http://dx.doi.org/10.3745/KTSDE.2013.2.9.621

A Component-Based Application Framework for Context-Aware Smartphone Applications Based on Android

Seyoung Hwang† · Hyunguk Lee† · Sangwon Park††

ABSTRACT

This paper presents a framework for context-aware smart phone applications based on Android. The mobile context-aware system is composed of a low level context collection module, a high level context generation module and a service provision module. Existing android system cannot provide an appropriate application framework to integrate these independent modules. In this paper, we provide an application framework which make each module a component, and provide appropriate services to each component. This framework hides the Android platform, so that the complexity for organical combination can be minimized and the application developers can make the mobile context-aware applications easily.

Keywords : Mobile Application, Application Framework, Context Awareness, Sensing and Application, Virtual Assistant

안드로이드에서 상황 인지 스마트폰 애플리케이션을 위한 컴포넌트 기반 애플리케이션 프레임워크

황 세 영† · 이 형 욱† · 박 상 원††

요 약

본 논문은 안드로이드 기반 상황 인지 스마트폰 애플리케이션들의 유기적인 결합을 위한 프레임워크에 관한 것이다. 스마트폰을 이용한 상황인지 시스템에는 콘텍스트 수집, 수집된 콘텍스트를 해석 및 추론한 하이레벨 콘텍스트, 그리고 사용자에게 직접적으로 서비스를 제공해주는 시스템 등이 있다. 기존의 안드로이드는 이러한 여러 독립된 모듈들을 통합하여 유기적으로 결합시킬 적절한 방법을 제시하지 못한다. 따라서 본 논문에서는 각 시스템을 컴포넌트화 하고, 각 컴포넌트가 필요한 서비스를 제공하며, 안드로이드 플랫폼을 감추는 방식으로 유기적인 결합을 위한 복잡도를 최소화시켜 개발자에게 편의를 제공하는 프레임워크를 제안한다.

키워드 : 모바일 애플리케이션, 애플리케이션 프레임워크, 상황 인지, 센싱 및 응용, 가상 개인 비서

1. 서 론

사용자의 작업과 관련 있는 적절한 정보 혹은 서비스를 제공하기 위해 사용자의 현재 상황에 대한 콘텍스트 정보가 사용되는 경우 이를 상황인지 시스템이라 한다[1]. 현재까지 착용형 컴퓨터를 활용한 웨어러블 컴퓨팅(Wearable Computing)[2]과 사용자 컴퓨터의 사용로그를 활용한 라이프 로그(Life Log)[3]와 같은 상황인지 시스템이 연구되었다. 하지만 웨어러블 컴퓨팅은 비용과 복잡도 증가의 문제가 있고, 라이프 로그는 고정된 위치로 인한 네트워크의 한계가 있다. 반면 스마트폰을 상황인지에 사용할 경우 높은 처리량과 다양한 센서를 탑재하여 수집과 처리를 동시에 수행할 수 있는 환경이 마련된다[4,5]. 동시에 웨어러블 컴퓨팅의 높은 비용 문제와 라이프 로그의 네트워크 한계 문제도 해결 가능하게 되어 스마트폰에 탑재된 다양한 센서를 이용한 상황인지 연구가 진행되고 있다[4].

상황인지 시스템에서 사용자에게 적절한 정보를 제공하기 위해서는 개체의 상황을 특징짓기 위해 사용될 수 있는 정보를 의미하는 콘텍스트 정보가 필요하다[6]. 스마트폰을 통한 상황인지 서비스에는 GPS, 조도 센서와 같은 물리적으로 존재하는 센서에 의해 측정되는 사용자의 위치와 같은 물리적 콘텍스트와, 하드웨어에 의존하지 않고 정보를 제공하는 센서에 의해 측정된 사용자의 일정, 애플리케이션 사용목록과 같은 소프트 콘텍스트가 필요하다[7]. 또한 물리적

※ 본 논문은 지식경제부 산업기술원천개발사업(10035348, 모바일 플랫폼 기반 계획 및 학습 인지 모델 프레임워크 기술 개발)의 지원으로 수행되었음.
† 준 회 원 : 한국외국어대학교 정보통신공학과 학사과정
†† 종신회원 : 한국외국어대학교 정보통신공학과 교수
논문접수 : 2013년 7월 16일
수 정 일 : 1차 2013년 8월 13일
심사완료 : 2013년 8월 14일
* Corresponding Author : Sangwon Park(swpark@hufs.ac.kr)

콘텍스트와 소프트 콘텍스트를 해석 및 추론하여 가공된 하이레벨 콘텍스트와 사용자에게 서비스를 제공해주는 시스템이 필요하다. 이러한 시스템들은 각각 다른 개발자에 의해 개발될 수 있고 필요에 따라 계속해서 추가 및 삭제될 수 있다. 따라서 시스템들이 독립적으로 하나의 모듈을 이루면서도 상황인지 서비스를 제공하기 위해 유기적으로 동작하려면 복잡도가 증가하게 된다. 하지만 안드로이드 플랫폼만을 이용해서 서로 연관되면서도 독립된 모듈로 존재하기 위해서는 복잡한 연결과정이 필요하고 디버깅이 어렵다. 따라서 독립적인 시스템을 통합해줄 프레임워크가 필요하다.

이에 대한 해결책으로 본 논문에서는 MIF(Mobile Intelligent Application Framework)를 설계 및 구현하였다. MIF는 상황인지 시스템의 각 요소를 컴포넌트[8]화 하여 유기적으로 동작할 수 있도록 해주며, 상황인지에 필요한 각 컴포넌트가 요구하는 서비스를 제공한다. 또한 MIF를 사용하면 안드로이드 플랫폼을 감출 수 있으므로 오류를 최소화할 수 있으며, 코드가 간결화되어 개발자에게 편의를 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 상황인지 대한 기존 연구와 현재의 연구에 대해 설명하고, 3장에서는 MIF를 사용하여 상황인지 서비스를 제공하는 '학습인지 모델 프레임워크'에 대해 설명한다. 4장에서는 MIF에 대해 자세히 기술한다. 5장에서는 MIF 구현을 위해 사용한 전략과 MIF를 이용한 예제 화면을 보이고 6장에서 결론을 맺는다.

2. 관련 연구

유비쿼터스 컴퓨팅의 확산과 함께 기술이 발전하면서 착용형 컴퓨팅에 관한 연구가 진행 중에 있다[2]. 착용형 컴퓨팅은 센서로부터 개인 정보를 수집하고 처리 및 분석함으로써 사용자의 콘텍스트에 따른 서비스를 제공할 수 있다[9].

사용자의 개인 정보를 수집하는 다른 예에는 라이프로그가 있다. 라이프로그[3]는 한 개인의 일상에서 일어난 일을 다양한 장비 및 센서를 통해 수집되는 모든 정보를 뜻하며 개인의 사진, 음성, 문서, 영상 등과 같은 다양한 형태의 데이터를 포함하는 개념이다.

라이프로그의 예로는 Mcrosoft에서 진행한 MyLifeBits [10] 와 Stuff I've Seen(SIS)[11]을 들 수 있다. MyLifeBits에서는 개인화된 정보를 디지털화하여 저장하는 시스템을 구축함으로서 필요한 정보를 쉽게 검색하고 브라우징 하는 것을 목적으로 하고 있다[12]. SIS는 데스크탑 애플리케이션 등에서 발생하는 개인 정보를 기록 및 재사용을 목표로 개발되었으며 기록된 정보를 바탕으로 인덱스를 제공한다[12]. 하지만 MyLifeBits와 SIS는 주로 데스크탑에서의 개인 문서, 사진, 영상, 이메일 등에 대한 제한된 정보를 수집하기 때문에 데스크탑을 사용하지 않는 시간에는 사용자의 개인 정보를 수집할 수 없으며 이러한 제한된 정보를 바탕으로 인덱싱과 브라우징을 하므로 사용자의 요구사항을 충족시키지 못하는 한계점을 지닌다[12]. 또한 데스크탑의 고정적인 위치로 인해 이동중에 사용자의 개인 정보를 수집할 수 없기 때문에 한 개인의 일상에서 일어난 모든 정보를 포함하지 못한다.

높은 처리량과 다양한 센서를 탑재한 스마트폰이 등장하면서 센서 데이터의 수집과 처리를 동시에 수행할 수 있는 환경이 제공되어 이를 바탕으로 한 상황인지 연구들이 활발히 진행이 되고 있다[4,5].

스마트폰에는 GPS, 조도 센서, 가속도 센서 등과 같은 피지컬 센서가 있으며 통화목록, 캘린더 일정, 애플리케이션 사용목록, 애플리케이션 사용시간 등 사용자의 선호도를 파악할 수 있는 로지컬 센서를 만들 수 있다[13,14]. 스마트폰의 피지컬 센서 중 가속도 센서 및 자이로 센서를 이용하여 실제 환경에서의 걷기, 뛰기, 정지, 버스 탑승 중, 지하철 탑승 중과 같은 행위를 인지할 수 있다[5]. 또한 스마트폰의 로지컬 센서를 이용하여 사용자가 특정 시간에 자주 사용하는 애플리케이션을 추천해줄 수도 있다[14].

더 나아가 스마트폰의 다양한 센서를 통해 수집된 데이터를 바탕으로 사용자의 방문지 및 이동경로를 학습하고 실시간으로 추론하여 사용자가 방문하려는 장소 및 이동할 경로를 실시간으로 예측할 수 있다[13].

스마트폰 상황인지 서비스의 대표적인 예에는 Google Now와 Siri가 있다. Google Now는 안드로이드 젤리빈 OS부터 탑재된 구글의 상황인지 서비스로 사용자가 요구하지 않아도 시간 장소에 따른 적합한 서비스를 제공하며 Siri는 애플에서 발표한 음성인식 서비스로서 사용자의 콘텍스트를 인식하여 사용자의 개인 비서 역할을 수행한다[15,16]. 하지만 사용자가Google Now와 Siri를 이용하는 애플리케이션을 작성할 수 없기 때문에 이에 대한 플랫폼을 제공해야 한다.

따라서 스마트폰 센서를 이용하여 수집된 사용자 정보를 해석, 추론 학습하여 얻어진 선호도를 바탕으로 시간, 장소, 행동에 따른 의미 있는 서비스를 제공해야하는 필요성이 있다. 하지만 이러한 서비스를 제공하기 위해 다수의 독립적인 상황인지 애플리케이션을 유기적으로 동시에 진행되도록 하는 것과 오류가 발생하여 수정하더라도 다른 애플리케이션에 영향을 미치지 않도록 해야 하는 메카니즘이 필요하다.

3. 학습인지 모델 프레임워크

사용자의 작업과 관련 있는 적절한 정보 혹은 서비스를 제공하기 위해 사용자의 현재 상황에 대한 콘텍스트 정보가 사용되는 경우 이를 상황인지 시스템이라 한다[1]. 학습인지 모델 프레임워크는 상황인지 시스템 중의 하나로 사용자의 스마트폰으로부터 물리적, 소프트 콘텍스트를 수집하고 이를 학습 및 추론한 데이터를 기반으로 사용자에게 적합한 추천 서비스를 제공해주는 프레임워크이다.

학습인지 모델 프레임워크[1]의 구성요소로는 모바일 센서

1) 2010년 3월 1일부터 진행 중인 지식경제부의 '모바일 플랫폼 기반 계획 및 학습 인지 모델 프레임워크 기술 개발' 프로젝트를 통해 제작중인 상황인지 시스템

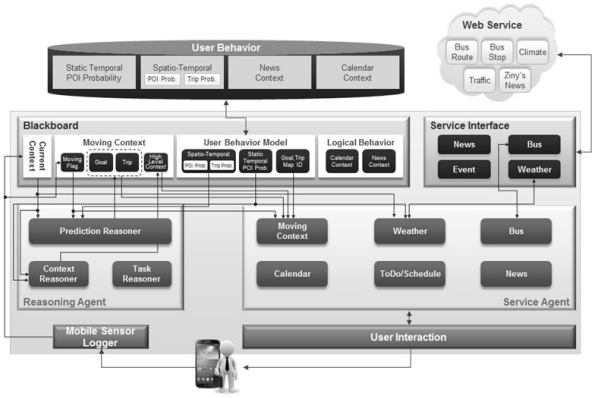


Fig. 1. Learning cognition model framework architecture

로거, 추론 에이전트, 서비스 에이전트, 상황인지 애플리케이션이 있다. 모바일 센서 로거는 사용자의 위치에 해당하는 물리적 콘텍스트와 애플리케이션과 같은 소프트 콘텍스트를 기록하고, 위치, 중력 등의 센서를 사용한 사용자 행동 추론 정보를 기록한다. 학습인지 모델 프레임워크에서는 행위 인지 기술[5]과 Logical Logger[14]를 결합한 Intergration Logger가 사용된다. 추론 에이전트로는 사용자의 이동경로 학습 및 실시간 예측 기법[13] 등이 사용되며 Integration Logger를 통해 수집된 콘텍스트 정보를 기반으로 해석 및 추론하여 하이레벨 콘텍스트를 생산하며, 이를 기계 학습하여 사용자의 관심지점, 경로 추론 정보 등을 얻는다. 서비스 에이전트는 상황인지 애플리케이션의 요청에 따라 수집된 콘텍스트 및 추론 정보, 웹 서비스 등을 활용하여 적절한 정보를 제공한다. 상황인지 애플리케이션은 서비스 에이전트에게 정보를 요청한 후 받은 정보를 GUI를 통해 사용자에게 서비스를 제공하는 역할을 한다.

이러한 학습인지 모델 프레임워크의 센서 로거, 추론 에이전트, 서비스 에이전트, 상황인지 애플리케이션 들은 각자 독립된 별도의 시스템이지만 하나의 시스템과 같이 유기적으로 동작하여 사용자에게 서비스를 제공해야 한다. 따라서 시스템을 통합 관리할 프레임워크가 필요하며 이를 4장에서 자세히 설명한다.

4. Mobile Intelligent Application Framework[2)]

4장에서는 MIF의 필요성에 대한 설명을 4.1절에, MIF 프레임워크를 사용하여 간결해진 시스템 개발의 예제를 4.2절에서 설명한다.

4.1 MIF

컴포넌트란 독립적으로도 완전한 하나의 시스템이지만 더 큰 시스템에서 '일부분'으로 사용될 수 있는 시스템을 말한다. 3장에서 설명한 모바일 센서 로거는 수집한 콘텍스트 정보를 다른 컴포넌트에 전달할 수 있어야 하고, 추론 에이전트는 모바일 센서 로거에서 수집한 콘텍스트 정보를 얻을 수 있어야 하며, 다른 컴포넌트에 해석 및 추론한 결과를 전달할 방법이 있어야 한다. 또 서비스 에이전트는 요청을 처리하기 위해 필요한 콘텍스트 및 해석, 추론한 결과를 얻을 수 있어야 하고 웹에 접근해서 데이터를 얻을 수 있어야 한다. 요청에 의해 생성한 정보를 상황인지 애플리케이션에 전달할 방법도 있어야 한다. 마지막으로 상황인지 애플리케이션은 서비스 에이전트에게 서로 나누어진 하나의 주제에 대한 화면 구성방식인 '카드' 생성을 요청하고 결과를 받을 수 있어야 한다.

이러한 컴포넌트들은 여러 팀에서 여러 컴포넌트들을 제작할 수 있다. 각 컴포넌트들은 위에서 설명한 바와 같이 상호 연관성이 있으나 디버깅 및 실행 등은 각자 별도의 프로세스에서 독립적으로 이루어져야 한다. 또한 상호 연관성을 위해 독립적으로 동작하면서도 각 컴포넌트 간에 동기적, 비동기적 호출이 가능해야 한다. 안드로이드는 컴포넌트 기반으로 설계되어 있다. 동기적 호출을 위해 안드로이드에서 제공하는 서비스, 비동기적 호출을 위해 인텐트 방송(intent broadcast)을 사용할 수 있다. 하지만 이것을 직접 이용할 경우 애플리케이션 코드가 매우 복잡해진다. 예를 들어 동기적 호출을 위해 서비스를 사용할 경우 모든 컴포넌트들은 다른 컴포넌트에 접근하기 위해 바인딩 과정을 거쳐야 하고, 다른 컴포넌트에서 자신에게 연결할 수 있도록 연결점을 만들어야 하며, 그 연결점에 해당하는 파일 또한 별도로 전달해 주어야 한다. 또, 비동기적 호출을 위해 인텐트 방송을 사용할 경우 컴포넌트 제작자들은 각각의 모든 컴포넌트들에서 정해둔 문자열을 알고 있어야 한다.

따라서 안드로이드 플랫폼을 사용자에게 감추면서 컴포넌트들이 각자 자신들의 요구사항을 충족하여 독립적으로 구성되어 있지만 하나의 큰 시스템과 같이 유기적으로 결합될 수 있도록 모든 컴포넌트를 통합해주는 프레임워크가 필요하다. 본 논문에서 제시하는 컴포넌트 통합 프레임워크인 MIF의 아키텍처는 Fig. 2와 같다.

Fig. 2에서 본 논문을 통해 제공되는 것은 Blackboard와 MIF이다. MIF에서 에이전트란 하나의 컴포넌트로 제작될

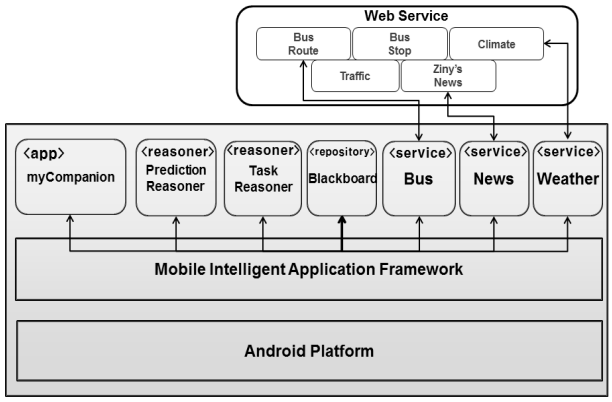


Fig. 2. MIF architecture

2) http://dislab.hufs.ac.kr/lab/mif에서 블랙보드 애플리케이션과 라이브러리 다운로드 및 라이브러리 사용 예제와 라이브러리의 형태로 제공된 클래스 및 함수에 대한 설명이 있다.

수 있는 독립된 시스템을 의미한다. 예를 들어 버스 정보에 관한 요청에 대해 결과를 보내는 Bus Agent, 날씨 정보에 관한 요청에 대해 결과를 보내는 Weather Agent 등을 제작할 수 있고, 사용자의 경로 추론 정보를 제공하는 Prediction Reasoner도 제작 가능한 하나의 에이전트이다. 블랙보드란 각 컴포넌트간에 데이터 공유를 위해 MIF에서 제공하는 공유 저장소이다. 상황인지 애플리케이션은 Bus Agent, Weather Agent와 같은 서비스 에이전트에게 요청을 보내 받은 결과를 사용자에게 GUI의 형태로 보여준다. 이 때 GUI는 카드의 형태로, 카드는 서로 나누어진 하나의 주제에 대한 화면 구성방식이다.

에이전트는 MIF에서 라이브러리의 형태로 제공된 함수를 사용하여 블랙보드에 접근할 수 있다. 상황인지 애플리케이션 역시 MIF에서 제공된 함수를 통해 서비스 에이전트에게 요청을 보내고 그 결과를 받을 수 있다. 따라서 에이전트들은 안드로이드 플랫폼에 신경쓰지 않고 MIF를 통해 시스템에 필요한 데이터를 얻을 수 있다. 이에 대한 자세한 접근 및 연결 방식은 5장에서 설명한다. 4.2절에서는 각 에이전트에서의 MIF 사용 예제를 보인다.

4.2 MIF 사용 예제

에이전트를 생성하는 예제는 Fig. 3과 같다.

에이전트를 만들기 위해서는 MIF의 Agent 클래스를 상속받아 새로운 에이전트를 만들고 미구현되어 있는 함수를 구현한다. 에이전트는 추론 에이전트가 될 수도 있고, 서비스 에이전트가 될 수도 있다. 에이전트에서는 subscribe함수를 통해 원하는 값에 대한 블랙보드 데이터 변경 리스너를 등록할 수 있다. 에이전트 개발자가 등록해 둔 리스너에 대해 블랙보드에서 값이 변경될 경우 onNotify를 통해 변경된 값에 대한 정보를 자동으로 얻을 수 있다. onNotify는 주로 변경된 값에 대해 새로운 해석 및 추론을 진행해야 하는 추론 에이전트에서 사용된다. onRequestCard는 상황인지 애플리케이션으로부터 정보 생성을 요청받을 경우 이에 대한 처리를 해주는 함수이다. 따라서 서비스 에이전트에서 주로 사용하는 함수이며, 위의 예제와 같이 Weather Agent는 onRequestCard로 받은 '5시간 뒤의 날씨 카드 생성 요청'에 대해 자료를 모아 리턴한다.

```
public class WeatherAgent extends Agent {
  @Override
  public void onNotify(String objName, String dataName, String oldValue, String
  newValue, String operation) {
    if(oldValue != Value from WeatherAgent) {
      processing for newValue;
    }
  }

  @Override
  protected void onRequestCard(String data) {
    If (data == request for weather information 5 hours from now) {
      get position resoning data 5 hours from now from the Blackboard;
      get weather data 5 hours from now from web services;
      return the result from Library methods;
    }
  }
}
```

Fig. 3. Example of Agent

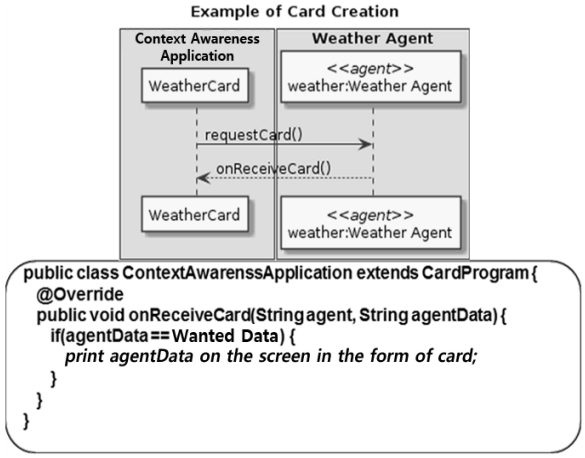


Fig. 4. Example of Card Creation

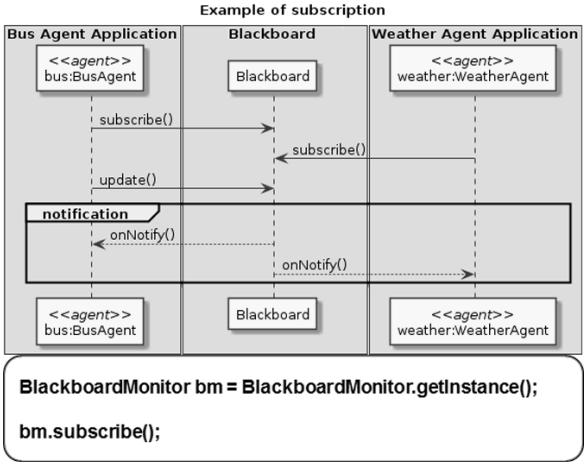


Fig. 5. Example of subscription

다음으로 상황인지 애플리케이션 입장에서의 카드 생성 요청에 대한 예제는 Fig. 4와 같다.

CardProgram을 상속받아 만든 상황인지 애플리케이션 객체는 requestCard 함수를 통해 프로세스 범위를 넘어서서 원하는 서비스 에이전트에게 원하는 데이터에 대한 카드 생성을 요청할 수 있다. 만들어진 카드는 onReceiveCard 함수에게 리턴되며 받은 정보가 원하는 정보라면 사용자에게 GUI의 형태로 제공해줄 수 있다.

마지막으로 블랙보드에 subscribe 함수를 통해 원하는 값에 대한 블랙보드 데이터 변경 리스너를 등록하는 예제는 Fig. 5와 같다.

MIF를 사용하는 프로그램은 예제 코드와 같이 싱글톤 형태인 BlackboardMonitor 객체를 얻을 수 있다. 이 객체의 subscribe 함수를 통해 블랙보드에 값 변경 리스너를 등록할 수 있다. subscribe 함수에는 인자로 어떠한 값 변경에 대해 알림을 받고 싶은지에 대한 문자열과, 값 변경 알림에 대한 onNotify 함수 처리를 할 에이전트의 이름이 들어간다. Fig. 5의 시퀀스 다이어그램에서 Bus Agent Application과 Blackboard Application 및 Weather Agent Application은

모두 독립된 프로세스 범위에 속하는 독립된 시스템이다. Bus Agent와 Weather Agent는 각각 블랙보드에 subscribe를 통해 리스너를 등록하고, Bus Agent에서 update를 통해 블랙보드의 값을 수정한다. 블랙보드는 리스너를 등록한 두 에이전트 모두에게 값 변경에 대한 알림을 보내게 된다.

4.2에서 보인 바와 같이 개발자는 간결한 코드를 사용하여 안드로이드 플랫폼에 접근하지 않고도 에이전트간 혹은 에이전트와 블랙보드간에 정보를 주고받을 수 있다. 위의 사용 예제에서 카드 생성 예제와 리스너 등록 예제는 비동기적 호출 방식을 사용하며, 에이전트 예제에서 Weather Agent는 동기적 호출 방식으로 블랙보드에 접근하여 정보를 모아 리턴하게 된다. 동기적, 비동기적 호출 방식에 대해서는 5장에서 자세하게 설명한다.

5. MIF 구현

5장에서는 MIF에서 요구사항을 만족하기 위해 취한 전략에 대해 설명하며, 마지막으로 5.7절에서 MIF를 이용해 제작된 애플리케이션 예제의 화면을 보인다. MIF 개발에는 안드로이드 플랫폼만을 사용하였으며 다른 프레임워크나 공개된 소스는 사용되지 않았다.

5.1 블랙보드와 컴포넌트 간의 동기적 호출

본 절에서의 동기적 호출은 프로세스 범위를 넘어서는 별개의 애플리케이션 간에 바인딩을 통해 연결지점을 만들어 연결된 컴포넌트의 함수를 바로 사용할 수 있는 것을 의미한다.

안드로이드의 서비스(Service)는 RPC(Remote Procedure Call)통신을 안드로이드에서 제공하는 기법이다. 컴포넌트들은 서비스를 통해 컴포넌트 간의 동기적 호출을 할 수 있다. 서비스를 이용하기 위해서는 AIDL(Android Interface Description Language) 파일을 생성해야 하며, 해당 AIDL 파일을 다른 컴포넌트도 가지고 있어야 한다. 또한 바인딩 과정을 거쳐야만 동기적 호출이 가능하다. 하지만 이와 같이 각각의 컴포넌트가 서비스를 이용하려면 각각의 컴포넌트 개발자는 자신의 컴포넌트에 대한 AIDL 파일을 만들어야 하고 모든 개발자에게 AIDL 파일을 넘겨서 추가하도록 해야 하며 각각에 대한 바인딩 코드를 작성하도록 해야 한다. 또한 다른 컴포넌트가 추가될 때마다 해당 컴포넌트 개발자에게 AIDL 파일을 넘겨받아 추가적인 작업을 해야 한다. 이는 매우 복잡한 작업이다.

MIF는 이러한 서비스를 각각의 에이전트 개발자가 사용하지 않아도 동기적 호출이 가능하도록 하며, 이를 위해 블랙보드를 사용한다. 블랙보드는 공유 저장소로 각 컴포넌트가 접근하여 데이터를 저장하거나 가져올 수 있는 별도의 독립된 애플리케이션이다. 이 때 컴포넌트와 블랙보드간에 동기적 호출을 하게 된다.

MIF 에서는 AIDL을 사용해 컴포넌트와 블랙보드 간 서비스 바인딩을 통해 원격지 함수 호출이 가능하도록 하였다. 원격지 함수 호출을 할 경우 컴포넌트는 프로세스 범위 밖에 있는 블랙보드의 함수를 원래 자신의 함수처럼 사용할 수 있다. 컴포넌트에서 함수를 호출하면 블랙보드의 함수가 바로 불리며, 리턴할 값이 있을 경우 처리의 결과물을 역시 바로 컴포넌트에 전달한다. 또한 에이전트 컴포넌트가 실행되는 시점에서 서비스 바인딩을 시행하여 최대 1초간의 시간을 잠자며(sleep) 기다린다. 바인딩은 리눅스 커널에서 수행되는데 시간이 다소 걸리는 과정으로, 1초는 여러 번 수행해 본 결과 일반적으로 정상적인 바인딩은 일반적으로 1초 이하의 시간이 걸리는 데에서 지정한 시간이며, 애플리케이션 시작 시의 1초는 사용자가 거의 느끼지 못하는 시간이다. 따라서 사용자는 항상 동기적 호출이 가능한 것처럼 느끼게 된다. 또한 MIF 사용자의 코드 추가 없이 이 과정이 일어나므로 편리하다.

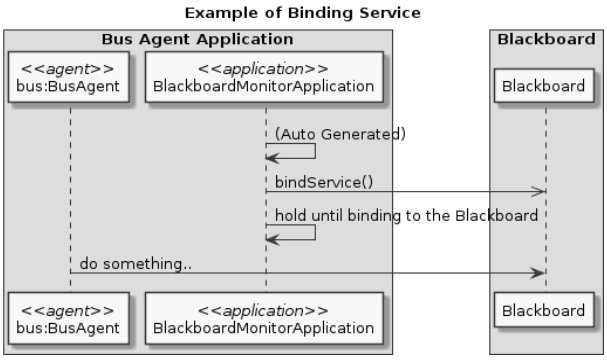


Fig. 6. Example of service binding

Fig. 6과 같이 Bus Agent Application 시작 시 BlackboardMonitorApplication 객체가 자동으로 블랙보드에 서비스 바인딩을 하며, 최대 1초까지 바인딩 여부를 기다리게 되고, 바인딩 되었을 경우 Bus Agent와 블랙보드는 동기적으로 연결되어 원하는 함수를 부를 수 있다. 이 때 BlackboardMonitorApplication 객체에 관한 설명은 5.4절에서 설명한다.

본 논문에서 동기적 호출방식을 사용한 연결은 컴포넌트들과 블랙보드간의 연결 방식이다. Fig. 5 에서의 subscribe와 update가 이러한 호출 방식을 취한다.

5.2 컴포넌트 간의 비동기적 호출

본 절에서의 비동기적 호출은 메시지를 전달하고자 하는 상대 프로세스의 메시지 큐에 호출을 넣어 수행하도록 하는 것이다. 예를 들어 Fig. 7은 Fig. 4에서 Weather Card가 Weather Agent에게 비동기적 호출을 하는 과정을 간단히 나타낸다. Weather Card는 Weather Agent가 현재 하고 있는 일이 무엇인지에 관계 없이 Weather Agent의 메시지 큐에 수행할 일을 넣기만 한다. 이후 루퍼가 다른 일을 한 후 Weather Agent가 넣어둔 일의 차례가 되면 수행하게 되고, 그 결과 또한 같은 방식으로 Weather Card에게 전달한다. 따라서 호출을 한 프로세스는 리턴을 기다리지 않고 다른 일을 처리할 수 있게 된다.

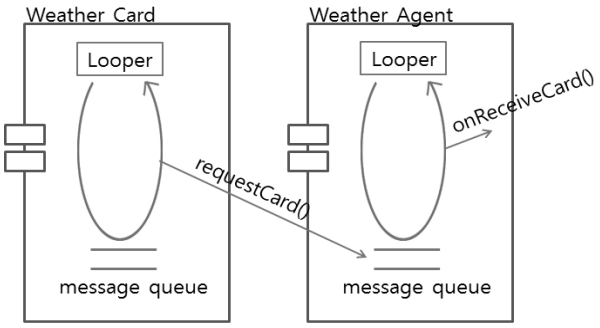


Fig. 7. Asynchronous Method Call

안드로이드에서는 인텐트 방송을 통해 이와 같은 비동기적 호출을 지원한다. 인텐트 방송을 통해 비동기적 호출을 하기 위해서는 우선 인텐트 방송을 받기위한 브로드케스트 리시버(broadcast receiver)를 등록해야 하며, 인텐트 방송을 보내는 상대방이 어떤 액션(action)을 통해 메시지를 전달할지 알아야 한다. 여기서 액션이란 안드로이드 시스템은 SMS, 배터리 등 많은 인텐트를 모든 애플리케이션에게 발송하므로 그 중 어떤 것에 대해 반응할지에 대한 구분 문자라 할 수 있다. 이에 따라 에이전트 개발자가 안드로이드의 인텐트 방송을 통해 비동기적 호출을 제작하게 될 경우 직접 브로드케스트 리시버를 제작해서 등록해야 하며 에이전트가 추가될 때 마다 서로에 대한 액션 문자열을 주고 받아야 하고, 이에 대한 처리를 브로드케스트 리시버에 추가해야 한다. 또한 결과값을 상대에게 리턴하기 위한 함수도 계속 수정해줘야 한다. 이는 매우 복잡한 과정이다.

MIF에서는 안드로이드의 인텐트 방송과 브로드케스트 리시버를 감추고 이를 이용해 만든 새로운 함수로써 비동기적 호출 방식을 제공한다. 브로드케스트 리시버를 등록하는 과정 또한 자동으로 진행되므로, 사용자는 별도의 처리 없이 MIF의 함수를 사용함으로써 간단히 에이전트 간의 비동기적 호출을 할 수 있다. MIF는 인텐트 방송을 기반으로 만들었으므로 동시에 여러 프로세스에게 값 전달이 가능하고, 값 변경 알림 리스너 및 에이전트 간의 카드 생성 호출은 서로에 대한 문자열 형태의 이름만을 알면 간단히 호출 가능하다.

본 논문에서 비동기적 호출은 블랙보드에서 값 변경 알림을 모든 에이전트에게 전송할 때, 상황인지 애플리케이션에서 서비스 에이전트에게 카드 생성 요청을 보낼 때 사용된다. 4.2절의 Fig. 4가 비동기 호출의 한 예로 상황인지 애플리케이션에서는 requestCard를 통해 에이전트에게 카드 생성 요청을 보내지만 즉시 리턴을 기다리는 것은 아니다. 에이전트 자신의 일을 하고 있다가 onReceiveCard를 통해 요청에 대한 응답이 왔을 경우 그에 대한 처리를 한다.

5.3 에이전트 초기화

에이전트 생성 시 개발자는 Agent 클래스를 상속받아 새로운 에이전트를 제작한다. 이 때 부모 클래스인 Agent 자체의 초기화 과정이 존재하고, 개발자 자신이 필요한 초기화 과정도 존재한다. 문제가 되는 것은 에이전트의 초기화 과정이다.

Fig. 8은 Agent를 상속받아 새로운 에이전트를 만드는 코드의 일부이다. WeatherAgent의 생성자에서 super를 통해 상위 클래스의 생성자를 호출하며 이 때 onInit함수와 감춰진 에이전트 등록 함수가 불리게 된다. super에서 onInit과 에이전트 등록을 수행하지 않을 경우 사용자가 직접 해줘야 하므로 복잡도가 증가하게 된다. 이 때 아무런 처리도 하지 않으면 메소드의 호출 순서는 Fig. 9과 같다.

```
public class WeatherAgent extends Agent {
  WeatherAgent() {
    super();        // Call constructor of super class
    process something in the WeatherAgent constructor;
  }

  @Override
  protected void onInit() {
    process something after creation of Agent;
  }
}
```

Fig. 8. Example of Agent creation

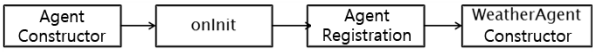


Fig. 9. General order of method calls

MIF에서는 Agent 클래스가 만들어질 때 핸들러를 통해 애플리케이션의 메인 루퍼의 큐에서 에이전트 등록 함수가 불리도록 하여 이 문제를 해결하였다. Fig. 10은 이러한 에이전트 초기화 방식을 나타낸다. 큐에는 Agent생성자와 WeatherAgent 생성자가 들어간다. 이 때 Fig. 9과 같은 문제를 해결하기 위해 루퍼에서 Fig. 10.(1)과 같이 Agent 생성자를 부를 때 Fig. 10.(2),(3)과 같이 onInit과 에이전트 등록 메소드를 루퍼의 큐에 넣게 된다. 큐는 선입선출의 구조로써 마지막에 들어온 것이 마지막에 처리된다. 따라서 Fig. 10.(4),(5)와 같이 순차적으로 처리되게 되어 Fig. 11과 같이 정상적으로 메소드 호출이 이루어지게 된다.

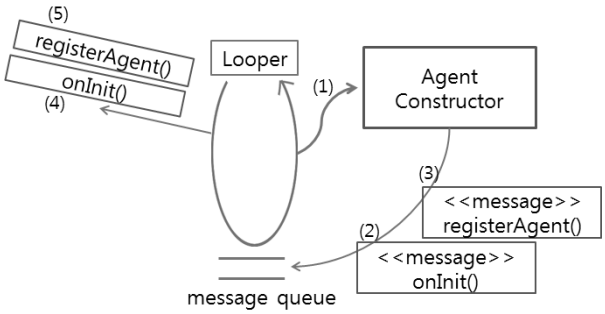


Fig. 10. Order of Agent initialization

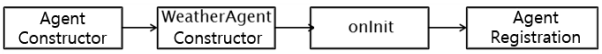


Fig. 11. Order of method calls using main looper

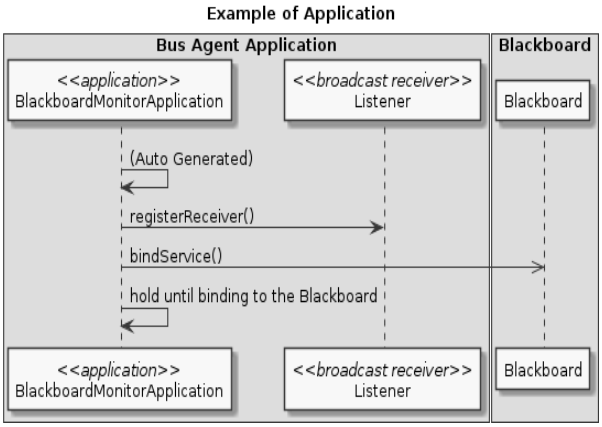


Fig. 12. How to initialize the Application

5.4 애플리케이션 초기화

각 컴포넌트가 5.1절과 5.2절 같은 호출방식을 사용하기 위해서는 동기적 호출을 위해 블랙보드에 서비스 바인딩을 해야 하고, 비동기적 호출을 위해 브로드케스트 리시버를 등록하는 과정이 필요하다. 이 과정을 개발자에게 맡길 경우 안드로이드 플랫폼을 감추고 동작시킬 수 없으며, 복잡도가 매우 증가하게 된다. 따라서 모든 연결 및 등록 과정은 MIF에서 자동으로 이루어져야 하며, 사용자에게 일련의 과정이 보이지 않아야 한다. 또한 블랙보드와의 서비스 바인딩 및 브로드케스트 리시버는 에이전트가 시작될 때부터 종료될 때 까지 항상 유지되어야만 한다.

이를 위해 Fig. 12의 시퀀스 다이어그램과 같이 버스 에이전트 실행 시 BlackboardMonitorApplication는 자동으로 생성되며 리스너를 등록하고 블랙보드에 서비스를 바인딩 시킨다. 그리고 서비스 바인딩을 최대 1초 정도 기다리며 서비스가 바인딩 되길 기다린다. 즉, 비동기적인 바인딩 서비스가 완료되길 기다려 동기적 방식으로 변경한다. 만약 서비스 바인딩이나 브로드케스트 리시버 등록이 이루어지지 않으면 해당 에이전트는 MIF에서 분리된 별개의 애플리케이션일 뿐이다.

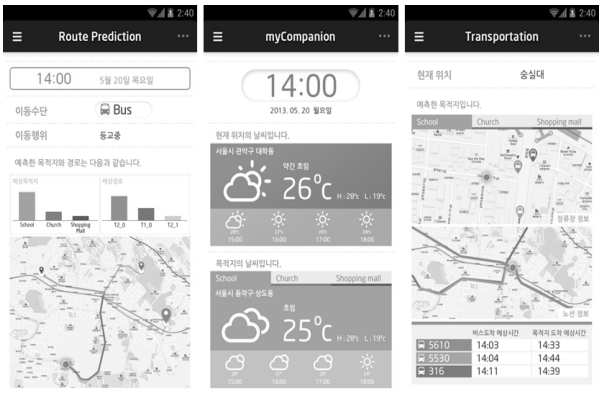
5.5 블랙보드 사용 방식 간소화

블랙보드에는 많은 종류의 데이터가 들어갈 수 있어야 한다. 하지만 블랙보드는 개발자가 어떤 형태의 데이터를 기록할지 모르므로 기록 가능한 데이터의 종류에 제한을 두어서는 안된다. 이를 위해 블랙보드의 필드를 개발자 마음대로 추가 및 제거 가능하도록 했고, 필드에 들어가는 데이터는 자바의 String 타입의 문자열을 사용하도록 했다. 다른 데이터 형식의 경우 String 타입으로 변환하여 사용 가능하다.

5.6 라이브러리 구분을 통한 복잡도 감소

컴포넌트는 크게 블랙보드를 사용하는 에이전트와, 에이전트에게 카드 생성을 요청하는 애플리케이션으로 나뉜다. 이 때, 하나의 라이브러리를 사용하게 되면 블랙보드를 사용하지 않는 애플리케이션에서도 블랙보드에 자동으로 연결이 되어 있게 되며, 불필요한 브로드케스트 리시버 등록 과정도 진행된다. 또한 블랙보드를 사용하지 않아도 되는 애플리케이션에서 블랙보드의 값을 수정 가능하도록 해서는 안된다. 따라서 블랙보드 사용을 위한 라이브러리와, 카드 생성 요청을 위한 라이브러리로 나누어 서로 필요한 연결 및 등록 과정만을 거치며, 필요한 함수에만 접근 가능하도록 하여 복잡도를 낮추도록 했다.

5.7 MIF를 이용한 예제 화면



(a) (b) (c)

Fig. 13. Example of context-awareness Applications using MIF

MIF를 통해 제작된 상황인지 애플리케이션의 예제 화면이다. Fig. 13.(a),(b),(c) 각각은 연관된 에이전트에게 Fig. 4와 같은 방식으로 카드 생성 요청을 보내어 받은 결과를 화면상에 출력하고 있다. 에이전트들은 센서 로거를 통해 모인 정보와 그 정보를 추론한 정보가 존재하는 블랙보드에 접근 가능하다.

Fig. 13(a)는 블랙보드의 정보를 기반으로 사용자가 특정 위치에서 어느 길로 갈 확률이 높은지를 추론하는 에이전트에게 정보 요청을 하여 받은 정보를 화면상에 띄운 것이다. 화면상에서는 버스로 등교중인 상황에서 어느 길로 갈 확률이 높은지에 대한 추론 결과를 나타낸다.

Fig. 13(b)에서는 Weather Agent에게 카드 생성 정보를 요청하고 받은 정보를 화면상에 나타낸 결과이다. 날씨 에이전트는 블랙보드에 다른 에이전트가 기록한 5시간 뒤의 이용자의 위치 추론 정보를 가져온 후, 해당 위치의 날씨 정보와 현재 위치의 날씨 정보를 인터넷을 통해 얻어와 결과로 리턴한다. 이 정보는 현재 날씨와 사용자의 목적지라 예상되는 지점의 날씨를 나타낸 것이다.

Fig. 13(c)는 Bus Agent로 부터 버스 정류장 근처에 갔을 경우 사용자의 다음 목적지라 예상되는 지점에 도착하는 버스를 기준으로 우선순위에 따른 버스 도착 예정 시간에 대한 정보를 얻어와 화면상에 출력한다.

MIF에서 상황인지 애플리케이션들은 Fig. 4에서 본 것과 같은 방식으로 Weather Agent와 같은 Service Agent에게 정보를 요청하여 스마트폰 사용자에게 서비스를 제공할 수 있다.

6. 결 론

본 논문에서는 스마트폰 상황인지 시스템을 이루는 여러 시스템을 컴포넌트화 하여 유기적으로 동작할 수 있도록 하는 MIF를 설계 및 구현하였다. 상황인지에 필요한 각 컴포넌트가 요구하는 서비스를 제공하고, 안드로이드 플랫폼을 감춰서 복잡도와 오류를 최소화할 수 있으며, 코드가 간결화 되어 개발자에게 편의를 제공해준다. 이는 결과적으로 전체 시스템의 개발 시간을 단축시키며, 오류의 감소로 시스템 전체의 성능 향상으로 이어진다.

참 고 문 헌

[1] Towards a Better Understanding of context and context-awareness, Anind K. Dey and Gregory D. Abowd., Georgia Institute of Technology, College of Computing., Technical Report GIT-GVU-99-22, 1999.

[2] M. Weiser, “Some computer science issues in ubiquitous computing,” Communications of the ACM, Vol.36, No.7, pp.75-84, July, 1993.

[3] E. Horvitz, S. Dumais, P. Koch, “Learning predictive models of memory landmarks,” 26th Annual Meeting of the Cognitive Science Society, pp.1-6, 2004.

[4] Gil Heo, Seoungkuk Yang, Mihwa Kim, hyangjin Lee, Jemin Kim, Junghwa Choi, Youngtack Park, “A Context-Aware System for Smart Phone Using Rule Base Inference Engine”, Korean Institute of Information Scientists and Engineers, Vol.37, No.2, 2010.

[5] Changmin Jung, Jaehoon Bang, Kihyun Han, Haneul Yeom, Sungyoung Lee, “Smartphone based activity recognition for real environment”, Korean Institute of Information Scientists and Engineers, 2013.

[6] Providing Architectural Support for Building Context-Aware Applications, Anind K. Dey., College of Computing, Georgia Institute of Technology, Dec., 2000.

[7] Junghwa Choi, Jemin Kim, Eunseok Seo, Youngtack Park, “Rule-based Context Modelling for Context-aware Services in Smart Phone Environments”, Korea Computer Congress, Vol.38, No.3, 2011.

[8] Component Group, “Component-based Development-an Overview”. White Paper, Component Group. at URL : www.componentgroup.com/whitepapers/overview.html, 2000

[9] Dongpyo Hong, Woontack Woo, “wear-UCAM: Unified Context-aware Application Model for Wearable Computing”, Korea Computer Congress, Vol.33, No.1, 2006.

[10] Jim Gemmel, Gordon Bell, and Roger Lueder, “MyLifeBits: A Personal Database for Everything”, Communications of the ACM, Vol.49, No.1, January, 2006.

[11] Susan Dumais, et al., “Stuff I’ve Seen: A System for Personal Information Retrieval and Re-Use”, SIGIR, 2003.

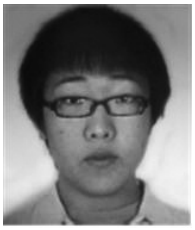
[12] Gihyun Gong, Sangkeun Lee, Sanggoo Lee, “A Lifelog System based on Context information”, Korean Institute of Information Scientists and Engineers, Vol.36, No.1, 2009.

[13] Minoh Heo, Myunggu Kang, Byoungkwon Lim, Kyubeak Hwang, Youngtack Park, Byoungtack Zhang, “Real-time Route Inference and Learning for Smartphone Users using Probabilistic Graphical Models”, Korea Computer Congress, Vol.39, No.6, 2012.

[14] Seyoung Hwang, Taejun Ryu, Sangwon Park, “The Design and Implementation of User Context Information Extraction System on Android-Based Smartphone”, Korean Institute of Information Scientists and Engineers, 2013.

[15] Google Now, http://www.google.com/landing/now/

[16] Siri, http://www.apple.com/kr/ios/siri/



황 세 영

e-mail : hwangs88@naver.com

2007년~현 재 한국외국어대학교 정보통신공학과 학사과정

관심분야 : Database, Mobile Application, Context Awareness

이 형 욱

e-mail : lhu114@nate.com

2008년~현 재 한국외국어대학교 정보통신공학과 학사과정

관심분야 : Database, Mobile Application

박 상 원

e-mail : swpark@hufs.ac.kr

1994년 서울대학교 컴퓨터공학과(학사)

1997년 서울대학교 컴퓨터공학과(석사)

2002년 서울대학교 컴퓨터공학과(박사)

2002년~2003년 세종사이버대학교 디지털 콘텐츠학과 전임강사

2003년~현 재 한국외국어대학교 정보통신공학과 교수

관심분야 : Flash Memory, Embeded Database, Mobile Computing