SmartThings의 SmartApp을 위한 페이지 구조 시각화 도구

박나연[1] 창병모[2] 최광훈[3]
[12]숙명여자대학교 컴퓨터과학과
[3]전남대학교 전자컴퓨터공학부
b_newyork@sm.ac.kr, chang@sm.ac.kr, kwanghoon@jnu.ac.kr

Visualization tool for page structure of SmartThings Applications

Na-Yeon Bak[1] Byeong-Mo Chang[2] Kwanghoon Choi[3]
[12]Division of Computer Science, Sookmyung Women's University
[3]Dept. of Electronics and Computer Engineering, Chonnam National University

요 약

증가하는 디바이스들의 개수와 늘어나는 인터넷 연결로 사물인터넷의 영향력은 더욱더 강해질 것으로 예측된다. 그중에서도 삼성의 스마트싱스는 다수의 디바이스와 SmartApp을 관리, 개발하는 개방형 사물인터넷 플랫폼이다. 사용자는 자신이 등록한 SmartApp과 디바이스들을 통해 스마트싱스로부터 자동화 서비스를 제공 받는다. 본 논문에서는 SmartApp의 페이지 구조를 시각화하는 도구를 설계, 구현했다. SmartApp의 'Preferences', 'Predefined Callbacks', 'Event Handlers'에서 정보를 수집하고 수집한 정보를 바탕으로 SmartApp의 페이지 구조를 시각화 한다.

1. 서 론

사물인터넷(IOT)은 각종 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술이다. 전 세계적으로 증가하는 디바이스들의 개수와 늘어나는 인터넷 연결로 사물인터넷의 영향력은 더욱 강해질 것으로 예측되고 있다. 그중에서도 삼성의 스마트싱스(SmartThings)는 다수의 디바이스와 SmartApp을 관리, 개발하는 사물인터넷 플랫폼이다. 사용자는 SmartApp과 디바이스를 통해 스마트싱스로부터 서비스를 제공 받는다. 스마트싱스의 강점은 삼성전자 제품 이외에 1000개 이상의 디바이스와 8000개 이상의 애플리케이션을 지원하는 유연성에 있다. 하지만 아직 스마트싱스의 영향력은 상대적으로 작으며 그 원인 중 하나는 SmartApp 개발의 어려움 때문이라고 판단된다.

본 논문에서는 SmartApp의 개발과 유지보수에 편의성을 지원하는 시각화 도구(visualization tool)를 설계, 구현한다. 시각화 도구는 복잡하고 방대한 양의 코드를 계층적인 구조로 보여준다. 그로써 프로그램이 어떻게 구성되어 있는지를 이해하고 점검하는데 시각적으로 도움을 받는다.[1] 시각화 도구에 대한 연구는 많이 진행되었는데 Groovy 스윙 콘솔의 Groovy AST 브라우저는 Groovy 코드를 시각화한다. Groovy AST 브라우저를 통해 Groovy 프로그램의 구조를 계층적으로 시각화함으로써 프로그램을 이해하고 유지, 보수, 개발하는데 많은 도움을 받는다. 하지만 Groovy AST 브라우저의 시각화는 Groovy에 특화된 시각화 도구이기 때문에 SmartApp를 시각화 하는데 적절하지 않다.[4]

스마트싱스는 사용자에게 사용자가 등록한 SmartApp과 디바이스들로 자동화 서비스를 제공한다. 디바이스 등록은 SmartApp의 구성 요소 중 하나인 Preferences의 page에서 이루어진다. 그래서 SmartApp의 페이지 구조를 이해하는 것은 SmartApp의 구조를 이해하는데 많은 도움이 된다. 이에 따라 본 연구에서는 SmartApp의 페이지 구조와 디바이스를 중심으로 SmartApp의 구조를 시각적으로 보여주는 도구를 설계, 구현하였다. 이 도구는 스마트싱스의 SmartApp이 어떠한 구조로 어떤 디바이스의 기능을 통하여 어떤 서비스를 제공하는 지를 간결하게 보여준다.

2. 스마트싱스의 주요 구조

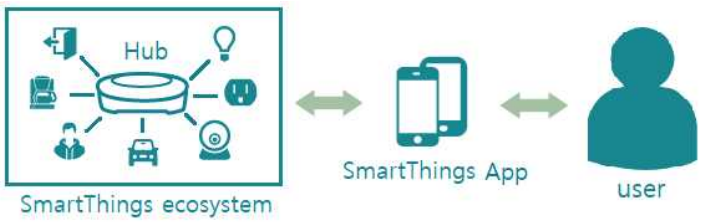

그림.1 스마트싱스의 주요 구조

스마트싱스는 다수의 디바이스와 SmartApp을 관리, 개발하는 개방형 사물인터넷 플랫폼이다.[3] 스마트싱스는 허브(Hub)를 통해 디바이스들을 관리한다. 허브를 중심으로 연결된 디바이스의 집합을 스마트싱스 생태계(SmartThings ecosystem)라고 한다. SmartApp은 디바이스의 기능을 이용하여 사용자에게 자동화 서비스를 제공하는 어플리케이션이다. 스마트싱스에 SmartApp을 등록하면 연결된 디바이스로 사용자에게 서비스를 제공한다.

SmartApp은 'Definition', 'Preferences', 'Predefined Callbacks' 그리고 'Event Handlers'로 구성된다.

- Definition : 그림.2와 같은 형태로 어플리케이션의 아이콘과 이름 등 어플리케이션을 식별하고 설명하는 여러 정보를 기술하는 부분이다.

```
definition(
    name: "Left It Open",
    namespace: "smartthings",
    author: "SmartThings",
    description: "Notifies you when a door or window open.",
    category: "Convenience",
    iconUrl: "https://s3.amazonaws.com/smartapp-icons/ModeMagic/bon-voya
    iconX2Url: "https://s3.amazonaws.com/smartapp-icons/ModeMagic/bon-voy
)
```

그림.2 Definition 코드

• Preferences : 사용자에게 자동화 서비스를 제공하기 위해 필요한 input 디바이스의 기능과 여러 정보들을 정의하는 부분이다. Preferences는 page, section, input등으로 구성되며 아래 그림.3과 같이 preferences는 1개 이상의 page로 구성되고 page는 1개 이상의 section으로 구성되고 section은 1개 이상의 input으로 구성된다.

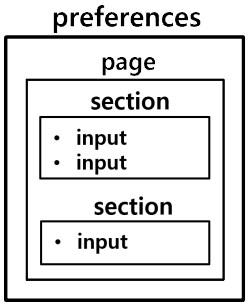

그림.3 Preferences 구조

```
preferences {
    page(name: "normalPage"){
        section("Monitor this door or window") {
            input("contact", "capability.contactSensor")
            input("switchf", "capability.switch")
        }
        section("Via text message at this number") {
            input("phone", "phone")
        }
    }
}
```

그림.4 Preferences 코드

page는 normal page와 dynamic page, 두 종류가 있는데 한 개의 page가 한 개의 화면을 정의한다. normal page는 1개 이상의 section으로 구성한 정적인 화면을 나타낸다. dynamic page는 section뿐만 아니라 변수의 정의, 연산, 함수 호출 등이 가능하여 동적으로 화면을 구성할 수 있다.

section은 input 등 SmartApp의 입력 요소들로 구성된다. 여러 요소들을 한 section안에 모아서 구성함으로써 사용자의 직관적인 SmartApp 설정을 지원한다.

input은 input의 이름과 디바이스의 기능을 지정하여 특정 디바이스나 변수를 정의할 수 있다. 정의된 디바이스의 기능과 정보는 input의 이름으로 참조할 수 있다.

• Predefined Callbacks: 어플리케이션의 설치, 갱신, 삭제를 지원하는 installed함수, updated함수를 호출하는 부분으로 디바이스와 SmartApp을 연결하는 subscribe함수를 호출하는 부분이다.

subscribe함수는 특정 디바이스의 어떤 이벤트가 일어났을 때 명시된 이벤트 핸들러 함수(event handler method)를 호출하는 기능을 수행한다. 디바이스의 정보를 저장한 input, 디바이스의 이벤트, 이벤트 핸들러 함수로 구성된다.

```
def installed() {
  subscribe(contact, "contact.open", doorOpenHandler)
}
def updated() {
  unsubscribe()
  subscribe(contact, "contact.open", doorOpenHandler)
}
```

그림.5 Predefined Callbacks 코드

• Event Handlers : 어플리케이션을 구성하는데 필요한 이벤트 핸들러 함수, dynamic page를 정의하는 함수 그리고 그와 관련된 다른 함수들을 정의하는 부분이다.

```
def doorOpenHandler(evt) {
  log.trace "doorOpen($evt.name: $evt.value)"
  switchf.on()
}
```

그림.6 Event Handlers 코드

3. 주요 기능 소개

본 시스템의 주요 기능은 SmartApp의 페이지 구조를 트리(Tree)의 형태로 시각화하는 것이다. preferences를 구성하는 page들의 구조를 중심으로 시각화를 진행하고 subscribe함수 정보를 수집하여 디바이스의 정보를 저장하는 input의 이름, 해당 디바이스의 기능, 이벤트 핸들러 함수의 등록 관계를 시각화했다.

‘Preferences’, ‘Predefined Callbacks’, ‘Event Handlers’ 부분을 참고하여 page와 subscribe함수 그리고 그와 관련된 여러 함수들의 정보를 수집하고 그 정보를 바탕으로 SmartApp의 페이지 구조를 아래 그림.7과 같은 트리의 형태로 시각화한다.

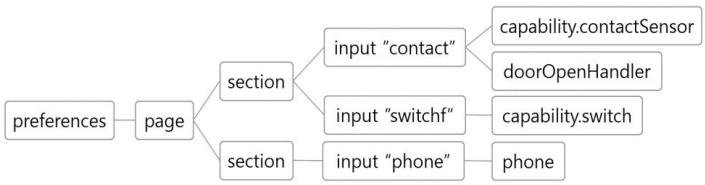

그림.7 시각화 모델

트리에서 preferences 노드는 1개 이상의 page를 자식노드로 가지고 있다. page 노드는 section을 자식노드로 가지고 있고 section 노드는 input을 자식노드로 가지고 있다. input 노드는 해당 input으로 참조되는 디바이스의 기능, 연결된 이벤트 핸들러 함수 명을 자식노드로 표현한다.

4. 시스템 구조

본 시스템의 구조는 아래 그림.8과 같이 Parsing, AST(Abstract Syntax Tree), “자료 수집’ 모듈과 ‘시각화’ 모듈로 구성된다.

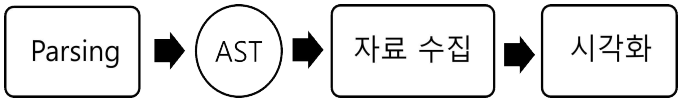

그림.8 시스템 구조

자료 수집 모듈에서는 SmartApp의 코드 조사하여 시각화에 필요한 자료를 수집한다. 시각화 모듈에서는 수집한 자료를 기반으로 SmartApp의 페이지 구조를 트리의 형태로 시각화한다.

• 자료 수집 모듈 : 그루비 라이브러리(Groovy Library)의 GroovyCodeVisitor를 이용하면 SmartApp의 AST를 쉽게 순회할 수 있다. 자료 수집 모듈은 1번 이상 SmartApp의 AST를 순회하여 시각화에 필요한 정보를 수집한다. 첫 번째 AST 순회에서는 page와 subscribe의 정보를 수집한다. 또한 dynamic page의 존재 여부도 확인한다. 만약 dynamic page가 존재한다면 다시 한 번 AST를 순회하여 dynamic page를 정의하고 있는 함수를 모두 찾아 해당 dynamic page의 정보를 수집한다.

• 시각화 모듈 : 자료 수집 모듈에서 수집한 정보를 바탕으로 SmartApp의 페이지 구조를 시각화 한다. page에 대한 정보를 바탕으로 트리를 구성하며 subscribe에 대한 정보를 참고하여 input과 연결된 subscribe의 이벤트 핸들러 함수와 디바이스 기능을 해당 input 노드의 자식 노드로 표현한다. 또한 dynamic page가 존재할 경우 dynamic page의 정보를 바탕으로 해당 dynamic page의 section, input을 차례대로 노드형태로 표현한다. 만약 section과 intput과 같은 입력요소가 조건문, 반복문, 함수 블록 안에서 정의되었다면 해당 구문들도 자식 노드로 표현된다.

5. 구현

5.1 구현결과

본 시스템에 SmartApp의 소스 코드를 등록하면 그림.9와 같이 SmartApp의 페이지 구조가 트리의 형태로 시각화된 것을 볼 수 있다.

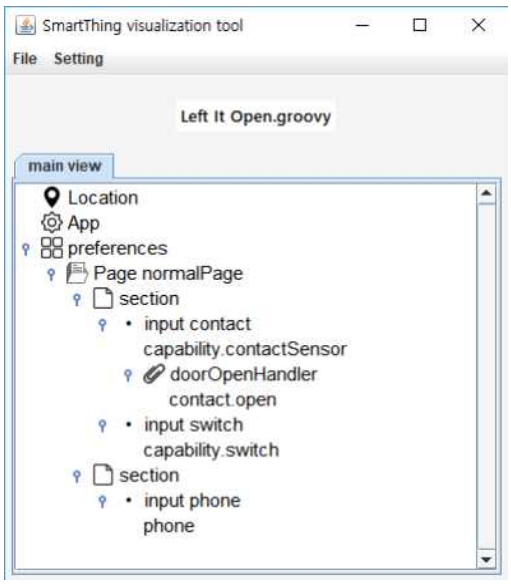

그림.9 시각화 도구 구현

5.2 시각화 도구 사용방법

상단의 파일(file)메뉴를 클릭한 후 열기(open)를 클릭하여 SmartApp의 소스 코드파일을 등록한다. 그러면 시각화 도구의 상단에 파일의 이름이 등록되고 시각화된 결과가 main view에 나타난다.

5.3 환경 설정 방법

상단의 설정(Setting)메뉴를 클릭하면 아래 그림.10과 같은 환경 설정 페이지가 나타난다. 환경 설정 페이지에서는 시각화 도구에서 표현하고자 하는 정보들을 설정할 수 있다.

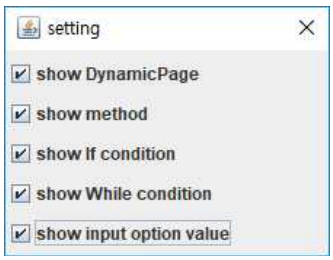

그림.10 환경 설정 페이지

6. 결론 및 향후 연구

이 논문에서는 SmartApp의 페이지 구조를 시각화 하는 시스템을 설계, 구현하였다. 특히 page와 subscribe등을 조사하여 등록된 디바이스와 이벤트 핸들러 함수들을 트리의 형태로 표현하였다.

조금 더 가치 있는 지원 시스템이 되기 위해서는 구조의 시각화뿐만 아니라 에러 탐지에 대한 추가적인 연구가 필요할 것이다. subscribe는 디바이스와 이벤트 핸들러 함수를 연결하는 중요한 기능을 수행한다. subscribe의 안정적인 실행을 위해서 아래와 같은 내용들이 검사되어야 한다.

1. subscribe함수의 input이 올바른 디바이스의 정보를 저장하고 있는지 검사한다.
2. subscribe의 등록된 이벤트가 연결된 디바이스에서 지원 가능한 기능인지 검사한다.
3. subscribe의 이벤트 핸들러 함수가 정의돼있는 함수인지 검사한다.

7. 참고문헌

[1]. Manas K. Saha, A Code Structure Visualization Tool for groovy, University of Houston, 12. 2013

[2]. Earlence Fernandes, Jaeyeon Jung, Atul Prakash, Security Analysis of Emerging Smart Home Applications, IEEE, 2016

[3]. SmartThings, SmartThings Developer Documentation, http://docs.smartthings.com, 07. 2017

[4]. The Apache Software Foundation, groovyConsole — the Groovy Swing console, http://groovy-lang.org/groovyconsole.html, 2003

[5]. 허순희, 창병 모, 정적분석을 이용한 자바 프로그램의 예외 전파 시각화, 정보과학회논문지, 2003

[6]. 김하영, 코드 클론을 효율적으로 관리하기 위한 시각화 방안, 동국대학교 학위논문, 2017