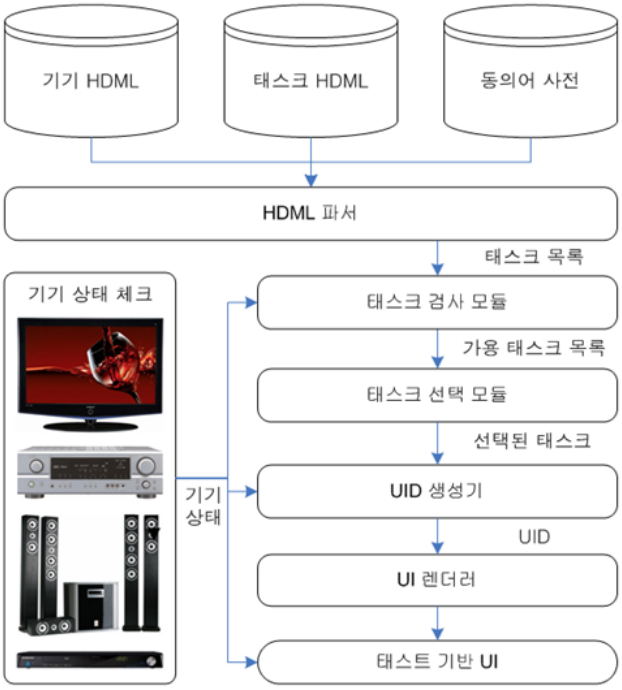

그림 4. 태스크기반 UI 생성 과정

(a) 태스크 선택 (b) 그룹 버튼 확인 (c) UI 생성

그림 5. 시스템 동작 과정

4. 실험 및 결과

본 논문에서는 제안하는 태스크 기반 사용자 인터페이

스의 성능을 평가하기 위해서 일반 통합리모컨 환경과 제안하는 통합리모컨 환경을 비교 평가한다. 실험은 10명의 피험자로부터 일반 통합리모컨 환경과 제안하는 통합리모컨 환경에서의 사용을 한 후, 태스크 수행을 위해서 사용한 버튼의 횟수 및 설문조사를 통해서 이루어 졌다.

피험자는 우선 TV, DVD가 갖춰진 환경에서 일반 통합리모컨과 통합리모컨을 이용하여 표 1과 같은 태스크를 수행한 후, AV 리시버가 추가된 환경에서 동일한 태스크를 수행하도록 하였다. 각 기기의 초기 상태는 표 2에서 보여주고, 표 3은 기기 환경에 따른 기기 연결 관계를 보여준다.

표 1. 사용자 태스크

태스크		내용
TV 보기	채널 변경	TV의 채널을 11번으로 변경한다.
	소리 스끄럽게	소리를 시끄러운 정도로 설정한다.
	소리 조용하게	소리가 작게 들릴 정도로 설정한다.
DVD 보기	DVD 재생	DVD를 재생한다.
	소리 시끄럽게	소리를 시끄러운 정도로 설정한다.
	소리 조용하게	소리가 작게 들릴 정도로 설정한다.

표 2. 기기의 초기 상태

기기	상태
TV	전원: off, 소리: 10, 채널: 35, 외부입력: TV
DVD	전원: off, 상태: 정지
AV리시버	전원: off, 소리: 10, 채널: 35, 외부입력 1, 외부출력 1

표 3. 기기 환경에 따른 기기 연결 상태

기기 환경	연결 상태
TV-DVD	TV 외부입력1에 DVD의 영상, 음성 연결
TV-AV 리시버 -DVD	TV 외부입력1에 AV리시버의 영상, 음성 연결 AV리시버 외부출력2에 TV의 영상, 음성 연결 AV리시버 외부입력2에 DVD의 영상, 음성 연결

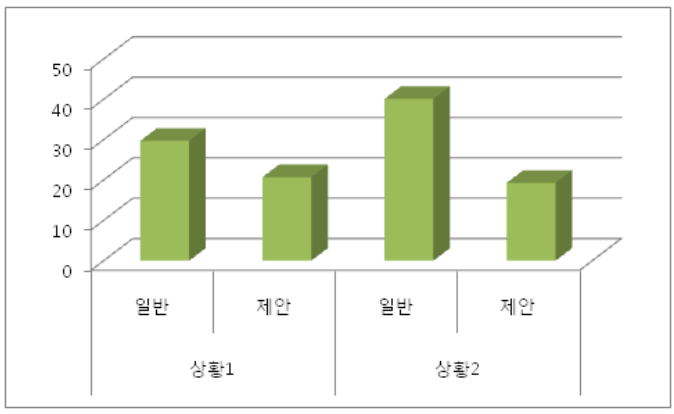

그림 6. 상황에 따른 버튼 클릭 횟수

그림 6은 실험 결과를 보여주고 있다. 상황 1은 기기 환경이 TV와 DVD인 경우이고, 상황 2는 TV, AV리시버, 그리고 DVD를 이용한 경우이다. 실험 결과에서 보여주듯이 제안하는 방법은 기기의 주변 환경에 상관없이 약 20번의 리모컨 클릭 횟수를 보이지만, 일반 통합리모컨을 이용한 경우에는 기기의 개수가 많아짐에 따라서 설정해야 할 기기 상태가 많아지기 때문에, 평균 10회 이상의 설정이 필요함을 알 수 있다.

5. 결 론

본 논문에서는 태스크에 참여하는 개별 기기의 User Interface Description으로부터 태스크 제어에 적합한 사용자 인터페이스를 실시간으로 자동 생성하기 위한 기술 개발을 수행하였다. 사용자는 가정환경의 각종 기기를 일일이 제어하기보다는 상위수준의 서비스를 제공받기 원하는데, 사용자에게 제공되는 태스크 단위로 인터페이스를 제공하여 하나의 리모컨으로 원하는 서비스를 제공 받을 수 있으며, 기기 사이의 연결성과 각종 설정에 대해 정확하게 숙지하지 않고도 사용할 수 있다.

가정환경에서 보다 다양한 서비스와 관련된 태스크 기반 사용자 인터페이스를 제공하기 위해서 기기 명세와 태스크 명세를 늘려야 할 뿐만 아니라 정보 소스나 의미 정보 사이의 관계를 보다 체계적으로 설계할 필요가 있으며, 기기나 태스크 명세의 설계를 용이하도록 효과적인 설계툴이 개발될 필요가 있다. 또한 사용자에게 개인화된 사용자 인터페이스를 제공하도록 기기와 태스크를 분석하여 사용자 인터페이스를 생성할 때 개인의 성향이나 취향을 함께 고려한 방법이 필요하다.

감사의 글

본 연구는 (주)삼성전자 종기원의 지원에 의해 수행되었습니다.

참고문헌

[1] D. Maddy, "Interfaces for consumer products: How to camouflage the computer?," *Proc. Sigchi COnf. Human Factors in Computing Systems, ACM Press*, pp. 287-290, 1992.

[2] Nielsen, “Remote control anarchy," http://www.useit.com/alertbox/20040607.html

[3] J. Nichols et al., “Huddle: Automatically generating interfaces for systems of multiple connected appliances,” *UIST 2006*, pp. 279-288, 2006.

[4] J. Nichols and B. Myers, “Controlling home and office appliances with smart phones,” *Pervasive Computing*, 2006.

[5] C. Isbell et al., “From devices to tasks: Automatic task prediction for personalized appliance control,” *Personal Ubiquitous Computing*, 2004.