KIPS Trans. Softw. and Data Eng.
Vol.8, No.7 pp.303~310 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2019.8.7.303

A Mobile Service Architecture for Knowledge-Based Services in Mobile Environments

Jihoon Oh† · Jaeho Lee††

ABSTRACT

In the current mobile environment that is indispensable to our everyday lives, various forms of new business models are created including personalized services such as Google's "Google NOW" and Apple's "Siri". These services would not have been possible without technologies on the effective integration of various services and models. The requirements for effective integration of services include, 1) the efficient data sharing among multiple services, 2) the data-driven asynchronous execution of services, and 3) the simple extensible interaction method for the services. In this paper, we propose a mobile service architecture that utilizes the blackboard architecture to satisfy the aforementioned requirements to enable effective integration of various services, sharing and management of data between services, and asynchronous execution of services.

Keywords : Mobile Service Architecture, Knowledge-Based System, Mobile Environment

모바일 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 설계

오 지 훈† · 이 재 호††

요 약

우리의 생활과 밀접한 관계를 맺고 있는 모바일 환경은 다양한 사용자 요구사항과 맞물려 새로운 비즈니스 모델을 만들어 내고 있다. 이와 같은 시장 변화에 따라 구글의 Google NOW와 애플의 Siri와 같은 개인 비서와 같은 서비스가 탄생하였고, 우리나라에서도 사용자 모델을 기반으로 한 다양한 서비스를 통합하여 제공하기 위한 기술적 연구가 이루어지고 있다. 기술적 요구사항으로서 첫째, 여러 서비스 간의 데이터 공유가 가능해야 하고, 둘째, 서비스의 실행이 데이터 변화에 따른 비동기성을 지녀야 하며, 셋째, 서비스의 확장이 용이해야 한다는 점이다. 본 논문에서는 이러한 요구사항을 충족하기 위한 방안으로 블랙보드 아키텍처를 모바일 환경에 적용하여 지식기반 서비스제공을 위한 모바일 서비스 아키텍처로서 제시한다. 이를 통해 모바일 환경에서 다양한 서비스를 통합하고, 서비스 간의 데이터를 공유·관리하며, 비동기적으로 기능을 수행할 수 있도록 한다.

키워드 : 모바일 서비스 아키텍처, 지식기반 시스템, 모바일 환경

1. 서 론

21세기에 들어서 현대인은 다양한 직업이나 서비스를 생산하고 있으며, 이러한 사회 활동은 사람에게 주어진 제한된 시간을 효율적으로 활용할 필요성을 요구하고 있다.

1990년대부터 시작된 정보화의 흐름에 따라 사람들은 컴퓨터, 인터넷, 모바일 등의 새로운 기술을 쉽게 접하고 빠르게 적응하고 있으며, 특히 모바일과 스마트 등의 키워드로 대표되는 모바일 디바이스의 보급은 괄목할 만한 성장 속에 있다. 시장조사 전문기관 제니스는 2018년 전 세계 성인 스마트폰 보급률을 약 66.5%로 추산하고 있으며, 이는 세계에 걸쳐 약 35억 이상의 인구가 스마트폰을 활용하여 생활을 영위하고 있음을 의미한다[1].

이처럼 빠르게 우리의 생활에 필수적인 모바일 환경은, 앞서 언급한 다양한 사용자 요구사항과 맞물려 새로운 비즈니스 모델을 만들어 내고 있다. 이와 같은 추세는 구글, 애플과 같은 굴지의 선진 기업에서 주도적으로 이루어지고 있다.

구글은 Google NOW[2]를 통해 사용자들에게 필요로 하는

※ 이 논문은 2017년도 서울시립대학교 교내학술연구비에 의하여 지원되었음.
† 비 회 원 : (주) 라인플러스 소프트웨어 개발자
†† 종신회원 : 서울시립대학교 전자전기컴퓨터공학부 교수
Manuscript Received : January 3, 2019
First Revision : March 28, 2019
Accepted : May 24, 2019
* Corresponding Author : Jaeho Lee(jaeho@uos.ac.kr)

정보를 묻기도 전에 알아서 알려주는 서비스를 제공하고 있다. 사용자의 위치 정보, 일정 등의 데이터를 이용하여 해야 할 일, 가야 할 곳, 교통상황 등의 정신없는 일상을 관리할 수 있도록 도와주고, 좋아하는 스포츠팀의 실시간 경기 정보나 관심을 가질만한 뉴스 등을 언제 어디서나 간편하게 확인할 수 있도록 했다. 말 그대로 손안의 개인 비서와 같은 서비스를 제공함으로써 사용자에게 더욱 효율적인 시간 활용이 가능하도록 한 것이다.

애플은 음성인식 서비스인 Siri[3]를 통해 사용자들에게 편리하고 다양한 서비스를 제공하고 있다. 마이크를 통해 전달되는 사용자의 음성을 인식하여 다양한 질문에 응답하고 적절한 동작을 수행한다. 예컨대, 모바일 디바이스를 손으로 통제하지 않고도 캘린더 일정을 관리할 수도 있고 문자나 이메일도 전송할 수 있다. 차량 운전 중과 같이 손을 사용하기 힘든 상황에서도 몇 마디 말로 필요한 기능을 수행하거나 정보를 쉽게 얻을 수 있다는 것이다.

이와 같은 세계적인 흐름에 발맞춰 우리나라에서도 기술적 선점을 위한 시도가 이루어지고 있다. 2010년 지식경제부에서 '모바일 플랫폼 기반 계획 및 학습 인지 모델 프레임워크[4]'과제가 추진되어 스마트폰을 통해 사용자의 위치 정보, 일정, 애플리케이션 사용 패턴 등의 사용자 모델을 기반으로 한 다양한 서비스를 통합하여 제공하기 위한 기술을 개발한 바 있으며, 이를 통해 앞서 언급된 기술적 진보를 선도하기 위한 노력을 진행 중이다.

본 논문은 위 과제에 대한 시도의 하나로 연구되어온, 통합 프레임워크를 개발하는 것을 목표로 삼고 연구를 진행하였다. 그러나 이러한 기술적인 도전은 다음과 같은 요구사항을 가지고 있다.

첫 번째는 여러 서비스 간의 데이터 공유가 가능해야 한다. 각각의 서비스는 사용자의 여러 데이터를 혼합하여 모델을 자동학습하거나 실시간으로 사용자의 현재 상황을 인지하여 필요한 정보를 제공해야 한다. 예컨대, 사용자의 일정 정보를 기반으로 새로운 할 일 정보를 제공하기 위해서는 일정 데이터뿐만 아니라, 위치 정보, 가족 관계 등의 복합 데이터를 이용한다. 따라서 다양한 사용자 모델을 각 서비스가 서로 공유할 수 있어야 한다.

두 번째는 서비스의 실행이 데이터 변화에 따른 비동기성을 지녀야 한다는 점이다. 모바일 기기는 휴대성을 극대화 시킨 디바이스이다. 이는 곧 배터리, CPU, 메모리 등의 한정된 자원을 효율적으로 사용할 수 있어야 한다는 것이다. 사용자의 위치 정보와 같이 실시간으로 변화하는 사용자 모델을 인지하고 학습하는 작업은 많은 양의 자원을 소모하게 된다. 하지만 사용자의 일정 정보와 같이 정적인 정보를 활용하는 서비스는 데이터의 변화를 기반으로 작동하여 효율적인 자원 활용이 가능할 수 있다. 따라서 서비스의 실행의 비동기성을 고려하여 아키텍처를 설계해야 한다.

마지막으로 세 번째는 서비스의 확장이 용이해야 한다는 점이다. 모바일 애플리케이션은 매우 빠르게 변화하는 사용자의 요구사항에 따라 민첩하게 변화될 수 있어야 한다. 다시 말해, 언제든 새로운 서비스를 쉽고 빠르게 추가할 수 있어야 한다는 것이다. 그러기 위해서는 각 서비스를 모듈 단위로 개발하면서도 앞서 이야기한 두 가지 요구사항을 모두 만족할 수 있어야 한다. 따라서 지식기반 서비스제공을 위한 모바일 아키텍처는 반드시 서비스의 확장성을 고려해야 한다.

이와 같은 요구사항들은 모바일이라는 제한된 환경 내에서 활용하기 위한 일반적인 아키텍처가 필요함에도 불구하고, 현재까지 안드로이드나 iOS에서 요구사항에 완벽하게 부합하는 아키텍처는 존재하지 않았다.

따라서, 본 논문은 앞서 언급한 요구사항을 부합하는 아키텍처로, 블랙보드 아키텍처를 기반으로 모바일 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처를 제안한다. 이 아키텍처는 온톨로지를 기반으로 하는 시스템에 적용되어 통합 서비스를 제공하기 위한 방법을 모바일 환경에 적용한 아키텍처로서 데이터 공유가 자유롭고, 이벤트 기반 비동기성을 지니며, 확장이 용이한 특징을 통해 요구사항에 대응한다.

2. 관련 연구

이 논문에서 제시하는 아키텍처는 첫째로 다양한 서비스 간의 데이터를 공유하는 방법, 둘째로 서비스의 실행이 데이터 변화에 따른 비동기성을 지니는 방법, 셋째로 서비스의 확장성을 확보할 방법을 제공한다.

다음은 이러한 문제를 다루는 분야에 관한 관련 연구 내용이다.

2.1 모바일 환경 연구

우선적으로 모바일의 대표주자인 구글의 안드로이드[5]와 애플의 iOS[6] 환경에서 데이터 공유, 비동기성, 확장성 요구사항을 만족하기 위한 기반 기술에 대한 이해와 연구가 필요하다. 따라서 각 모바일 운영체제에서 제공하고 있는 기술에 대한 연구를 진행하였다.

1) 안드로이드 환경 연구

안드로이드 환경에서 애플리케이션은 Activity, Service, Broadcast Receiver, Content Provider 4가지의 컴포넌트로 구성된다. 이 중 Broadcast Receiver와 Content Provider는 애플리케이션 및 서비스 간의 데이터 공유를 지원하기 위한 컴포넌트이다.

먼저 Broadcast Receiver는 단말기 내에서 발생하는 이벤트 및 데이터를 수신하는 컴포넌트이다. 기본적으로 모바일 기기의 네트워크 변화, 배터리, 전화 수신 등의 이벤트를 수신하여 동작할 수 있도록 설계되어있고, Intent를 통해 데이터를 전달 받을 수 있다. 또한, 안드로이드 운영체제에서 미리 정의해놓은 이벤트 이외에도 애플리케이션 개발자가 직접 이벤트를 구현하여 다른 애플리케이션과 특정 이벤트에 따라 데이터를 주고받을 수 있다. 특히 Broadcast Receiver는 애플리케이션의 life cycle과 무관하게 동작한다는 특징을 지니고 있다.

다음으로 Content Provider는 애플리케이션 및 서비스 간의 데이터 공유를 지원하기 위한 컴포넌트로, 이를 이용하여 외부에 데이터베이스를 노출시키고 이를 이용하여 데이터를 공유할 수 있다. 외부에서는 Content Resolver 인터페이스를 이용하여 해당 애플리케이션의 URI를 통해 접근할 수 있다. 즉, Content Provider와 Content Resolver를 이용하여 데이터를 공유할 수 있다는 것이다.

하지만 안드로이드의 기본 컴포넌트인 Broadcast Receiver와 Content Resolver만으로는 요구사항을 모두 만족하지 못한다. 두 가지의 컴포넌트를 이용하여 비동기적으로 데이터를 공유할 수 있으나, 서비스의 확장성을 충족하지는 못하기 때문이다. 공유하고자 하는 서비스의 수에 따라 Content Resolver를 늘리거나, 각 서비스의 특정 Event를 모든 서비스가 등록해야 한다는 단점이 발생할 수밖에 없다. 이는 서비스 통합 과정에서의 개발 시간 증가로 이어질 수 있으며, 빠르게 변화하는 사용자의 요구사항에 민첩하게 반응할 수 없을 수 있다. 따라서 서비스의 확장성을 충족하기 위한 새로운 방안을 모색해야 한다.

2) iOS 환경 연구

iOS 환경에서는 애플리케이션마다 별도의 Sandbox를 생성하여 다른 애플리케이션과 데이터를 공유할 수 없도록 설계하였다. iOS의 Sandbox는 애플리케이션마다 별도의 파일을 생성하고 외부와는 공유되지 않도록 보안 체계를 갖추고 있다. 각 애플리케이션은 자체 Sandbox를 지니고 있으며 내부는 4개의 폴더로 구성되고 각 폴더는 사용자의 정보 및 애플리케이션 정보를 저장하는 역할을 한다. 즉, iOS 환경에서는 원천적으로 외부에서의 애플리케이션 데이터 접근을 차단한 셈이다.

원천적으로 데이터 공유가 불가능하지만, Custom URL Schema라는 방법을 통해 일부 데이터는 공유할 수 있다. 각 애플리케이션에서 등록한 URL schema를 통해 해당 애플리케이션을 호출하여 공유하고 있는 데이터를 얻을 수 있지만, UI switching이 발생하여 백그라운드 상태에서 작업할 수 없다는 점이 존재하여 서비스 실행의 비동기성 요구사항을 충족할 수 없다. 또한, URL을 통해 한정된 데이터만을 공유하기 때문에 서비스 간의 원활한 데이터 공유를 위한 요구사항을 충족하지 못한다.

따라서 iOS 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 설계는 원천적으로 불가능하다는 결론을 내릴 수 있다.

2.2 Mobilis Architecture

Thomas Springer는 다양한 모바일 환경에서의 통합 서비스제공을 위한 개념적 아키텍처인 'Mobilis'[7]를 제안하였다. Mobilis 아키텍처는 기능에 따라 Device OS layer, Basic Service layer, Mobilis Service layer, Application layer의 4가지 layer로 구성하였다.

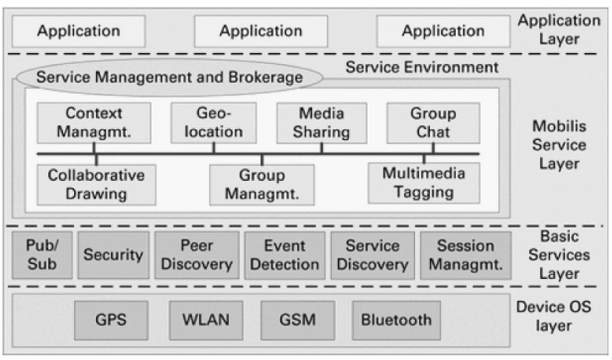

Fig. 1. Mobilis Architecture[7]

먼저 Device OS layer에서는 안드로이드, iOS, Windows Mobile, Symbian OS 등의 모바일 OS에서 GPS, WLAN 등의 하드웨어 기능을 제공하는 layer로 구성하였고, Basic Service layer는 각 OS의 기능에 따라 기본적으로 제공하는 서비스 layer로 구성하였다.

다양한 서비스를 통합하는 Mobilis Service layer에서는 모바일 내부가 아닌 외부에서 Context Management Service, Geo-location Service, Multimedia Tagging Service 등의 여러 서비스를 service bus로 연결하고 각 서비스 간의 기능을 공유할 수 있도록 구성하고 있다. 이를 통해 서비스 간의 데이터를 공유할 수 있고, service bus와 같은 인터페이스를 통해 서비스의 추가·확장 또한 용이하다는 장점이 있다.

하지만 Mobilis 아키텍처는 서비스 통합을 모바일 외부에서 구성하기 때문에 실시간적으로 변화하는 사용자의 모델을 활용한 비동기적 서비스 구현이 힘들다. 사용자의 위치정보와 같은 Basic Service layer 데이터를 공유하기 위해서는 네트워크를 통해 Mobilis Service layer와 통신하는 과정이 반드시 필요하고, 모델을 바탕으로 학습된 데이터를 다시 각 모바일 디바이스에 비동기적으로 전송하는 과정이 필요하다. 이는 곧 모바일 내부의 서비스 컴포넌트가 지속적으로 Mobilis Service layer와 통신해야 할 수밖에 없다는 단점을 지니게 되어 서비스 실행의 비동기성을 충족하지 못한다.

2.3 Open Data Kit (ODK)

Waylong Brunette는 안드로이드 환경에서 모바일 외부 센서들의 통신 채널을 하나의 인터페이스로 통합하고, 사용자 애플리케이션에서 쉽게 접근하여 사용할 수 있도록 Open Data Kit[8]를 제안하였다.

Open Data Kit는 여러 외부 센서의 설정 및 작동 등의 작업을 Service Interface로 통합하였다. 이는 안드로이드의 기본 컴포넌트 중 하나인 Broadcast Receiver를 통해 구현되어 애플리케이션에서 각 센서를 비동기적으로 제어할 수 있도록 하였다.

Service Interface로 통합된 다양한 센서 데이터는 안드로이드에서 제공하는 Content Provider를 이용하여 한 곳에 저장하고, 다양한 애플리케이션에서 접근할 수 있도록 허용하였다. 이를 통해 각 센서에서 모인 데이터를 쉽게 공유할 수 있다.

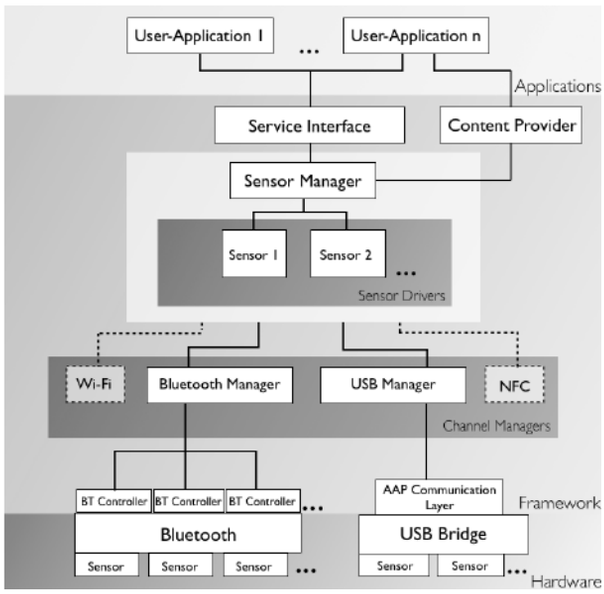

Fig. 2. Open Data Kit Architecture[8]

Waylong Brunette이 제안한 ODK 아키텍처는 Content Provider를 여러 센서의 공용저장소로 사용하여 다양한 애플리케이션에 센서 데이터를 공유할 수 있다. 또한, Broadcast Receiver로 구현된 Sensor Interface를 통하여 센서의 확장 또한 자유롭다는 장점이 있다.

하지만, ODK 아키텍처에서 각 센서의 실행은 애플리케이션의 life cycle에 의해 결정된다. 예컨대 사용자의 위치정보와 같이 실시간적인 데이터를 취득하기 위해서는 애플리케이션이 반드시 실행되어야 한다는 점이다. 이에 따라 ODK 아키텍처로는 데이터 변화에 따른 비동기적인 이벤트 추적이 불가능하다는 결론을 얻을 수 있고, 이는 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 설계 요구사항에 부합하지 않는다.

2.4 요구사항과의 비교 분석

모바일 운영체제의 대표주자인 안드로이드 및 iOS 환경과 Mobilis 아키텍처, 그리고 Open Data Kit를 지식기반 서비스 제공을 위한 모바일 서비스 아키텍처 요구사항을 종합적으로 분석하면 다음과 같다.

Table 1. Comparison of the Requirements

OS	Data Sharing	Asynchronous Operation	Extensibility
Android	○	○	X
iOS	X	X	X
Mobilis	○	X	○
ODK	○	X	○

먼저 안드로이드 환경에서는 Broadcast Receiver와 Content Provider를 이용하여 서비스 간의 데이터 공유와 데이터 변화에 따른 비동기적 서비스 실행이 가능하다. 하지만 다중 서비스를 위한 인터페이스의 부재로 인하여 서비스 추가에 대한 부담이 존재하여 요구사항을 만족하지 못한다.

반면 iOS 환경은 애플리케이션에 대한 강도 높은 보안 체계로 인하여 서비스 간의 데이터 공유가 원천적으로 불가능하며, Custom URL schema를 활용한 데이터 전달 방식은 UI switching 및 백그라운드 상태에서의 비동기적 구현이 불가능하여 요구사항 모두를 만족할 수 없다.

다음으로 Mobilis 아키텍처는 service bus를 이용한 서비스 통합이 가능하고, 서비스 간의 데이터 공유도 일부 가능하다. 하지만 각 서비스의 데이터를 복합적으로 사용하기 어려운 구조를 지니고 있으며, 모바일 기기 외부 환경에서 service layer를 구성하기 때문에 GPS와 같은 basic service layer의 서비스를 비동기적으로 실행할 수 없다는 단점을 지닌다. 따라서 Mobilis 아키텍처는 요구사항의 데이터 공유와 비동기성 요구사항을 만족하지 못한다.

마지막으로 Open Data Kit 아키텍처는 안드로이드 환경에서 제공하는 Content Provider를 이용하여 구현된 공용저장소가 존재하여 센서 서비스 간의 데이터 공유가 가능하다. 또한, Broadcast Receiver로 구현된 Sensor Interface를 통해 서비스의 확장성 요구사항을 부합하고 있다. 하지만 센서 서비스의 실행이 애플리케이션의 life cycle에 따라 결정된다는 단점을 지니고 있어 비동기성 요구사항을 만족하지 못한다.

따라서 모바일 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 요구사항을 모두 만족하기 위한 새로운 아키텍처에 대한 깊이 있는 연구가 필요하다.

3. 연구 내용

본 논문은 모바일 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 요구사항을 모두 충족하기 위한 접근으로 multi-agent 시스템에서 사용되는 Blackboard 아키텍처를 안드로이드 환경에 적용하여 해결 방안으로 제시한다.

Blackboard 아키텍처는 다양한 agent의 knowledge source (KS)를 Blackboard에 Post하여 다른 agent들과 KS를 공유하도록 구성된다. 각 agent는 Blackboard에서 관심 KS를 subscribe하고, Notify를 통해 갱신된 KS를 비동기적으로 취득할 수 있는 특징을 지니고 있다. 또한, 모든 agent는 Blackboard Interface를 통해 Blackboard에 접근하기 때문에 agent의 추가 또한 용이하다는 장점을 지닌다.

이와 같은 Blackboard 아키텍처의 특성을 모바일 환경에서 적용하여 요구사항을 만족하도록 제안한다.

3.1 모바일 환경에서의 Blackboard 아키텍처 설계

본 논문에서는 모바일 환경에서 Blackboard 아키텍처를 Fig. 5와 같이 설계하고, 단일 인터페이스인 Blackboard Interface를 이용하여 다중 서비스 간의 데이터를 공유하고 관리하도록 구성하였다.

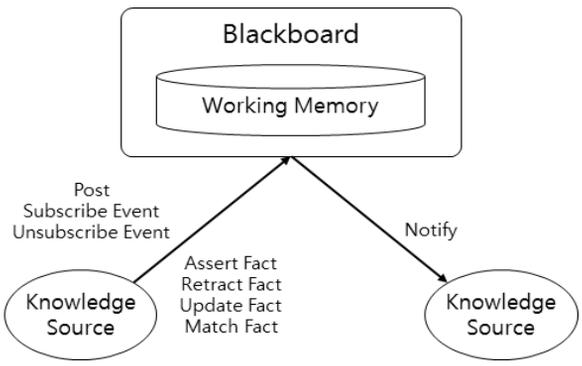

Fig. 3. Knowledge Management Framework[9]

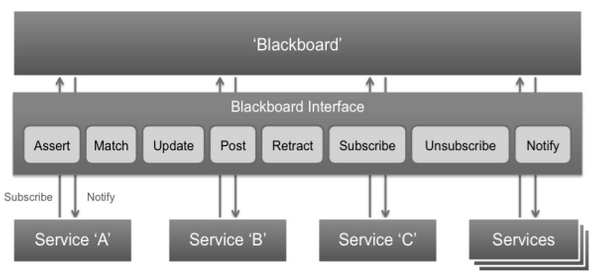

Fig. 4. Blackboard Architecture

Fig. 5와 같이 각 서비스는 Blackboard Interface를 통해 Blackboard에 접근할 수 있다. 공유하고자 하는 데이터를 인터페이스의 Assert, Retract, Update, Match, Post 등의 질의를 이용하여 등록·관리할 수 있도록 한다.

각 서비스는 Blackboard에 공유 데이터 중 관심 있는 데이터를 Subscribe하고, Blackboard는 공유 데이터의 변화에 따라 구독하고 있는 특정 서비스에게 데이터 변화를 Notify하여 비동기적인 실행이 가능하도록 한다.

사용자의 일정 충돌 가능성을 판단해주는 서비스로 예를 들자면 다음과 같이 작동할 것이다.

사용자의 일정 정보를 관리하는 Service'A'는 Blackboard에 사용자의 일정을 Post하여 다른 서비스와 데이터를 공유한다. 사용자의 일정에서 위치정보와 시간을 이용하여 일정 충돌 가능성이 있는지를 판단하는 Sevice'B'는 Service'A'의 사용자 일정 정보를 Subscribe한다. 이때 Service'B'는 초기값으로 Service'A'가 Post한 데이터를 취득하여 일정 간의 충돌 가능성을 판단하여 새로운 데이터를 생성한다.

이후에 사용자가 일정을 변경할 경우, Service'A'는 변경된 일정을 Blackboard에 Update하고, Blackboard는 일정 정보를 Subscribe하고 있는 Service'B'에게 데이터 변경을 Notify하여 다시 일정 충돌 가능성을 판단하도록 한다. 여기에서 중요한 점은 사용자 일정 정보가 변하기 전까지 Service'B'는 동작하지 않는다는 것이다. 따라서 사용자 일정 데이터 변화에 따른 비동기적인 이벤트 추적이 가능해진다.

Service'B'에서 추론한 일정 충돌 데이터 또한 Blackboard를 통해 공유함으로써 사용자의 필요에 따라 언제든지 관련 정보를 Activity에 전달할 수 있다.

3.2 Blackboard Interface

Blackboard는 공유 데이터 관리를 위한 Blackborad Interface를 제공한다. 다음은 인터페이스가 제공하는 메소드를 정리한 표이다.

Table 2. Blackboard Interface

Method Name	Description
Assert	Asset a new fact
Retract	Retract the fact
Update	Update the existing fact
Match	Match the fact and bound variables
Post	Post an event
Subscribe	Subscribe to an event
Unsubscribe	Unsubsribe to an event

각 서비스는 공유하고자 하는 context를 Blackboard Interface의 Post, Assert 질의를 통해 Blackboard에 등록할 수 있으며, 공유 context를 삭제하거나 변경할 경우, Retract, Update 질의를 통해 Blackboard에 등록된 context를 삭제·변경할 수 있다.

Blackboard에 등록된 관심 context를 구독하여 변경에 따른 추적을 할 경우, Subscribe 질의를 통해 Blackboard에 등록하여 context 변경을 Notify받을 수 있고, 언제든지 Unsubscribe를 통해 구독을 삭제할 수도 있다. context 구독을 통한 context 획득이 아닌 일시적으로 context를 얻기 위해서는 Match를 통해 가능하다.

Blackboard는 등록된 context에 따라 구독하고 있는 서비스에게 context가 변경될 경우 Notify를 하여 새로이 갱신된 context를 전달하는 기능을 수행한다.

이와 같은 질의들은 모두 Blackboard Interface를 통해 가능하며, 이를 통해 필요에 따라 언제든지 서비스를 쉽게 추가할 수 있다는 장점을 가진다. 더 나아가 context의 변화에 따라 갱신된 context를 얻을 수 있어 비동기적 이벤트 추적이 가능하다.

3.3 Blackboard 구현

Blackboard와 Blackboard Interface는 안드로이드 환경에서 기본적으로 제공하는 컴포넌트인 Content Provider와 Broadcast Receiver를 wrapping하여 구현되었다.

Blackboard는 Content Provider를 wrapping하여 각 서비스에서 공유하는 데이터를 저장하고, 데이터를 구독하고 있는 서비스의 정보를 함께 저장하여 데이터 변경에 따른 구독이 가능하도록 구현되었다.

Blackboard Interface는 Blackboard의 Content Provider에 접근하기 위한 Content Resolver를 Asset, Match, Update, Post, Retract, Subscribe, Unsubscribe 질의들과 함께 wrapping 하였으며, Blackboard에 등록된 관심 데이터의 변경에 따른 Broadcasts를 수신하도록 Broadcast Receiver를 이용하여 Blackboard의 Notify를 구현하였다.

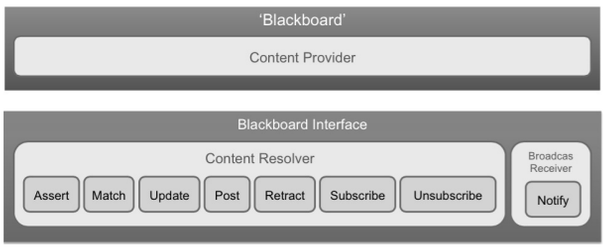

Fig. 5. Blackboard with the Blackboard Interface

3.4 요구사항과의 비교

Blackboard는 단일 공용저장소로서의 기능을 가지며 이를 통해 다양한 서비스들의 데이터를 한 곳에서 공유하고 관리할 수 있다. 각 서비스마다의 저장소를 공유하는 형태가 아닌 공용저장소를 이용하여 데이터 공유가 이루어지기 때문에 하나의 인터페이스만으로도 다양한 서비스의 데이터를 쉽게 공유할 수 있다는 장점이 생긴다. 이를 통해 서비스간의 데이터 공유가 가능해야 한다는 요구사항을 충족한다.

안드로이드의 Broadcast Receiver는 애플리케이션의 life cycle과 무관하게 특정 이벤트에 따라 비동기적으로 동작할 수 있다는 특징을 지니고 있다. Broadcast Receiver의 이러한 특성을 이용하여 Blackboard의 Notify를 구현하여 데이터 변화에 따른 이벤트 추적이 가능해진다. 따라서 각 서비스는 구독하고 있는 데이터 변화에 따라 비동기적으로 실행·동작할 수 있고 이는 서비스 실행의 비동기성 요구사항을 만족한다.

Blackboard는 Blackboard Interface를 이용하여 접근하도록 설계되었다. 각각의 서비스마다 Content Provider를 이용할 경우, 각각에 맞는 Content Resolver를 구현해야 하므로 서비스의 확장이 복잡해질 수 있다. 하지만 Blackboard는 하나의 Content Provider를 이용하여 데이터를 공유하고, 이에 대한 접근은 단일 인터페이스인 Blackboard Interface를 이용하므로 서비스의 확장성 요구사항을 만족한다.

4. 결과 검증

지식기반 서비스제공을 위한 모바일 서비스 아키텍처의 요구사항으로서 첫 번째, 여러 서비스 간의 데이터 공유가 가능해야 하고, 두 번째, 서비스의 실행이 데이터 변화에 따른 비동기성을 지녀야 하며, 세 번째, 서비스의 확장이 용이해야 한다는 것을 도출하였다.

본 논문에서 제시한 Blackboard 아키텍처가 세 가지 요구사항을 모두 만족하고 있는지를 검증하기 위하여 Blackboard 아키텍처를 이용하여 서비스를 통합·구현하고, 각 요구사항에 대한 적절한 검증방법을 통해 Blackboard 아키텍처의 유용성을 제시한다.

4.1 데이터 공유 검증

세 가지의 서비스를 구현하고, Blackboard를 이용하여 데이터를 공유함으로써 서비스 간의 데이터 공유 요구사항을 검증하였다.

사용자 일정 정보를 Blackboard에 Post하여 공유하는 Calendar Agent 서비스와 공유된 캘린더 데이터를 Subscribe하여 일정의 위치와 시간 정보를 이용한 일정 충돌을 가능성을 추론하는 Conflict Agent, 그리고 일정을 기반으로 새로운 할 일 정보를 제안하는 Todo Agent를 구현하였다. Conflict Agent는 일정 간 충돌을 비교하기 위하여 시간 및 위치 정보가 포함된 일정 정보를 구독하며, Todo Agent는 일정의 종류에 따른 추론을 수행하기 위하여 이름이 정의된 일정 정보를 구독한다.

Calendar Agent에서 사용자의 일정 정보를 Post 하고, Conflict Agent와 Todo Agent는 Blackboard에 등록된 일정 정보를 Subscribe하여 캘린더 데이터를 가져온다. Conflict Agent는 모든 일정 정보를 활용한 시공간 추론을 통해 충돌하는 일정을 확인하고 이를 표시한다. Todo Agent는 일정 종류에 따른 행위가 정의된 Todo Model을 활용한 추론을 통해 일정의 종류에 맞는 할 일 정보를 제안한다.

실행결과, Fig. 7과 같이 Calendar Agent에서 Blackboard를 통해 공유한 사용자 일정 정보가 Conflict Agent와 Todo Agent에게 올바르게 전달되어 두 가지 서비스 모두 정상적으로 작동함을 확인할 수 있다.

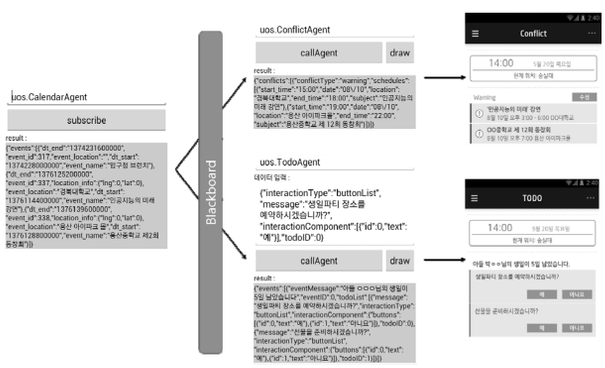

Fig. 6. Sharing of Data among Services using Blackboard

4.2 비동기성 검증

서비스 실행의 비동기성 검증은 다음과 같은 순서로 이루어진다.

Calendar Agent를 통해 사용자 일정 정보를 Blackboard에 Post한다.

Conflict Agent와 Todo Agent는 일정 정보를 Subscribe하여 Blackboard에 등록하고 서비스와 애플리케이션을 terminate한다.

임의로 복수의 일정을 변경하여 Calendar Agent를 통해 Blackboard에 공유한 일정 정보를 Update한다.

Blackboard에 공유된 일정 정보가 Update되어 Conflict Agent와 Todo Agent에 Notify를 보낸다.

Conflict Agent와 Todo Agent가 실행될 경우, “UOS Schedule Manager has been launched”라는 Toast 메시지를 띄우고 각 서비스를 수행한다.

이와 같은 순서로 서비스를 구동하여 실험한 결과, 5번의 Toast 메시지가 Fig. 8과 같이 정상적으로 출력되었다. Conflict

Agent는 추론을 통해 각 일정 간의 충돌을 확인하고, Alert가 발생함에 따라 알림을 통해 사용자에게 일정을 수정할 것을 제안하였으며, Todo Agent는 생일 일정을 위한 장소 예약 제안을 수행하였다. 이러한 일정 변경에 따른 각 서비스가 정상적으로 수행되었음을 확인할 수 있으며, 이를 통해 서비스 실행의 비동기성을 검증할 수 있다.

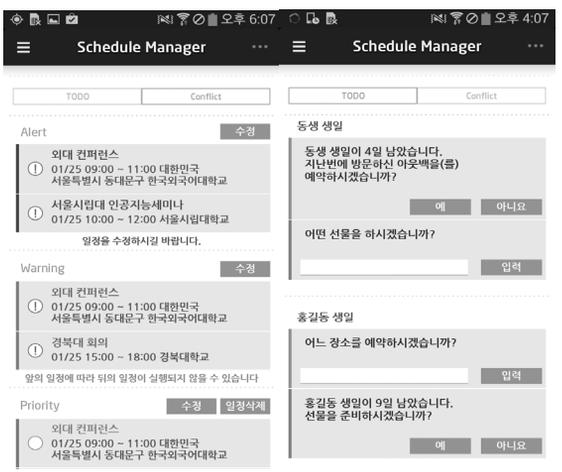

Fig. 7. Toast Messages

4.3 확장성 검증

서비스의 확장성 요구사항 검증은 기존의 Content Provider (CP)와 Content Resolver(CR)를 이용한 데이터 공유방법과 우리가 개발한 모바일 서비스 아키텍처를 이용한 공유 방법의 차이를 통해 이루어진다.

Content Provider를 이용하여 데이터를 공유할 경우, query, insert, update, delete, getType, onCreate 메소드를 필수적으로 Override하여 구현해야 한다. 하지만 모바일 서비스 아키텍처를 이용할 경우, Override해야 하는 메소드 없이 Blackboard에 새로운 서비스를 위한 field를 생성하는 것만으로도 데이터를 공유하고 관리할 수 있다.

데이터를 취득하는 방법에서도 차이가 발생한다. CP를 이용한 공유 데이터에 접근하기 위해서는 각 CP에 맞는 CR를 모두 구현해야 한다. 즉, n개의 서비스가 서로 연결하기 위해서는 n개의 CP와 n개의 CR을 구현하여야 하므로 n:n결합으로 구성된다. 하지만 모바일 서비스 아키텍처에서는 Blackborad Interface의 onNotify 메소드를 Override하고 이를 구현하는 것을 통해 손쉽게 다른 서비스와의 연결을 설정할 수 있다. 이를 통해 Blackborad와 각 서비스 간의 연결은 1:n 결합으로 구성할 수 있다.

CP와 CR을 이용한 n:n 결합은 새로운 서비스를 추가하기 위해선 기존의 모든 서비스와 개별적인 연결이 필요하므로 서비스의 확장성이 부족하다고 판단할 수 있다. 하지만 모바일 서비스 아키텍처를 이용할 경우, 새로운 서비스를 추가하더라도 Interface의 구현을 통해 손쉽게 Blackboard와 1:n 결합을 생성할 수 있으며, 이를 통해 서비스 확장성 요구사항을 만족하는 검증 결과를 나타낸다고 할 수 있다.

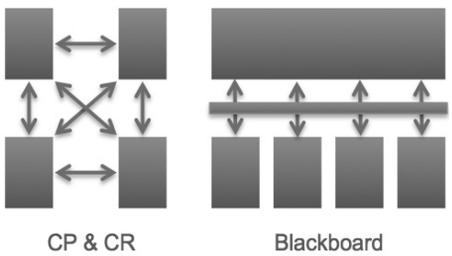

Fig. 8. CP & CR vs. Blackboard

5. 결 론

본 논문에서는 모바일 환경에서의 지식기반 서비스제공을 위한 모바일 서비스 아키텍처 설계의 방향으로 서비스 간의 데이터 공유가 가능하고, 비동기적인 서비스 제어가 가능하며, 서비스의 확장이 자유로운 Blackboard 아키텍처를 제안하였다.

이를 통해 다음과 같은 강점을 얻을 수 있다.

Blackboard를 이용하여 서비스들의 데이터를 효과적으로 공유하고 관리할 수 있다.

데이터 변화 이벤트를 기반으로 서비스 실행을 비동기적으로 제어할 수 있고, 이에 대한 효과로 모바일의 한정된 배터리와 메모리 절감 효과를 기대할 수 있다.

하나의 Blackboard Interface를 이용함으로써 통합 서비스 개발 비용 절감 효과를 기대할 수 있다.

따라서 모바일 환경에서 다중 서비스를 기반으로 하는 아키텍처를 고려할 경우, 본 논문에서 제안한 Blackboard 아키텍처를 적용하여 더욱 효율적이고 유연한 개발이 가능할 것으로 기대된다.

하지만 현재의 연구결과는 몇 가지 문제점을 지니고 있다. Blackboard 구현에 사용된 Content Provider의 보안성 문제가 지속적으로 제기되고 있다는 점[10]이다. 각 서비스의 정보를 통합하고 관리하기 때문에 Blackboard는 높은 보안 레벨을 지녀야 한다. 하지만 Content Provider 자체의 보안성 문제가 대두되는 만큼 다른 방향으로의 공용저장소 구현에 대해 고려하거나, 인터페이스에서의 보안 강화 방법에 관한 연구가 필요하다.

따라서 향후의 연구는 이러한 보안성 문제를 해결하기위한 방법을 고안해야 할 것이다.

References

[1] 이초희, “글로벌 스마트폰 보급률 연내 30% 돌파,” 아시아 경제, [Internet], http://www.asiae.co.kr/news/view.htm?idxno=2017101707023014904.

[2] Google, “Google NOW,” [Internet], http://www.google.com/intl/ko/landing/now.

[3] Apple, “Siri,” [Internet], http://www.apple.com/kr/ios/siri/?cid=wwa-kr-kwg-features-com.

[4] 지식경제부, 2010년도 산업원천기술개발사업 신규지원 대상과제 공고. 서울: 지식경제부, 지식경제부공고 제2010 - 35호, 2010.
[5] Google, “Android” [Internet], http://developer.android.com/guide/components.
[6] Apple, “iOS,” [Internet], https://developer.apple.com.
[7] T. Springer, et al., “A Flexible Architecture for Mobile Collaboration Services,” *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*, ACM, 2008.
[8] W. Brunette, et al., “Open Data kit Sensors: a Sensor Integration Framework for Android at the Application-level,” *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM, 2012.
[9] J. Lee, “A Data-Centered Integration Framework for Intelligent Service Robots. Robot and Human interactive Communication, RO-MAN,” *The 16th IEEE International Symposium on*, IEEE, 2007.
[10] E. Chin, et al., “Analyzing Inter-application Communication in Android,” *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ACM, 2011.

오 지 훈
https://orcid.org/0000-0002-8512-8841
e-mail : cloudysky32@gmail.com
2013년 서울시립대학교 전자전기컴퓨터공학부(학사)
2015년 서울시립대학교 전자전기컴퓨터공학과(석사)
2015년~2017년 (주) 매버릭 소프트웨어 개발자
2018년~현 재 (주) 라인플러스 소프트웨어 개발자
관심분야 : 인공지능, 지능형 시스템

이 재 호
https://orcid.org/0000-0002-3332-3207
e-mail : jaeho@uos.ac.kr
1985년 서울대학교 계산통계학과(학사)
1987년 서울대학교 계산통계학과(석사)
1997년 University of Michigan(박사)
1998년~현 재 서울시립대학교 전자전기컴퓨터공학부 교수
관심분야 : 인공지능, 지능 로봇