사용 시나리오 분석을 위한 부분 실행 아키텍처의 도출 방법

정필수*, 허준행*, 안휘*, 강성원*, 권오인**, 서만수**

* 한국과학기술원 전산학과
대전광역시 유성구 대학로 291
{psjung, jhcorea, ahnhwi, sungwon.kang}@kaist.ac.kr
** LG 전자 MC 연구소 Architecture 팀
서울시 금천구 벚꽃로 298
{abraham.kwon, mickey.seo}@lge.com

요약: 실행 아키텍처를 구축하는 과거의 방법들은 대부분 완전한 실행 아키텍처를 재구축하기 때문에, 특정 사용 시나리오에 관련된 분석의 수행을 위해서는 불필요한 부분까지 포함하여 분석에 들어가는 노력을 크게 만들고 분석의 정확성도 떨어뜨릴 수 있는 단점을 갖는다. 본 논문에서는 주어진 사용 시나리오로부터 부분 실행 아키텍처를 도출하는 체계적인 방법을 제안한다. 주어진 사용 시나리오에 맞추어진 부분 실행 아키텍처는 해당 시나리오에 관련된 컴포넌트와 커넥터만을 포함함으로써 관련 시나리오 관점의 분석을 용이하게 만들고 도출과정에 전문가의 개입을 최소화함으로써 높은 정확성을 갖는 아키텍처가 재구축된다. 본 연구의 사례연구에서는 제안 방법을 적용하여 안드로이드 오픈소스 프로젝트의 SystemUI 서브시스템로부터 부분 실행 아키텍처를 도출하고, 도출된 부분 실행 아키텍처에 대하여 사용 시나리오의 분석을 수행하여 제안 방법이 효과적임을 보인다.

핵심어: 실행 아키텍처, 부분 실행 아키텍처, 아키텍처 재구축, 사용 시나리오, 실행 요소

1. 서론

소프트웨어 아키텍처는 소프트웨어 지배적 시스템(software-intensive system)의 복잡한 구조를 이해하는데 도움을 준다. 뿐만 아니라, 시스템의 변화를 관리하고 시스템의 구성 요소를 재사용하기 위해 활용될 수 있는 중요한 개발 산출물이다. 소프트웨어 아키텍처의 여러 가지 뷰 중에서 실행 아키텍처는 실행 측면의 시스템 품질을 분석하고 평가하기 위해 필요한 실행 시 시스템의 아키텍처 뷰이다. 이를 활용하여 실행요소들로 구성된 구조와 행위를 관찰하고 발생하는 오류를 발견하거나 성능 분석을 할 수 있다.

현업에서는 시스템의 기능이 올바르게 수행되고 요구되는 성능을 확보하였는지 평가하기 위해, 실행 시 시스템의 사용 시나리오들을 분석하는 활동이 중요하다. 이때, 실행 아키텍처는 사용 시나리오 관점의 분석을 수행하는데 중요한 역할을 한다. 그러나 실행 아키텍처를 재구축하는 과거의 연구들은 대부분 완전한 실행 아키텍처를 재구축하여, 특정 사용 시나리오 관점의 분석을 위하여 불필요한 부분까지 포함하고, 이로 인해 해당 시나리오 관점의 분석을 어렵게 만든다. 따라서 본 논문은 주어진 사용 시나리오와 관련된 컴포넌트와 커넥터만을 포함하는 부분 실행 아키텍처를 도출하는 방법을 제시하고자 한다.

지금까지 부분 실행 아키텍처를 재구축하는 방법은 거의 연구되지 않았다. 따라서 부분 실행 아키텍처 재구축 연구를 위해 기존의 완전한 실행 아키텍처 재구축 방법을 활용할 수 있다. 완전한 실행 아키텍처를 재구축하기 위한 방법들이 많은 연구들에서 과거에 제안되었다[3,4,5,7,11]. 이들은 사용 시나리오 관점의 분석을 지원하는데 있어서 다음과 같은 단점을 가진다. Symphony[3]와 Focus[7]는 시스템의 변화를 관리하거나 발생된 문제를 해결하기 위해서 아키텍처 진화 관점에서 재구축을 수행하므로 시스템의 기능이나 품질을 분석하기 위한 시나리오 관점 분석을 수행하기 어렵다. 그리고 DiscoTect[4]는 시스템과 사전에 설계된 문서간의 적합성(conformance)을 평가하기 위해 전문가가 많이 개입되어야 한다는 단점을 지닌다. QADSAR[5]는 시나리오 관점의 분석을 지원하지만, 시스템의 품질을 평가하기 위해 품질 시나리오 관점의 분석만 지원한다. 끝으로, FARSAR[11]는 시나리오관점에서 아키텍처를 재구축하는 방법을 제시하지만, 구체적인 가이드라인을 제시하지 않는다는 단점을 지닌다.

본 논문은 기존 연구들보다 사용 시나리오 관점의 분석을 더 용이하게 만들기 위하여, 주어진 시나리오로부터 부분 실행 아키텍처를 체계적으로 도출하는 방법을 제안한다. 이를 위해, 제안 방법은 전문가가 개입하지 않고 얻을 수 있는 입력물과 전문가의 개입을 최소화하여 수행할 수 있는 절차를 정의하고,

이를 위한 구체적인 가이드라인을 제공한다. 제안 방법을 통해 도출된 부분 실행 아키텍처는 주어진 사용 시나리오와 관련된 컴포넌트와 커넥터만을 도출함으로써 해당 시나리오 관점의 분석을 용이하게 만들고, 도출과정에 전문가의 개입을 최소화하여 도출된 실행 아키텍처는 높은 정확성을 갖게 된다.

이 논문은 다음과 같이 구성된다. 제 2 절에서는 아키텍처를 재구축하는 기존 연구들을 소개한다. 제 3 절에서는 주어진 사용 시나리오에 대한 부분 실행 아키텍처를 도출하는 방법을 제시하고 제 4 절에서는 사례연구를 통해 부분 실행 아키텍처를 도출하는 과정을 보인다. 제 5 절에서는 사례연구의 결과로부터 발견한 사실들을 기술하고 평가한다. 끝으로, 제 6 절에서는 제안 방법의 기여와 향후 연구 방향에 대해 논의한다.

2. 아키텍처 재구축에 관한 기존 연구

과거에 적절히 확보되지 못한 아키텍처를 재구축하는 방법이 많은 연구들에서 제안되었다. 이들은 크게 아키텍처 문제를 해결하기 위한 연구와 일반화된 아키텍처 재구축을 위한 프레임워크를 제공하는 연구로 분류할 수 있다.

아키텍처적인 문제(architectural problem)를 해결하기 위한 연구로는 Symphony[3], Focus[7], DiscoTect[4] 등이 있다. Symphony 는 아키텍처 문제를 해결하기 위한 아키텍처 관점 주도적(view-driven) 재구축 방법으로서, 아키텍처 문제 분석 단계와 아키텍처 재구축 단계로 구성된다. 이는 아키텍저적 문제 해결을 위해 고려해야 할 뷰를 재구축하는 가이드라인을 제공한다. 또한, 다른 연구들의 방법들과 비교할 때 사용될 수 있는 공통 체계(common framework)를 제공한다. Focus 는 아키텍처 재구축을 통해, 객체 지향 시스템에서 변화에 민감한 부분을 찾아내기 위해 시스템을 분석하는 연구이다. DiscoTect 는 시스템의 구현이 기존에 결정된 설계 결정과 일치하는지 평가하기 위한 연구이다. 이를 위해, 시스템의 실행 요소들을 추출하여 실행 아키텍처를 재구축하여 시스템이 기존 설계 결정에 맞게 구축되었는지 그 적합성을 확인한다. 세 연구는 아키텍처 재구축을 통해, 시스템의 기능을 분석하고 품질을 평가하는데 활용될 수 있다. 그러나 시스템에서 원하는 부분을 분석하기 위해, 필요한 아키텍처 요소만을 재구축할 수는 없기 때문에 정확한 분석이 어렵다는 점에서 한계를 지닌다.

일반화된 아키텍처 재구축을 위한 프레임워크를 제공하는 연구로는 QADSAR[5], FASAR[11] 등이 있다. QADSAR 는 아키텍처를 재구축하여 시스템의 품질 속성을 평가하는 연구이다. 이를 위해, 품질 속성 시나리오를 정의하여 분석 범위를 결정하고 함수, 클래스, 파일, 디렉토리와 그들간의 관계를 추출하여 아키텍처 요소를 도출한다. FASAR 는 도구를 기반으로 아키텍처를 재구축하는 프레임워크를 제공한다. 이를 위해, 대상 시스템의 특성을 추출하는 단계, 자동화 도구 선택 단계, 아키텍처 재구축 단계를 통해, 도구를 기반으로 아키텍처를 재구축하는 프레임워크를 제공한다. 두 연구는 시나리오를 통해 분석 범위를 결정하여 시스템의 원하는 부분에 대해서 아키텍처 재구축이 가능하다. 그러나 QADSAR 는 재구축된 아키텍처를 사용하여 품질 속성 시나리오를 검증하는 방법만 제안하기 때문에 사용 시나리오의 분석에 대한 방법을 지원하지 않는다. 그리고 FASAR 는 사용 시나리오를 수행하면서 나타나는 시스템의 실행 요소를 활용하여 실행 아키텍처를 도출하지만 FASAR 는 시스템의 실행 요소로부터 실행 아키텍처를 도출하는 구체적인 방법을 제시하지 않는 한계점을 가지고 있다.

3. 부분 실행 아키텍처 도출 방법

본 절에서는 사용 시나리오에 맞추어진 컴포넌트와 커넥터를 도출하여 부분 아키텍처를 도출하는 방법을 제시한다. 그림 1 은 부분 실행 아키텍처를 도출하는 절차이다. 절차를 수행하기 위한 세 가지 입력물은 다음과 같다:

1) **사용 시나리오**는 사용자가 어떻게 시스템을 사용할 수 있는지에 대한 명세로[8], 이는 시스템의 기능을 분석하고 성능을 평가하기 위한 부분 실행 아키텍처를 도출하는데 사용된다.
2) **모듈 아키텍처**는 시스템의 기본 구현 단위인 모듈과 모듈간의 관계를 나타낸 것으로[8], 실행 요소와 함께 컴포넌트와 커넥터를 도출하는데 사용된다. 모듈 아키텍처는 사람의 개입을 최소화하기 위해 소스코드를 이용하여 Enterprise Architect[6], Rational Rhapsody[9], StarUML[12] 등의 역공학 도구를 통해 얻은 결과물을 의도에 맞게 재구성할 수 있다.
3) **실행 파일**은 시스템이 사용 시나리오를 수행하면서 생성되는 실행 요소들을 추출하기 위해 사용된다.

이 입력물은 전문가의 판단이 개입되지 않고 얻을 수 있기 때문에 도메인 지식 수준에 큰 영향을 받지 않고 일관되게 얻을 수 있는 개발 산출물이다.

아래 각 소절에서는 그림 1 의 절차의 각 단계를 설명한다. 각 단계의 설명에서는 그 단계가 자동화된(automated) 단계인지, 아니면 사람의 개입을 필요로 하는지를 명확히 구분한다.

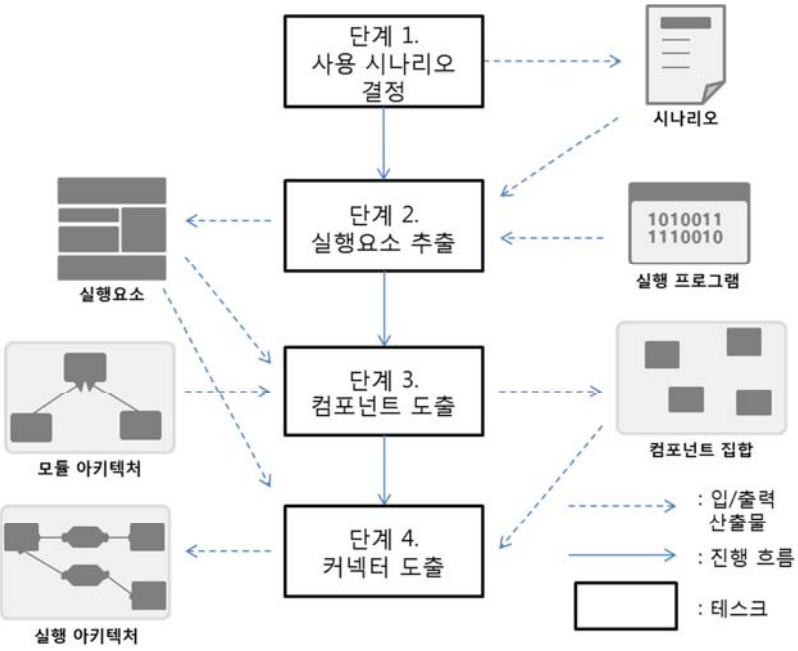

그림 1. 부분 실행 아키텍처의 도출 절차

3.1 단계 1: 사용 시나리오 결정

사용 시나리오는 기존에 작성된 아키텍처 문서에 기술된 시나리오 중에 선택하거나 새롭게 정의할 수 있다. 그러나 규모가 큰 시나리오는 시나리오를 수행하는 절차를 명세할 때 사람의 판단이나 결정이 크게 개입될 수 있기 때문에 일관된 결과를 얻기 어렵다. 따라서 최소한의 기능을 수행하는 시나리오를 정의해야 정확한 결과를 얻을 수 있다. 규모가 큰 시나리오는 작게 나눠서 각각의 시나리오에 대하여 부분 실행 아키텍처 도출 절차를 적용하고 이들을 통합하는 과정이 필요하다.

이 단계는 자동화되지 않은 단계이다. 하지만 최소한의 기능을 가지는 시나리오를 결정하고 그 절차를 명세함으로써, 결과는 이 단계를 수행하는 사람에 따라 크게 달라지지 않는다.

3.2 단계 2: 타겟 시스템의 실행 요소 추출

결정한 시나리오에 따라 시스템이 실행되면서 객체, 쓰레드, 프로세스 등의 실행 요소들을 생성한다. 부분 실행 아키텍처 도출 절차를 수행하기 위해서 객체 식별자와 객체를 생성한 클래스 명, 객체간의 함수 호출정보가 필요하다. 이들을 수집하기 위해 바이트 코드 인젝션 도구, 로그 삽입 도구, 프로파일링 도구 등을 사용 할 수 있다. 동일한 시나리오에 대해 추출되는 실행 요소는 항상 동일하기 때문에 일관된 결과를 얻을 수 있다.

이 단계는 자동화된 단계이다. 그 이유는 시스템이 결정된 사용 시나리오를 수행하면서 실행 요소들을 추출하는 활동은 실행 요소 수집 도구를 통해 수행될 수 있기 때문이다.

3.3 단계 3: 컴포넌트 도출

컴포넌트는 단계 2 에서 얻은 실행 요소들을 그룹화하여 도출할 수 있다. 그러나 그룹화를 위한 명확한 기준이 필요하다. 제안 방법은 모듈과 객체간의 인스턴스화 관계를 활용하여 그룹화를 수행한다. 예를 들어, 그림 2 와 같이, 각 객체를 인스턴스화 한 클래스가 어떤 모듈에 포함되는지 연결함으로써 그룹화를 수행할 수 있다. 제안 방법은 한 모듈로부터 생성된 객체 그룹은 그 모듈의 기능을 제공하는 아키텍처 요소이므로 이를 컴포넌트로 정의한다. 이 경우, 컴포넌트와 모듈이 일대일 관계가 된다. 일반적으로 컴포넌트와 모듈이 다대다 관계를 가질 수 있다. 이 경우는 동일한 모듈에 대응되는 실행 요소들을 세부 기능에 따라 서로 다른 그룹으로 나눌 수 있으며 이 경우에도 제안 방법은 변경없이 수행된다.

컴포넌트의 추상화 수준은 모듈 아키텍처를 구성하는 모듈의 추상화 수준에 따라서 결정된다. 예를 들어, 모듈의 추상화 수준이 높으면 하나의 모듈에 많은 기능을 포함하므로 많은 실행 요소와 연결되어 컴포넌트의 추상화 수준이 높아진다. 반대로, 모듈의 추상화 수준이 낮으면 해당 모듈과 연결되는 실행 요소가 적어서 컴포넌트의 추상화 수준이 낮아진다. 따라서 원하는 추상화 수준의 실행 아키텍처를 얻기 위해 모듈 아키텍처를 신중히 구축하는 것이 중요하다.

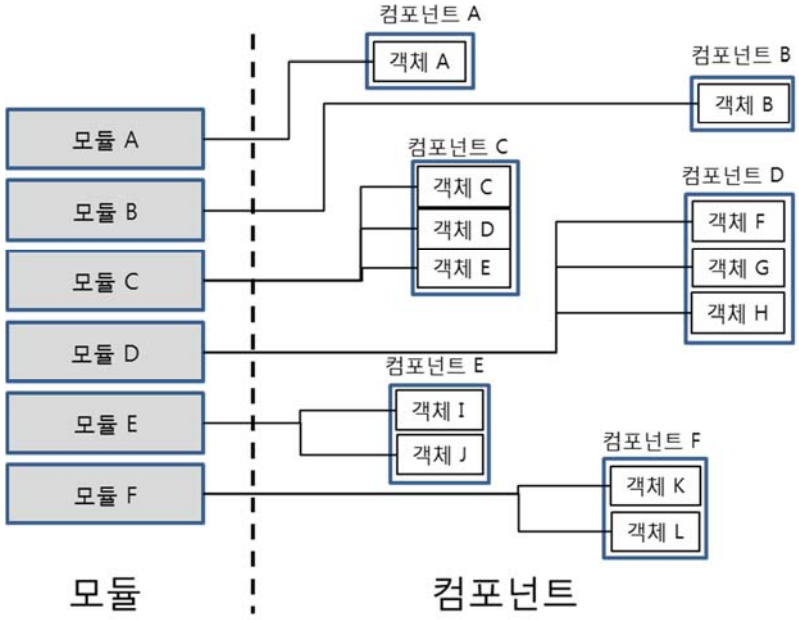

그림 2. 모듈과 객체간의 연결을 통한 컴포넌트 도출

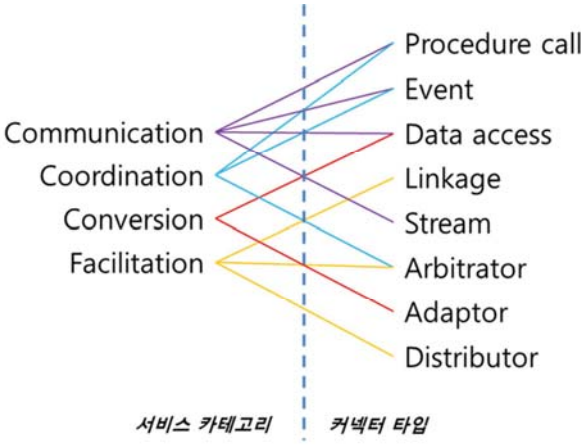

그림 3. 서비스 카테고리와 커넥터 타입의 관계

이 단계는 자동화된 단계이다. 그 이유는 각 모듈에 포함된 클래스로부터 인스턴스화 된 객체를 연결하는 활동과 같은 모듈과 연결된 객체들을 그룹화하는 활동은 사람의 개입 없이 자동으로 수행될 수 있기 때문이다.

3.4 단계 4: 커넥터 도출

제안 방법은 커넥터를 도출할 때 사람의 개입을 최대한 줄이기 위해 Taylor 등의 커넥터 분류법을 활용한다. Taylor 등[10]은 그림 3 과 같이, 커넥터를 네 가지 서비스 카테고리와 여덟 가지 커넥터 타입으로 분류하였다. 그리고 서비스 카테고리와 커넥터 타입 간의 관계를 정의하였다. 예를 들어, Procedure call 은 한 컴포넌트가 다른 컴포넌트에게 인자를 전달하고 기능의 수행 결과를 얻는 과정이기 때문에 두 컴포넌트는 서로 통신서비스를 제공한다고 볼 수 있다. 마찬가지로, Event, Data access, Stream 은 컴포넌트간의 데이터를 주고 받기 위한 수단이므로 통신서비스를 제공한다고 볼 수 있다.

커넥터가 제공하는 서비스는 기존에 작성된 설계 문서를 활용하여 도메인 전문가가 개입하여 판단할 수 있지만, 이는 도메인 전문가에 따라 결과가 크게 달라질 수 있기 때문에 적절하지 않다. 따라서 단계 2 를 통해 추출한 실행 요소의 정보를 활용하여 어떤 커넥터 타입을 지니고 있는지 판단한다면 사람의 개입을 줄일 수 있다.

커넥터를 도출하기 위해, 제안 방법은 생성된 객체 정보와 객체간의 함수 호출 정보를 활용한다. 수집한 객체 정보와 함수 호출 정보를 분석하면 두 컴포넌트간의 커넥터 타입을 결정할 수 있다. 예를 들어, 두 컴포넌트 간에 호출 되는 함수 원형(prototype)이 event 객체를 포함하는 경우, 두 컴포넌트간의 커넥터 타입을 Event 타입으로 정의할 수 있다. 또한, 두 컴포넌트간에 SQL 객체를 주로 사용하거나 데이터베이스 접근 함수를 사용하는 경우, 이들은 데이터를 접근하기 위해 사용되는 객체이므로 Data access 타입으로 정의할 수 있다. 마찬가지로, pipe 객체를 사용하거나 TCP/UDP 또는 소켓 객체를 사용하는 경우는 이들은 데이터 스트림을 전달하기 위해 사용되기 때문에 Stream 타입으로 정의할 수 있다. 그 밖에도 Taylor 등의 연구는 각 커넥터 타입이 어떤 구현 스타일에 적용되는지 기술하고 있다.

이 단계는 반자동화된(semi-automatic) 단계이다. 그 이유는 커넥터 타입을 결정하는 활동은 사람의 도메

인 지식 혹은 프로그래밍 지식을 요구하지만, 이 활동은 자동화된 방법으로 얻을 수 있는 실행 요소들을 기반으로 명확히 구분된 여덟 가지 커넥터 타입 중 한 가지를 결정하므로 사람의 개입을 줄일 수 있기 때문이다.

4. AOSP 를 이용한 사례연구

이 절에서는 3 절에서 소개된 제안 방법을 적용한 사례연구를 수행한다. 사례연구에서는 AOSP(Android Open Source Project) Jellybean 버전의 SystemUI 서브시스템을 재구축한다. AOSP 는 구글에서 제공하는 안드로이드 오픈소스이며, SystemUI 는 안드로이드 장비의 설정 및 시스템 정보 열람에 대한 유저 인터페이스를 제공하며, 120 여개의 클래스들로 구성된다.

본 사례연구의 입력물은 사용 시나리오, 모듈 아키텍처, AOSP 실행파일로 구성된다. 사용 시나리오는 사용자가 AOSP SystemUI 서브시스템에서 수행할 수 있는 시나리오가 될 수 있다. 표 1 은 SystemUI 사용 시나리오 목록의 일부이다. 이들 중, 원하는 시나리오를 선택할 수 있다. 모듈 아키텍처는 사람의 개입이 반영되지 않도록 자동화 도구를 사용하거나 적절한 규칙을 적용할 수 있다. 본 사례연구에서는 모듈 아키텍처를 Enterprise Architecture 도구를 활용하여 그림 4 와 같이 나타내었다. 모듈은 SystemUI 서브시스템의 각 패키지로 정의하고, 그 중 하위 패키지가 존재하는 패키지를 서브시스템으로 정의하였다. 그리고 모듈간의 관계는 Enterprise Architecture 도구로 추출된 클래스들간의 관계를 활용하여 도출하였다. 실행 파일은 AOSP 를 통해 컴파일 하여 얻을 수 있다.

4.1 단계 1: AOSP SystemUI 사용 시나리오 선택

표 2 는 원하는 시나리오에 대한 부분 실행 아키텍처를 확보하기 위해 표 1 의 사용 시나리오들 중 하나를 명세한 것이다. 선택한 사용 시나리오는 사용자가 안드로이드 장비의 화면 밝기를 설정하는 기능을 나타내는 시나리오이다.

표 1. AOSP SystemUI 의 사용 시나리오 목록

시나리오 번호	시나리오 명칭
1	와이파이 설정 변경
2	화면 밝기 설정
3	소리 설정
4	시스템 알림 확인
…	….

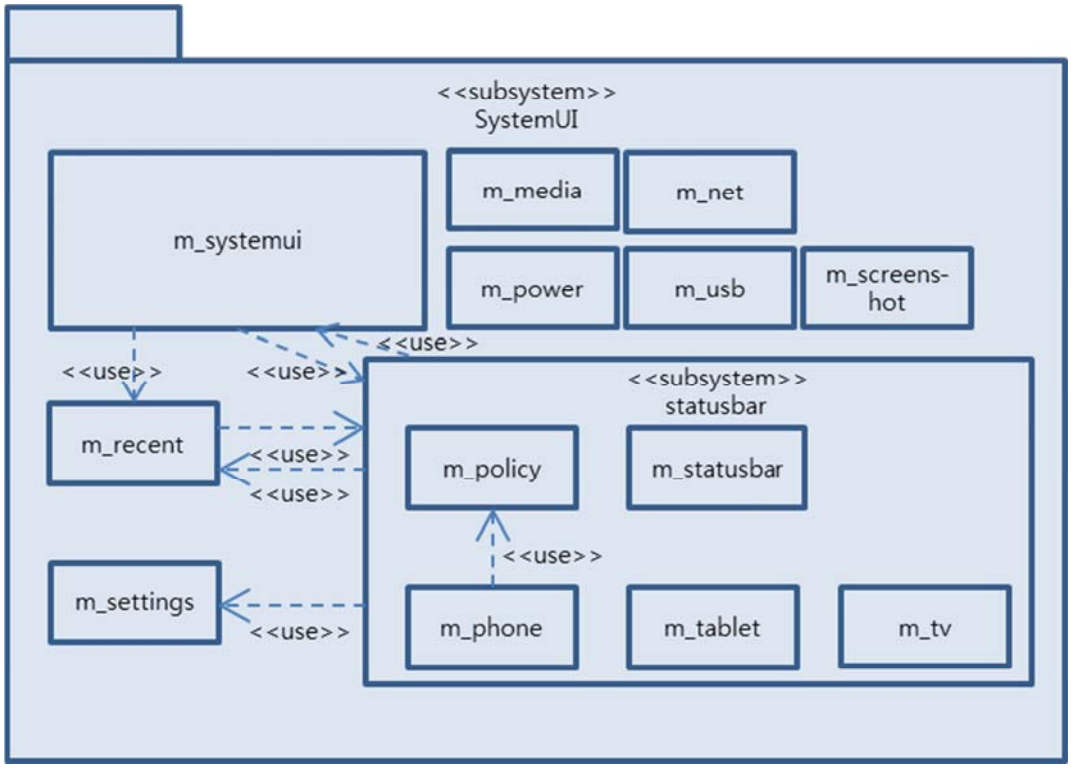

그림 4. AOSP SystemUI 의 모듈 아키텍처

표 2. 화면 밝기 설정 시나리오

시나리오: 화면 밝기 설정	
ID	UC1
행위자	User
절차	1. 상태바를 터치한 후, 화면 아래로 슬라이드 한다. 2. 시스템 정보 Layout 의 우측 상단에 위치한 버튼을 누른다. 3. 컨트롤러를 조작하여 화면 밝기를 조절한다.

4.2 단계 2: AOSP SystemUI 실행 요소 추출

본 사례연구에서는 Android Virtual Device[2]와 Android DDMS[1], 로그 삽입 도구를 이용하여 단계 1 에서 선택한 사용 시나리오에 대한 객체 식별자와 객체를 생성한 클래스 명, 객체간의 함수 호출정보를 추출하였다. Android Virtual Device 는 AOSP 를 실행하기 위한 에뮬레이터로, 선택한 시나리오를 수행하기 위해 사용되었다. Android DDMS 는 에뮬레이터의 실행 정보를 추출하는 도구이다. 로그 삽입 도구는 모든 클래스의 생성자에 로그를 삽입하여 객체가 생성될 때 로그를 출력하는 도구로, 객체의 식별자와 객체를 생성한 클래스 정보 추출해 사용되었다.

4.3 단계 3: 컴포넌트 도출

그림 5 는 단계 2 에서 추출한 객체들과 모듈 아키텍처를 구성하는 모듈들을 연결시킴으로써 그룹화한 결과이다. 총 6 개의 그룹이 생성되고 이들은 각각 컴포넌트로 도출된다. 예를 들어, LocationController, BatteryController, NetworkController 객체는 Policy 모듈과 대응된다. 따라서 세 객체들은 Policy 모듈의 기능을 수행하는 컴포넌트라고 볼 수 있다.

4.4 단계 4: 커넥터 도출

이 단계에서는 단계 2 에서 추출한 함수 호출 관계를 이용하여 단계 3 에서 도출한 컴포넌트들 중 Taylor 등의 커넥터 정의에 부합하는 컴포넌트들을 선택하여 실행 아키텍처의 커넥터를 정의한다.

그림 5 의 컴포넌트들간의 함수 호출 관계를 관찰했을 때, Policy 컴포넌트는 항상 quickSettingModel 객체를 통해 Phone 컴포넌트의 데이터와 기능을 사용한다. 이는 두 컴포넌트는 quickSettingModel 객체에 의해 서로 직접적으로 접근할 수 없기 때문이다. 따라서 quickSettingModel 을 Linkage 커넥터로 정의할 수 있다. Phone 컴포넌트는 Statusbar 컴포넌트에 사용자의 입력을 Event 객체를 통해 전달하므로 Event 커넥터로 정의할 수 있다. 또한, Settings 컴포넌트의 데이터에 접근하기 위해 Intent 객체를 사용한다. Intent 객체는 데이터를 담아서 전달하기 위해 사용되기 때문에 Data access 커넥터로 정의 할 수 있다. 그림 6 은 단계 3 과 단계 4 에서 도출한 컴포넌트와 커넥터를 결합한 결과이다.

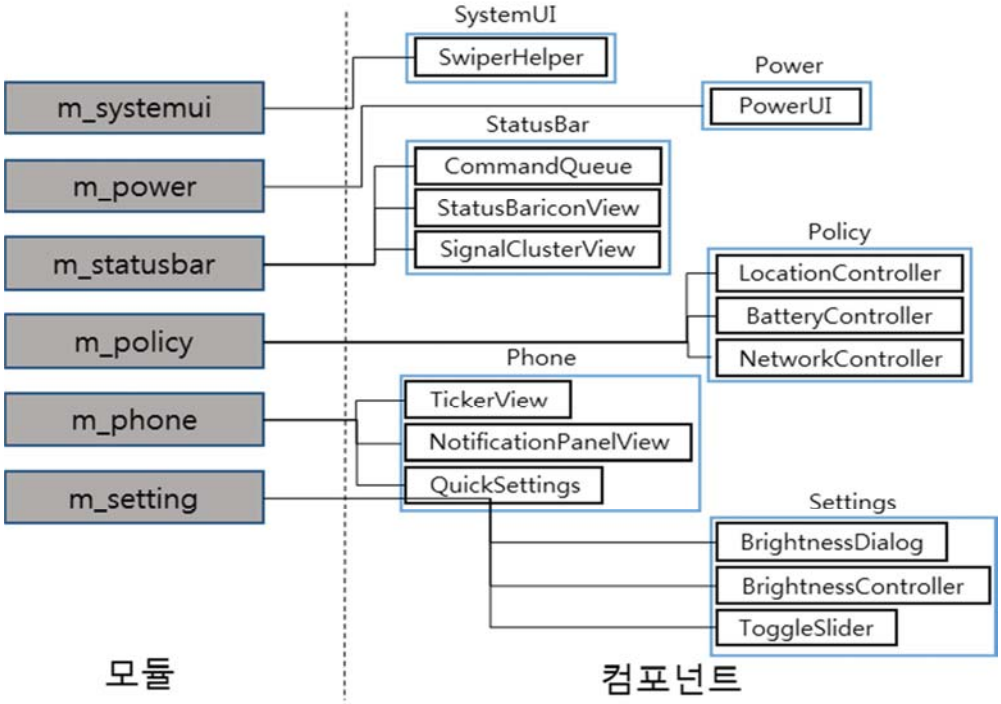

그림 5. 모듈과 객체들간의 대응관계를 통한 컴포넌트 도출

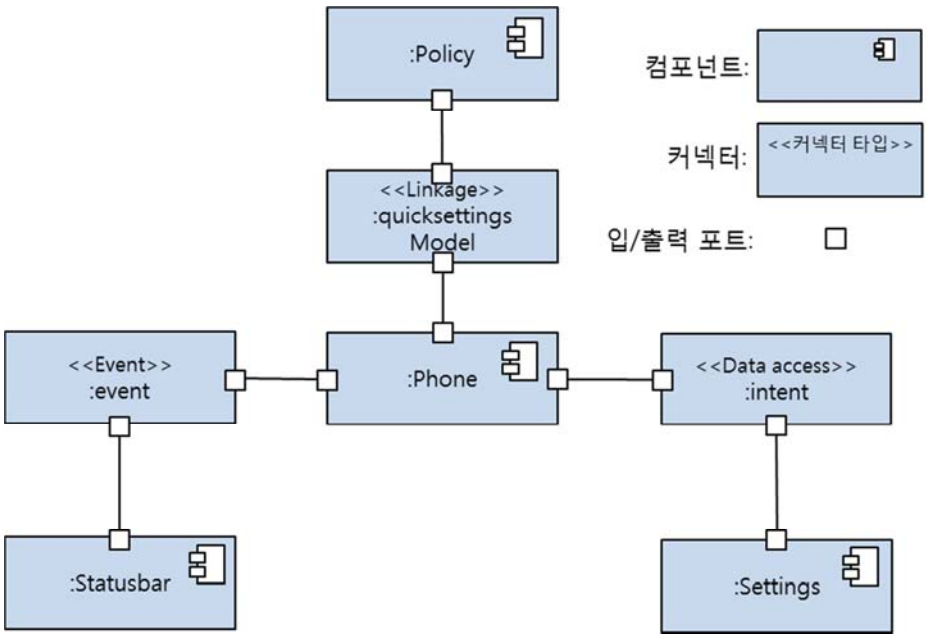

그림 6. 화면 밝기 설정 시나리오에 대한 SystemUI 의 부분 실행 아키텍처

5. 사례연구에 대한 논의

사례연구는 제안 방법에 의하여 도출된 부분 아키텍처가 사용 시나리오 관점의 분석에 효과적임을 보여준다.

그 이유는 첫째, 주어진 시나리오에 대한 부분 아키텍처는 그 시나리오와 관련되지 않은 부분을 나타내지 않기 때문에 기능과 품질의 분석을 용이하게 만든다. 예를 들어, 사례연구를 통해 화면 밝기를 설정하는 기능은 네 가지 컴포넌트와 세 가지 커넥터가 관여하고 이들의 상호작용을 통해 안드로이드 상태바에 밝기 설정 결과를 출력한다고 분석할 수 있다. 그리고 네 컴포넌트는 서로 직접적으로 접근하지 않고 event, intent 와 같은 매개체를 통해 접근하므로 한 컴포넌트의 변경이 다른 컴포넌트에 영향을 미치지 않도록 설계되었다고 볼 수 있기 때문에 변경용이성(modifiability)과 유지보수성(maintainability)이 좋다고 평가할 수 있다.

둘째, 도출과정에 전문가의 개입을 최소화함으로써 높은 정확성을 갖는 아키텍처가 도출된다. 제 3 절에서 언급했듯이, 부분 아키텍처를 도출하기 위해 필요한 입력물들은 사람의 개입 없이 얻을 수 있는 것들이고, 제안 방법의 단계들은 최소한의 사람의 개입만을 필요로 하기 때문에, 사람의 잘못된 판단이 반영되는 위험이 줄어들고, 높은 정확성을 갖는 아키텍처의 도출이 가능해 진다.

6. 결 론

본 논문은 시나리오 관점의 분석을 위한 부분 실행 아키텍처를 도출하는 방법을 제안하였다. 제안 방법은 시스템이 사용 시나리오를 수행하면서 발생하는 실행 요소를 수집하고 이들을 활용하여 컴포넌트와 커넥터를 정의함으로써 부분 실행 아키텍처를 도출한다. 본 논문의 제안 방법은 주어진 시나리오와 관련된 컴포넌트와 커넥터만을 포함함으로써 시나리오 관점의 분석을 용이하게 만든다.

사례연구에서는 AOSP SystemUI 서브시스템을 대상으로 화면 밝기 설정에 대한 부분 실행 아키텍처를 도출하였다. 이를 통해, 시스템의 기능이 올바르게 수행되고 요구되는 품질을 확보했는지 평가하기 위해 부분 아키텍처를 활용하는 것이 효과적임을 보였다. 또한, 사례연구에 대한 논의에서 언급한 바와 같이, 사용 시점에서 주어진 시나리오에 대한 시스템의 구조를 분석하는데 도움을 줄 뿐만 아니라, 가용성(availability), 성능(performance), 보안성(security), 변경용이성(modifiability) 등과 같은 실행 측면의 시스템 품질의 평가에도 활용될 수 있다. 더불어, 사람의 개입 없이 얻을 수 있는 입력물과 부분 실행 아키텍처를 도출하는 구체적인 가이드라인을 통해 높은 정확성을 갖는 부분 실행 아키텍처를 도출할 수 있다.

추후 연구에서는 다양한 추상화 수준의 실행 아키텍처를 재구축하는 방법을 연구할 계획이다. 또한, 사람의 개입을 완전히 배제하면서 커넥터를 도출하는 자동화된 방법을 연구할 계획이다.

7. Acknowledgement

이 논문은 2013 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2013R1A1A2060586)

참고문헌

[1] Android DDMS, http://developer.android.com/tools/debugging/ddms.html.
[2] Android Virtual Device, https://developer.android.com/tools/devices/index.html.
[3] A. Van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva, "Symphony: View-driven software architecture reconstruction," *in Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on, 2004*, pp. 122-132.
[4] B. Schmerl, J. Aldrich, D. Garlan, R. Kazman, and H. Yan, “Discovering Architectures from Running Systems,” *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 454–466, Jul. 2006.
[5] C. Stoermer, L. O'Brien, and C. Verhoef, "Moving towards quality attribute driven software architecture reconstruction," *in 2013 20th Working Conference on Reverse Engineering (WCRE)*, 2003, pp. 46-46.
[6] Enterprise Architect, http://www.sparxsystems.com.au/products/ea/11/.
[7] N. Medvidovic and V. Jakobac, "Using software evolution to focus architectural recovery," *Automated Software Engineering*, vol. 13, pp. 225-256, 2006.
[8] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, R. Little, *Documenting software architectures: views and beyond: Pearson Education*, 2002.
[9] Rational Rhapsody, http://www-03.ibm.com/software/products/ko/ratirhaparchforsoft.
[10] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software architecture: foundations, theory, and practice*: Wiley Publishing, 2009.
[11] S. Kang, S. Lee, and D. Lee, "A framework for tool-based software architecture reconstruction," *International Journal of Software Engineering and Knowledge Engineering*, vol. 19, pp. 283-305, 2009.
[12] StarUML, http://staruml.io/.