韓國IT서비스學會誌
第10卷 第3號
2011年 9月

모바일 앱 개발, 배포, 관리 및 운영을 위한 통합 플랫폼*

오상헌** · 천두완*** · 김수동****

Integrated Platform to Develop, Deploy, Manage, and Operate Mobile Application*

Sang-Hun Oh** · Du-Wan Cheun*** · Soo-Dong Kim****

■ Abstract ■

Mobile devices are widely accepted as a convenient machine which provides computing capability as well we cell phone capability. Because of limited resources on mobile devices, complex applications could not be deployed on the devices. Service-based mobile applications (SMAs) can provide a solution to overcome the limitation by subscribing cloud services. Since SMAs have complex structures than standalone applications, it is challenging to develop high quality SMAs, to manage both services and mobile applications, and to implement automated billing for subscribed services. Therefore, there is a great demand for a platform for super mobile computing, which supports all key activities in managing life cycle of SMAs. In this paper, we present technical aspects of a platform which is under development; Super Mobile Autonomous Reliable plaTform (SMART). We believe that it provides a number of practical features which are essential in supporting life-cycle of SMAs; development, deployment, management, and operation.

Keyword : Mobile Computing, Service-based Mobile Application, Mobile Platform, Development, Deployment, Management, Operation

논문투고일 : 2011년 04월 22일 논문수정완료일 : 2011년 06월 10일 논문게재확정일 : 2011년 09월 02일

* 이 논문은 정보통신산업진흥원의 SW공학 요소기술 연구개발사업과 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원(No. 2009-0076392)을 받아 수행된 연구임.

** 숭실대학교 컴퓨터학과 박사과정

*** 숭실대학교 컴퓨터학과 박사과정, 교신저자

**** 숭실대학교 컴퓨터학부 교수

1. 서 론

모바일 장치들은 커뮤니케이션 장치 뿐만 아니라 컴퓨팅 장치로 최근 각광을 받고 있다[10]. 그러나 모바일 장치가 가진 핵심 문제는 크기가 작은 요인 때문에 자원에 한계가 있다는 점이다[18]. 따라서 복잡한 로직과 대용량의 자원을 소비하는 어플리케이션들은 개발하기 어렵다. 이러한 한계를 극복하고 모바일 장치의 최대 장점을 활용하기 위해 SMAs(Service-based Mobile Applications)가 최근 이슈화 되고 있다[18, 20].

SMAs는 등록된 서비스를 사용하며, 서비스 사용자들이나 어플리케이션은 블랙박스 형태와 소유권 때문에 관리가 어렵다[11]. 또한 기존의 모바일 플랫폼은 모바일 오피스와 같이 기업 업무환경을 모바일 화하기 위해 MEAP(Mobile Enterprise Application Platform)영역에 국한되어 고객용 모바일 앱을 개발하고자 하는 기업들은 많은 시간과 비용, 유지의 어려움이 있다[7, 9, 14, 27, 29]. 그러므로 고품질의 모바일 어플리케이션을 기존의 웹 기술만으로도 쉽고 빠르게 개발할 수 있는 통합 개발환경, 한 번의 개발로 다양한 모바일 운영체제와 다양한 모바일 기기에 적용 가능한 실행환경, 기업의 기간 시스템과 모바일 연계와 산업별 특화 기능 적용이 용이한 서비스 플랫폼을 제공하기 위한 통합 플랫폼이 필요하다.

현재 유용한 통합 플랫폼의 경우[9, 14], 개발, 배치 및 배포, 관리, 운영을 모두 통합한 플랫폼이 아니라 몇 개의 기능만을 제공하고 있기 때문에, 개발된 모바일 어플리케이션 및 서비스의 생명주기를 관리하기가 어렵다. 개발 플랫폼의 경우[1], 독립적인 모바일 어플리케이션 개발을 위해 IDE에 초점을 두고 있다. 그러나 실제 필요한 것은 독립적인 모바일 어플리케이션뿐 아니라 재사용 가능한 서비스 모바일 어플리케이션 설계 및 개발을 지원하는 방법론이 필요하다. 배치 및 배포 플랫폼의 경우 모바일 어플리케이션 및 서비스의 배치, 배포를 위한 저장소를 데이터 센터로 관리하고 있으나, 중앙에서 통제 및 관리하고 있는 구조로 되어 있어서 가용성(Availability)의 한계점을 갖고 있다. 관리 플랫폼의 경우[2] 품질을 실시간으로 관리하지 못하고, 에러 등이 발생하면 수정하여 업데이트하는 방식을 취하고 있다. 또한 서비스를 위한 플랫폼의 경우, 문제 발생 시 개발자가 수작업으로 수정하는 방식을 취하고 있다. 그러나 모바일 어플리케이션 및 서비스가 사용되는 양이 많기 때문에, 한계가 있다. 마지막으로 운영 플랫폼의 경우 사용자의 피드백을 기반으로 평가하는 방안을 제안하고 있다. 그러나 평가된 결과를 기반으로 서비스나 모바일 어플리케이션을 업데이트할 수 있는 전산 처리된 자료를 만드는 데에는 부족함을 갖고 있다.

위의 문제점들을 해결하기 위해 본 논문에서는 개발, 배치 및 배포, 관리, 운영을 지원하는 모바일 컴퓨팅 플랫폼을 제안하며, 이를 SMART라 한다. SMART는 자원의 한계를 해결할 수 있으며, 최적화된 자원을 소비하는 SMAs와 복합적인 어플리케이션의 개발을 지원한다. 개발 플랫폼은 기존의 플랫폼과 달리 방법론 도구가 클라우드 서비스로 제공되며, 사용자 중심으로 사용자 인터페이스를 제공한다. 배치 및 배포 플랫폼은 기존의 플랫폼보다 효율적으로 저장소 공간을 활용할 수 있는 기법을 포함한다. 관리 플랫폼은 기존의 플랫폼보다 자원을 효율적으로 사용할 수 있는 방법을 제공하며, 마지막으로 운영 플랫폼은 기존의 플랫폼보다 추상화 단계가 높은 정보인 사용자 피드백을 활용하는 기법을 제공한다.

2. 관련 연구

MEAPs와 같은 모바일 엔터프라이즈 어플리케이션 플랫폼은 많이 존재한다[11]. 그러나 플랫폼의 초점에 따라 기술이 다양하게 제공된다. 현재 플랫폼은 모델기반의 코드생성에 초점을 맞춘 어플리케이션 개발과 설계를 지원한다. Greenfiled의 연구[7]와 Gronmo의 연구[12]는 웹 서비스를 위한

모델 중심의 개발을 이용하였다. 이들 연구에서는 코드를 플랫폼 종속적 모델로 변환하여 설계하였다. 따라서 디자인모델과 평가모델이 요구된다. IBM은 모든 중요 활동을 지원하는 관리 플랫폼에 대해서 연구를 하였다[21]. IBM은 비즈니스 서비스들의 관리를 단순화하고 서비스 관리 아키텍처 디자인을 자동화 하였다. 이것은 시스템 수준의 관리를 제공하는 디자인이다. 그러므로 서비스 사용자 관점에서 품질측면을 관리하는데 한계를 가진다.

애플의 플랫폼은 가입을 통해 모바일 어플리케이션 개발뿐만 아니라 청구서 발행, 지불, 광고 등이 포함된 분산 모바일 어플리케이션을 지원한다. 애플의 플랫폼과 유사한 구글의 플랫폼은 개발과 분포를 지원한다. 그러나 애플의 플랫폼과 비교했을 때 테스트 절차를 제공하지 않는다. 또한 두 플랫폼은 개발 방법론뿐만 아니라 서비스와 어플리케이션을 위한 관리도 제공하지 않는다. 게다가 플랫폼들은 어플리케이션 배포가 대부분이며, 서비스는 아니다. 그러나 SMART는 서비스 기반 모바일 어플리케이션을 위해 한 플랫폼에서 개발, 배치, 배포, 관리, 회계 등 중요한 활동들을 통합하였다. 이를 통해, SMART는 상호활동과 의사소통에 대한 현재 문제들을 해결한다.

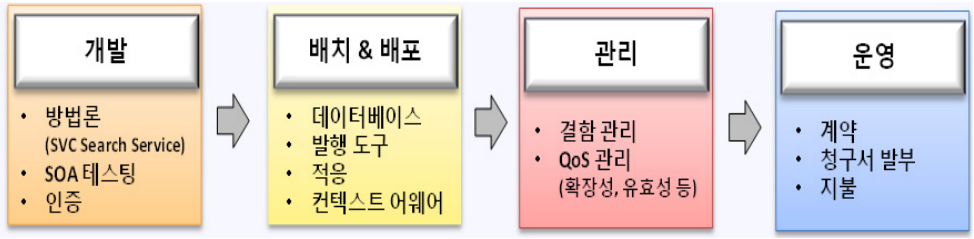
Kranenburg의 연구[19]에서는 적응할 수 있는 커뮤니케이션 서비스들을 제공하고, 내포된 컨텍스트 관리, 개인관리, 인프라스트럭처의 이질성을 감춘 어플리케이션들에 대한 기능을 제공하는 서비스 플랫폼을 제안하였다. 이 서비스 플랫폼은 서비스의 다양성, 이동성, 보안, 비용, 최종 사용자를 위한 편의도구들이 최적의 상태로 제공된다. 그러나 본 연구에서는 컨텍스트 관리 방법론, 평가방법이 부족하다.

Unhelkar의 연구[28]에서는 많은 모바일 어플리케이션을 위해 효과적(많은 사용자들이 실시간으로 정보를 접근 할 수 있도록 제공)으로 운영될 수 있는 모바일 기술과 즉각 반응할 수 있고, 높은 경쟁력과 모바일 혁명, 새로운 사용자들의 요구를 만족시킬 수 있는 엔터프라이즈를 제안하였다. 그러나 본 연구에서는 다양한 모바일 어플리케이션들에 대한 구체적인 아키텍처와 이를 평가할 수 있는 방법들에 대한 연구가 미흡하다.

Najafi의 연구[24]에서는 서비스를 이용하는 클라이언트 측과 일반적인 서비스와 특정 서비스를 커스터마이즈 하여 제공하는 제공자 측 사이에서 이를 조율하는 커스터마이즈 에이전트를 제안한다. 본 연구의 커스터마이즈 에이전트 기능은 클라이언트 컨텍스트를 기반으로 일반적인 서비스를 개인적인 취향에 맞게 가공하여 제공한다. 본 연구에서는 혁신적인 Context-aware 서비스 제공 방법과 향상된 웹 서비스의 보안과 프라이버시에 대한 접근 방법도 제시하였다. 그러나 본 연구에서는 컨텍스트 서비스 제공을 위한 아키텍처를 제공하지만 이에 대한 개발 방법론 및 품질평가 실시간 감시 및 관리에 대한 내용이 미흡하다.

3. SMART의 기능분류

SMART의 주요 기능은 모바일 어플리케이션 및 서비스의 기본적인 생명주기에 해당하는 프로세스를 지원하는 것이다. 그러므로 먼저 모바일 어플리케이션 및 서비스의 기본적인 생명주기에 대해 정의한다. 모바일 어플리케이션 및 서비스는 큰 의미에서 소프트웨어다. ISO 12207 표준에 따르면, 소프트웨어의 기본적인 생명주기 프로세스는 획득, 공급, 개발, 운영, 유지보수로 이루어지고, 지원생명주기는 문서화, 형상관리, 품질보증, 검증, 확인, 합동검토, 문제해결, 감사로 이루어지며, 조직생명주기는 기반구조, 교육훈련, 관리, 개선으로 이루어진다[13]. 이런 5가지 프로세스 중 플랫폼 관점에서 지원 가능한 프로세스를 추출한 다음, 모바일 어플리케이션 및 서비스에 특화되어 정의하면, [그림 1]과 같이 4가지로 요약된다. 즉, 본 장에서는 SMART에서 지원할 기능을 도출하고, 도출된 기능에 대해 상세하게 명세한다.



[그림 1] SMART 전체 프로세스

3.1 개발 지원 기능

SMART에서의 개발 대상은 엔터프라이즈에서 사용 가능한 모바일 어플리케이션과 서비스이다. 즉, 일반적인 모바일 어플리케이션 및 서비스 개발뿐만 아니라, 이보다 더 복잡한 기능을 요구하는 모바일 어플리케이션 및 서비스 개발을 지원한다. 이를 위하여 SMART에서는 다음과 같은 도구와 기술들을 제공한다.

모바일 어플리케이션 개발 방법론 제공 : SMART에서는 모바일 어플리케이션 개발 방법론인 GREEN 모바일을 제공한다. GREEN 모바일은 서비스 기반의 모바일 어플리케이션을 개발하기 위하여 만들어진 개발 방법론으로, 이 방법론에는 프로세스 명세서, 각 단계별 공학적 지침, 품질측정을 위한 지침, 각 단계별 산출물을 만들기 위한 템플릿을 포함한다[25]. 이런 방법론이 필요한 이유는 모바일 어플리케이션의 복잡도 향상 때문이다. 스마트 폰과 같은 모바일 장치의 컴퓨팅 파워 등의 자원이 예전에 비해 향상되면서, 복잡한 계산을 요구하는 모바일 어플리케이션의 개발 역시 요구되고 있다[11]. 그러나 대부분의 모바일 어플리케이션은 모바일 플랫폼에 종속적인 개발 도구를 사용하여 구현되고 있다. 이로 인하여 실제 모바일 어플리케이션을 개발 및 사용에 많은 문제가 발생하고 있다. 대표적인 모바일 어플리케이션 마켓인 앱스토어에 배포되어 있는 모바일 어플리케이션들에 대한 사용자들의 평가를 보면, 많은 수의 모바일 어플리케이션이 안정적으로 개발되지 않아서 불편함을 겪는 경우를 많이 볼 수 있다. 또한, 모바일 어플리케이션 배포 후에도 문제를 해결하기 위하여 지속적으로 관리해야 하는 경우도 많이 볼 수 있다. 이는 모바일 어플리케이션 개발 시 분석이나 설계 과정이 충분하지 않았기 때문으로 판단할 수 있다. 그러므로 SMART에서는 모바일 어플리케이션을 개발할 수 있도록 방법론을 제공한다. 또한, 플랫폼 차원에서 방법론을 제공하기 위해 본 연구에서는 방법론 매뉴얼을 클라우드 서비스 형태로 제공한다. 클라우드 서비스 형태로 제공하였을 경우, 얻을 수 있는 장점은 방법론 매뉴얼의 접근성 향상과 사용성 향상이다.

클라우드 서비스 개발 방법론 제공 : SMART에서는 서비스 형태의 매쉬업(Mash-up, Component-as-a-Service)개발을 위한 방법론인 GREEN.서비스를 제공한다. GREEN.서비스 역시 방법론이기 때문에, 클라우드 서비스 개발을 위한 체계적인 프로세스를 서비스 개발자에게 제공한다. 여기에는 활동 규정, 지침, 품질측정 가이드라인 그리고 예제 등이 포함된다. 특히, 이 개발 방법론에서 제공하는 공학적 활동은 재사용 서비스 설계 방법 및 서비스 인터페이스 설계, 서비스 등록이다. 이 방법론 역시 SMART에서 플랫폼 차원으로 제공하기 위하여 클라우드 서비스 형태로 제공한다.

모바일 어플리케이션의 경우, SMART에서 제공하는 방법론에 따라 설계하고 검증된 아키텍처를 기반으로 각 모바일 플랫폼에서 제공하는 개발 도구를 이용하여 실행 가능한 모바일 어플리케이션을 개발할 수 있다. 서비스의 경우 또한, SMART에서 제공하는 방법론에 따라 설계한 모델을 기반으로 Eclipse 등을 활용하여 클라우드 서비스를 개

발할 수 있다. SMART에서는 기존에 있는 개발 도구를 배제하는 것이 아니라, 기존에 있는 개발 도구를 체계적으로 활용할 수 있는 방법을 플랫폼 형태로 제공하고 있다.

3.2 배치와 배포지원 기능

SMART는 애플 앱스토어나 구글 안드로이드 마켓의 기능과 유사한 어플리케이션 배포 기능을 제공한다. 그러나 SMART를 이용한 어플리케이션은 특정 플랫폼에 독립적인 형태로 분산되어 운영된다. 기존의 모바일 어플리케이션 배포 플랫폼은 개발자가 오직 하나의 플랫폼에만 해당 어플리케이션을 업로드 할 수 있다. 이러한 문제점들을 개선하기 위해 SMART에서는 다음과 같은 기술과 기능들을 제공한다.

서버 분산 시스템 제공 : SMART의 서버를 분산 아키텍처로 변경 한 후, 여러 개의 서버가 SMART 노드의 그리드를 구성한다. 따라서 어플리케이션을 배포하기 위한 저장소들 간에도 서로 연결되어 있다. 주요 커널 컴포넌트로서, 비공개 P2P 통신 커널은 저장소와 분산된 SMART 노드들 사이에서 정보를 공유하고 관리한다. 이러한 기능을 위해서는 모바일 장치에 SMART 에이전트가 있어야 한다. SMART 에이전트 사이에서는 응답 메시지와 에이전트의 엔드포인트가 포함된 요청 메시지를 주고받으며, 서로를 인식할 수 있는 기능이 있다. 이러한 기능을 통해 SMART 배치와 배포 서브시스템은 높은 가용성을 제공한다.

서비스 배치기능 제공 : SMART는 서비스 배치를 위한 기능들을 제공한다. 또한, SMART에 등록되고 배포되면, SMART의 구독 서브시스템과 관리 서브시스템에서 능률적으로 관리된다. 또한 플랫폼에서는 서비스 제공자들이 서비스를 배치할 수 있도록 인터페이스를 제공한다.

어플리케이션 배포기능 제공 : 어플리케이션 배포기능은 SMART에서 어플리케이션 제공자들이 어플리케이션을 배포하거나 광고 할 수 있도록 인터페이스를 제공한다. 서비스 배치기능과 어플리케이션 배포기능은 기능면에서 비슷하지만 서로 차이가 있다.

서비스 제공자가 서비스를 배치하거나 어플리케이션을 배포할 때, 서비스 배치 모듈은 서비스 저장소에 서비스 설명과 제공자 정보 둘 다 저장한다. 하지만, 어플리케이션 배포 모듈은 어플리케이션 저장소에 이름과 설명 등을 포함하여 배포된 어플리케이션 정보를 저장하고 유지한다. 이 정보는 앱스토어나 안드로이드 마켓과 같은 저장소를 기반으로 정의된다.

3.3 관리 지원 기능

SMART는 배포 및 배치된 모바일 어플리케이션 및 서비스를 자율적(Autonomous)으로 관리할 수 있도록 지원하며, 자율 컴퓨팅은 관리비용에 대한 문제를 효율적으로 해결하고 진보된 개념적 프레임워크를 제공한다[4]. 기존 시스템과 서비스 관리 시스템은 관리자에게 모니터링과 시스템 상태를 보고하는 정도의 기능을 자동화 해주는 도구를 제공했다. 수동적인 관리는 많은 단점을 가지고 있다; 관리 비용과 노력의 증가, 낮은 관리 효율성, 서비스 결함 분석이 어려우며, 정상적인 작동을 지속시키기 위해 원래 상태로 만드는데 걸리는 시간. 서비스 관리를 위해 자율 컴퓨팅에 핵심 규율(discipline)을 적용하면 루틴(Routine) 작업을 자동화 할 수 있고, 많은 문제들을 해결할 수 있다. 이를 위해 자율컴퓨팅의 MAPE 패러다임에 따라 SMART에서는 다음과 같은 기능을 제공한다.

모니터링 기능 제공 : SMART에 배치 및 배포된 모바일 어플리케이션 및 서비스의 상태를 모니터링하고, 모니터 된 정보를 관리한다. 여기서 모바일 어플리케이션 및 서비스의 상태는 각각 다르다. 모바일 어플리케이션의 경우, 사용자의 모바일 장치에서 실행되기 때문에 모바일 어플리케이션의 실시간 상태를 알기는 어렵다. 그러므로 SMART에서는 모바일 어플리케이션의 다운로드 수, 사용

자의 평가 등을 모니터링 한다. 반면, 서비스의 경우는 SMART 플랫폼을 통해서 실행되기 때문에 서비스의 실시간 상태, 즉 서비스의 응답 시간, 서비스 요청 횟수, 서비스의 성공적인 응답 횟수 및 실패한 응답 횟수 등을 모니터링 할 수 있다. 또한, 서비스 사용자의 정책에 따라서 결함과 관련된 증상을 식별할 수도 있다. 여기서 증상은 서비스 사용자의 정책에 맞지 않게 서비스가 동작하지 않아서 발생하는 결과를 의미한다.

분석 기능 제공 : 분석 기능은 주로 증상의 원인을 진단하는 것을 의미한다. 모바일 어플리케이션의 모니터링 결과에 따른 분석 결과는 자율적인 관리에서 활용될 수 없기 때문에 해당사항이 없다. 반면 서비스의 모니터링 결과는 증상 식별의 기반이 된다. 또한, 증상이 식별된 이후에는 이 기능을 통하여 서비스 결함에 대한 원인을 분석한다. 여기서 분석을 위해서는 모델 기반 진단 기법과 확률 기반 진단 기법이 모두 사용될 수 있다[15, 26].

어댑테이션 기능 제공 : 어댑테이션 기능은 주로 분석된 결과에 따라 자율적으로 해결하는 것을 의미한다. 즉, 이 기능에는 원인 해결을 하기 위한 기능 및 해결 방법을 적용하기 위한 기능이 모두 포함된다. 여기에서의 가정은 해결 방법을 제공하는 모듈이 기본적으로 제공되어야 하며, 원인에 따른 해결 방법을 찾기 위한 공유 저장소가 관리되어야 한다. 공유 저장소의 경우, SMART에서 관리하지만, 해결 방법을 제공하는 모든 모듈을 SMART에서 포함하고 있지는 않다. 이런 가정이 만족한다면, 모니터링 된 결과에 따라 증상을 식별하고, 식별된 증상을 분석하여 원인을 찾아낸 다음, 원인에 따른 해결 방법을 자율적으로 적용할 수 있다.

SMART에서는 위의 세 가지 기능이 자율적으로 실행될 수 있도록, 공유 저장소를 관리 및 운영한다. 공유 저장소에는 분석 및 어댑테이션에 필요한 규칙 및 통계 자료 등이 있으며, 자율 관리 기능이 실행될 때마다 혹은 특정 시간마다 관련 규칙 및 통계 자료를 업데이트할 수 있도록 설계된다.

3.4 운영 지원 기능

플랫폼 운영의 중요한 목적은 다양한 개발자들로부터 개발된 서비스나 어플리케이션을 배포하는 절차와 효과적인 서비스 구독, 최소한의 노력으로 개발자들이 쉽게 수익을 얻게 하는데 있다. 즉, SMART는 개인 개발자뿐 아니라 기업에서도 만족할 수 있는 One-stop 비즈니스 시장이다. SMART는 서비스의 온라인 구독, 자동결제 및 지불처리, 서비스 제공자와 자동화된 수익분배 서비스, 서비스 배포 시 광고 등을 지원한다. SMART의 운영 서브시스템은 사람의 노력을 최소화 하여 설계되었다. 서비스가 사용자로부터 접근되거나 요청이 있을 때, SMART는 서비스 구독의 품질평가와 회계 작업에 필요한 활동을 수행한다. 또한, 여러 개의 다른 비즈니스 트랜잭션 보고서를 다양한 이해 당사자들을 위해 생성한다. 따라서 SMART에서는 운영지원을 위해 다음과 같은 기능을 지원한다.

서비스 구독 및 사용량 측정 제공 : SMART에서는 서비스 구독을 위한 활동을 지원한다. 서비스 구독은 요금기반과 광고기반으로 분류된다[22]. 제공되는 서비스는 개발자가 요금을 지정하며, 광고기반의 서비스는 특정 요금 지불 없이 서비스를 이용할 수 있다. 또한 SMART에서는 서비스의 사용량이나 어플리케이션의 다운로드 횟수에 대한 측정 기능을 제공한다. 사용량 측정은 사용자에게 서비스나 어플리케이션을 고품질로 제공하고, 청구서 발행 및 수익분배, 비즈니스 요약 보고서를 생성하는데 기반이 된다.

청구서 발행 및 지불 기능 제공 : SMART에서는 청구서 발행 및 지불에 대한 기능을 지원한다. 청구서 발행은 서비스의 사용량과 어플리케이션 다운로드 횟수를 기반으로 하여 사용자에게 청구된다. 서비스 사용량의 경우 이용 빈도에 따라 청구되며, 어플리케이션의 경우는 한 번의 다운로드 시에만 청구된다. 즉, 같은 어플리케이션을 다시 다운로드 해도 비용을 지불할 필요가 없다. 따라서 청구서가 발행되면 사용자는 현금과 신용카드를

이용하여 지불한다. 현금 지불 시에는 은행에서 제공하는 전자송금을 이용하고, 신용카드 이용 시에는 다른 외부 회사에서 제공하는 결제 시스템을 활용한다.

자동 수익분배 및 비즈니스 요약 보고서 제공 : 제공자의 서비스나 어플리케이션이 판매되면, 제공자와 SMART 플랫폼 매니저와 수익을 자동으로 분배한다. 분배에 대한 비율은 사전에 SMART 플랫폼 매니저와 협의 하에 이루어진다. SMART는 일정 기간 동안 비즈니스와 관련된 정보를 요약하여 보고서를 생성한다. 보고서는 3가지 타입(최종사용자, 개발자, 플랫폼 관리자)으로 각각 다르게 생성된다.

4. SMART 설계 및 기법

4.1 서비스 기반 모바일 어플리케이션 설계 기법 및 이를 지원하는 클라우드 서비스

본 절에서는 SMART에서 지원하는 개발 단계의 주요 활동과 이를 지원하는 클라우드 서비스에 대해 다룬다.

서비스 기반 모바일 어플리케이션 설계는 크게 재사용 가능한 서비스의 설계와 이를 기반으로 한 모바일 어플리케이션 설계로 나뉜다. 먼저, 재사용 가능한 서비스를 설계하기 위한 주요 활동은 다음과 같다.

1) 공통성 및 가변성 분석 활동은 재사용 가능한 서비스의 기능성을 식별하는 단계이다. 먼저, 도메인 요구사항 및 잠재 요구사항을 기반으로 공통성을 식별한다. 이를 위한 식별 기법은 이미 제품 계열 공학과 관련한 연구에서 다루고 있다[16, 17]. 그런 다음, 식별된 공통성을 기반으로 가변성을 분석한다. 가변성은 공통성 중에 각 요구사항보다 다른 일부의 다른 점으로써, 재사용성을 향상시키기 위한 주요 요소이다. 서비스와 관련한 가변성 유형에는 대표적으로 복합 서비스에서의 워크플로우 가변성, 단일 서비스 컴포넌트 가변성 등이 있다[5].

2) 서비스 설계 활동은 이런 분석 결과를 활용하여 서비스의 기능성을 구현하기 위한 방법을 정의하는 단계이다. 설계하는 방법은 기존 객체지향 설계 방법론에서 다루고 있는 내용과 크게 다르지 않다. 그러나 서비스에서는 기존과 다르게 추가적인 요소들이 있기 때문에 이를 위한 스테레오 타입을 정의하여 사용한다. 예를 들어 서비스는 <service>라는 스테레오타입을 기존 클래스에 추가 명세함으로써 서비스임을 모델링할 수 있다.

3) 인터페이스 명세는 설계된 서비스를 사용할 수 있도록 기능성 및 비기능성, 사용 시 제약사항 등을 명세하는 단계이다. 이 활동의 경우는 기존에 있었던 방법론과 다르다. 첫 번째 이유는 서비스의 경우 표준화되어 있는 인터페이스 명세서 기술 양식을 따르기 때문이다. 대표적으로 사용하는 인터페이스 표준에는 WSDL(Web Service Description Language), REST(Representational Stateless Transfer)가 있다.

두 번째 이유는 서비스 인터페이스 명세 시 서비스를 접근할 수 있는 바인딩 정보를 추가 기술하여야 하기 때문이다. 즉, 서비스는 인터넷을 통한 기능 호출을 가정하고 있기 때문에 서비스 주소, 서비스 메시지 전송 시 사용할 프로토콜 등을 추가적으로 명세하여야 한다.

이렇게 설계된 서비스를 기반으로 모바일 어플리케이션을 설계하기 위하여 필요한 주요 활동은 다음과 같다. 1) 서비스 기반 모바일 어플리케이션 아키텍처 설계의 경우, 전체 구현하여야 할 기능의 기능성을 나누는 것부터 시작한다. 예를 들어, 위치 기반 모바일 어플리케이션을 설계한다고 하자. 이 어플리케이션의 주요 기능은 모바일 디바이스의 센서를 활용하여 GPS 센서 정보를 가져오는 기능, GPS 센서 정보를 활용하여 위치 정보로 변환하는 기능, 변환된 위치 정보를 지도에 매핑하는 기능으로 나뉜다. 앞에 두 개의 기능이 모바일 어플리케이션에서 지원하고, 남은 하나의 기능이 서비스에서 지원할 수 있다. 또한 처음 하나의

기능만 모바일 어플리케이션에서 지원하고 남은 두 개의 기능이 서비스에서 지원될 수도 있다. 이처럼 전체 기능 중 어떤 기능을 모바일 어플리케이션 혹은 서비스에서 구현할지 먼저 결정한다. 그런 다음, 모바일 어플리케이션과 서비스의 상호 연동 방법을 결정한다. 모든 상호 연동은 크게 동기적 통신과 비동기적 통신으로 나뉜다. 동기적 통신은 모바일 어플리케이션과 서비스가 주고받는 데이터가 개인적이고 보안이 요구되는 경우에 주로 사용된다. 비동기적 통신은 관련 기능을 처리하기 위하여 일회성 데이터를 주고받는 경우에 사용된다.

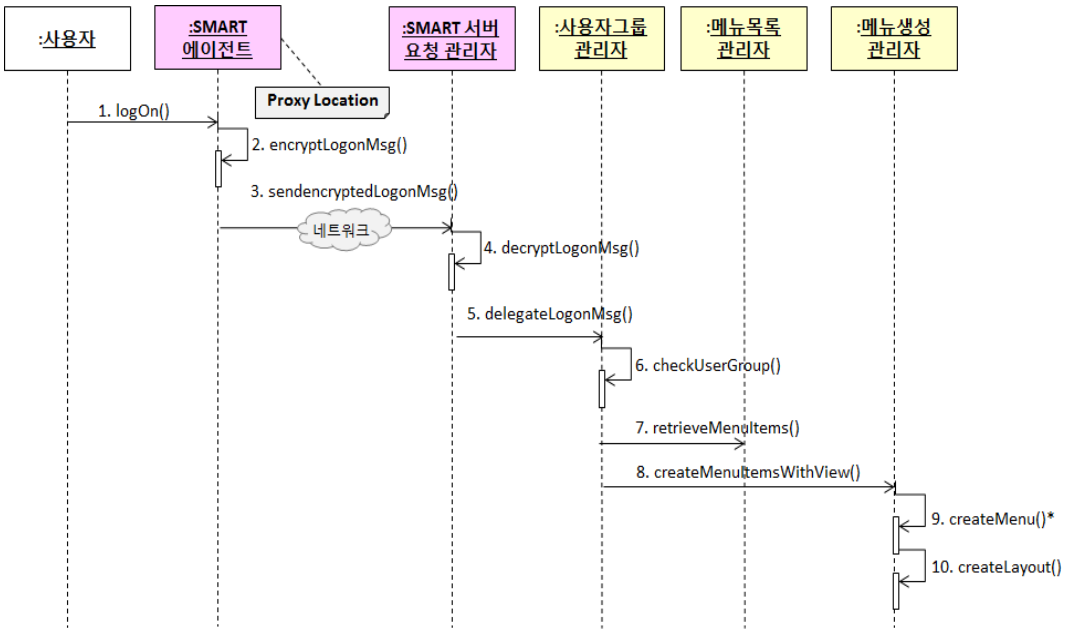
이를 기반으로 모바일 어플리케이션의 기능성을 위한 2) 컴포넌트를 설계한다. 설계하는 방법은 기존에 객체지향 설계 방법론에서 다루고 있는 내용과 크게 다르지 않다. 단, 모바일 어플리케이션은 모바일 디바이스에서 실행되기 때문에 모바일 장치의 자원 제약사항(메모리, CPU 등), 네트워크 제약 사항(사용하는 망 및 이에 따른 프로토콜) 등을 고려하여 최적화할 수 있도록 설계하여야 한다. 서비스의 경우는 이미 설계되어 있고, 제공되는 것이므로 필요한 서비스가 있는지 확인하는 작업으로 완료될 수 있다.

그러나 이런 활동에 대한 기술은 문서로 제공되는 경우가 많아서 일반적인 개발자들이 관련 문서를 이해하여 설계하기에 어려움이 존재한다. 이런 어려움을 해결하기 위하여 등록된 사용자의 분류에 따라 능동적으로 변경되는 사용자 인터페이스(U.I, User Interface)를 제공한다. 이는 다음과 같이 수행된다.

1. 메뉴 목록 관리자 컴포넌트는 능동적으로 변경되는 사용자 인터페이스를 제공하기 위하여서는 클라우드 방법론 서비스에서 제공하여야 하는 모든 메뉴를 식별하고, 식별된 메뉴를 관리한다.
2. 메뉴 생성 관리자 컴포넌트가 동적으로 필요한 메뉴 항목들의 생성을 요청 받을 때, 실시간으로 메뉴 항목을 구성한다. 여기서 중요한 것은 요청이 들어올 때마다 메뉴를 생성하는 것이 아니라, 이미 만들어져 있는 메뉴 항목을 보일 수 있도록 설정한다는 것이다.
3. 필요한 모든 메뉴 항목을 보일 수 있도록 설정한 다음에는 메뉴를 구성하는 레이아웃을 변경하여야 한다. 우리는 HTML5에서 제공하는 기술을 활용하여, 메뉴 아이템의 개수에 따라 자동으로 크기를 변경해주는 방법을 사용한다.

본 연구에서는 위와 같은 기능을 방법론 도구에서 제공하며, 방법론 도구의 접근성 향상을 위하여 SaaS 형태의 클라우드 서비스 형태로 제공한다. 예를 들어, 방법론 검색을 위해서는 http://{Endpoint of deployment}/smart/metho-dology/search? = ”keyword”와 같은 URL 기반의 API를 사용하게 된다. 예제에서 사용한 ‘Endpoint of deployment’는 SMART에서 제공하는 방법론이 배치될 URI를 의미한다. 그렇지만, 이런 분류된 기능에 따라 각 사용자가 필요한 기능과 방법론의 내용만을 보여주기 위해서는 별도의 인증 기법이 필요하다. 일반적으로 클라우드 서비스의 경우, 브라우저에서 접근할 수 있는 URL 기반의 API로 제공되는 경우가 많기 때문에 관련 기법이 없는 경우, URL을 통하여 불법적으로 클라우드 서비스에 접근하는 것이 가능할 수 있다. 앞에서 언급한 바와 같은 URL 형식은 인증을 관리하는 모듈의 처리가 되지 않은 것이어서, 누구나 접근이 허용한 형태이다. 그러므로 여기에 보안성을 강화하기 위하여 다음과 같은 인증기법을 수행한다.

1. 클라이언트 프로그램에서 먼저 SMART 에이전트로 로그온 관련 요청을 보낸다. 관련 요청을 보낼 때 proxy로 구성된 URL API로 접근하여 클라이언트에서는 요청하게 된다.
2. SMART 에이전트에서는 관련 요청을 받아서 케르베로스, 인증키 등의 방식을 활용한 암호화 알고리즘을 적용한 후, SMART 서버에 요청 메시지를 보낸다.
3. SMART 서버에서는 관련 요청 메시지에서 사용자 정보를 이용하여 사용자가 접근 가능한 메뉴 목록을 식별하여 관련 기능을 수행한다.

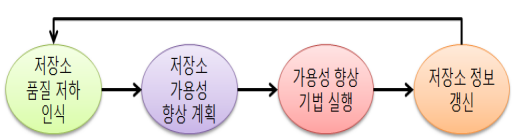


[그림 2] 사용자 중심의 클라우드 방법론 도구 동작을 위한 시퀀스 다이어그램

즉, 인증 기법을 적용한 능동형 사용자 인터페이스를 제공하기 위한 전체 흐름은 [그림 2]와 같다.

4.2 배치, 배포를 위한 저장소 가용성 관리 기법

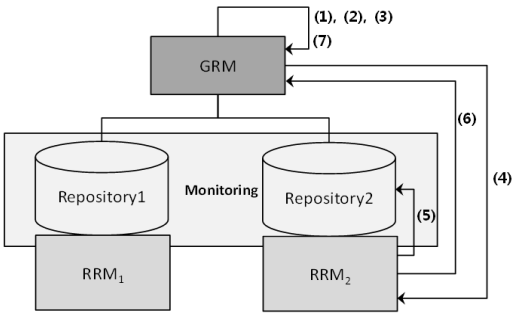
일반적으로 저장소에 한 번에 접근할 수 있는 요청의 수는 제한적이다. 이로 인해 요청의 수가 많아지는 경우, 저장소의 가용성이 저하되는 문제가 발생할 수 있다. 또한, 이 요청은 저장소를 쓰거나 업데이트하는 요청과 검색을 위한 요청으로 나뉘는데, 이로 인하여 저장소의 무결성이 훼손될 수 있다. SMART 환경에서는 여러 개의 저장소가 이미 존재하고 있고, 네트워크로 연결되어 있다. 그러므로 이런 저장소를 활용하여 가용성을 향상시킬 수 있으며, 이를 위한 기법을 본 절에서는 제안한다. 저장소의 가용성 관리 기법의 절차는 다음 [그림 3]과 같이 네 단계로 이루어진다.



[그림 3] 저장소 가용성 관리 기법 절차

'저장소 품질 저하 인식' 단계에서는 저장소가 일정 수준의 품질을 유지하고 있는지 모니터링하고, 유지하지 않는 경우 이를 문제로 식별한다. 여기서 말하는 품질은 가용성과 관련된 품질 요소를 의미하는데, 대표적으로는 저장소의 데이터 처리량, 저장소의 응답 시간 등이 이에 해당한다. '저장소 가용성 향상 계획' 단계에서는 저장소의 가용성을 저하의 유형을 분석하고, 분석된 유형에 따라 이를 해결하기 위한 기법과 실행 계획을 수립한다. 이 단계를 위해서는 자주 발생하는 문제와 이에 대한 해결책을 관리하고 있는 지식베이스 및 추론 엔진이 필요하다. '저장소 가용성 향상 기법 실행' 단계에서는 이전 단계에서 수립된 계획에 따라 이주 기법을 실행한다. 소프트웨어 가용성 향상 기법에는 일반적으로 복제 혹은 이주기법이 있으나, 저장소를 복제하는 기법은 주로 저장소의 신뢰도를 향상시키기 위한 방법으로 더 자주 활용

되기 때문에 본 논문에서는 이주 기법만을 다룬다. '저장소 정보 갱신' 단계에서는 실행된 가용성 향상 기법에 따라 변경된 저장소에 대한 정보를 갱신한다. 이주기법은 특정 저장소에 물리적인 문제가 발생하여 더 이상 서비스를 제공하기 힘들 때, 해당 저장소에 배치된 서비스나 어플리케이션을 새로운 저장소로 이동시켜 동일한 역할을 수행하도록 하는 활동이다.



[그림 4] 배치, 배포를 위한 저장소 가용성 향상 기법의 기능 흐름

[그림 4]는 저장소 가용성 향상 기법 중 이주 기법의 기능흐름을 보여준다. 이 기능 흐름에는 다음 두 개의 컴포넌트가 관여한다.

- GRM(Global Repository Manager) : 저장소에 등록되거나 업로드 된 모든 서비스와 어플리케이션을 통합하여 관리하고 실시간 상태를 고려하여 저장소의 계획을 수립.
- RRM(Regional Repository Manager) : 해당되는 저장소의 자원 상황을 감시하고, 수집된 정보를 GRM으로 전달하고, 저장소의 계획을 수행.

이주 기법의 기능흐름은 다음과 같다. 먼저, GRM이 새롭게 저장될 서비스와 어플리케이션을 위해 저장소 상황과 위치 등에 대한 정보를 확인한다. 정보가 확인되면 테이블의 정보를 갱신한다. 자원량을 기준으로 후보 저장소를 검색하여, 결정된 RRM2에게 서비스 및 어플리케이션 정보를 전달한다. RRM2는 전달받은 서비스와 어플리케이션을Repository2에 배치하고 배치결과를 GRM에게 전달한다. 배치결과에 따라 GRM은 Repository2를 분산 테이블 정보에 추가하고, 서비스와 어플리케이션에 관련된 모든 가용한 처리량 비율을 고려하여

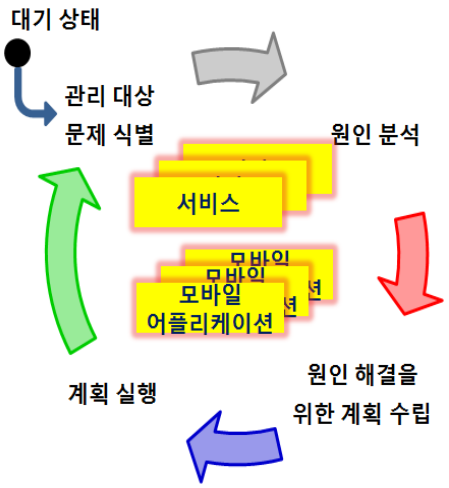
〈표 1〉 배치, 배포를 위한 저장소 가용성 관리 기법 알고리즘

```
1 : Algorithm ServiceMigration
2 : Input : System Status
3 : Output : null
4 : CheckStatusandLocation(CurrentStatus, CurrentLocation)
5 : // 현재 저장소의 상황, 위치 등에 대한 정보를 확인
6 : RepositoryFromList(RepositoryAddress) {
7 : Retrieve RepositoryList, TotalAvailableNumberOfRequests
8 : Set TotalAvailableNumberOfRequests to 0
9 : Remove Repository n with same RepositoryAddress from RepositoryList
10 : for each Repository n_i from RepositoryList
11 : Add AvailableNumberOfRequests of n_i into TotalAvailableNumberOfRequests
12 : end for
13 : return
14 : FindCandidateRepositorys(RequiredThroughtput)
15 : // 저장 될 저장소의 후보 목록을 반환한다.
16 : DeployService(SVCPackage, ApplicationPackage)
17 : // 저장소에 서비스와 어플리케이션을 배치, 배포하고 결과를 ReplicationResult에 저장한다.
18 : Result(Result, RepositoryAddress)
19 : // 저장 결과t와 해당 저장소의 주소 RepositoryAddress를 GRM으로 전달한다.
20 : UpdateLoadBalancingTable(Result, RepositoryAddress)
21 : // 각 저장소의 자원량의 비율을 기준으로 상대적인 부하 할당량을 부여
```

분산 비율을 다시 설정한다. <표 1>은 이주 기법의 기능 흐름을 지원하는 에이전트의 알고리즘이다.

4.3 자율 관리 생명주기를 적용한 에이전트 기반 관리 기법

SMART 환경에서는 각 어플리케이션과 서비스가 구동되는 노드에 에이전트가 존재한다. 그리고 에이전트와 관리를 위한 서버는 [그림 5]와 같은 자율 관리 생명주기에 따라 실행된다.



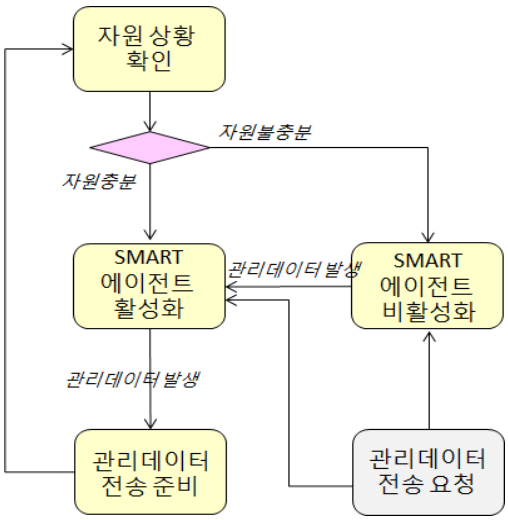
[그림 5] 자율 관리 생명주기를 적용한 SMART 관리 컴퓨팅 모델

먼저, 에이전트는 백그라운드에서 실행되면서 필요한 관리 정보를 수집하고, 이런 관리 정보를 추론할 수 있는 서버에 보내는 역할을 한다. 그러면 서버에서 관련된 결과를 추론하여 에이전트에 보내주고 이에 따라 관련 관리를 수행하거나, 서버에서 분석된 결과를 갖고 관리자가 관리를 수행하게 된다. 그러나 모바일 장치의 경우 이런 에이전트가 지속적으로 백그라운드에서 실행되게 되면 자원 부족 현상이 발생할 수 있어서 오히려 문제를 발생시킨다. 그러므로 이를 위하여 에이전트의 실행 패턴을 설계한다.

먼저 에이전트의 실행 패턴은 크게 두 가지로 나뉜다. 첫째는 능동형 실행 패턴으로 모니터 된 이벤트가 발생할 때 에이전트를 활성화 시키는 방법이다. 여기서 활성화의 의미는 Daemon 상태로 설정되어 있다가 실제 동작을 할 수 있는 상태로 변경함을 의미한다. 앞에서 언급한 바와 같이 에이전트가 늘 상주해 있을 경우 많은 자원을 소모하므로, daemon 상태로 설정해놓는 방법은 여러 곳에서 사용되고 있다.

둘째는 수동형 실행 패턴으로 SMART 에이전트에 특정 요청이 왔을 때만 활성화 시키는 방법이다. 외부에서 여러 수집된 데이터를 한 번에 가져가기 위한 요청을 에이전트로 하는 경우, Daemon 상태에 있었던 에이전트를 활성화시켜서 데이터를 가져오기 위한 매개 변수를 전송한다.

이런 실행패턴을 지원하기 위하여서는 [그림 6]과 같은 기능 흐름이 필요하다.



[그림 6] SMART 에이전트 동적 실행패턴

1. 먼저 모바일 디바이스 혹은 서비스가 구동되는 서버의 자원상황을 확인한다. 여기서 확인하여야 하는 자원의 종류는 메모리, 네트워크 전송량, 배터리 등이 있다. 자원의 상황은 자원 충분과 불충분 상태로 나뉘며 이를 판단하는 기준

은 관리 정책에 따라 달라질 수 있다.

2. 자원 불충분일 경우, SMART 에이전트를 비활성화 하여 Daemon 상태로 만들어 놓는다. 이 상태가 되면 최소의 자원으로 관련 상태를 저장하고 있을 수 있으나, 실행이 되지는 않는다.
3. 자원 충분일 경우는, SMART 에이전트를 활성화하여 늘 가용한 상태로 만들어놓는다. 이 경우, 자원의 소모는 크지만, 관리의 효율성은 높아질 수 있다.
4. 가) 관리데이터와 관련된 이벤트가 발생하면 비활성화 되어 있는 SMART 에이전트의 경우 활성화된 후에 관리 데이터를 전송하기 위한 준비를 한다. 즉, 통신 패턴에 따라 관리 데이터 이벤트의 처리 방법이 달라진다.
5. 나) 관리데이터와 관련된 이벤트가 발생하면 활성화되어 있는 SMART 에이전트의 경우, 관리 데이터를 전송하기 위한 준비를 한다.
6. 관리 데이터 전송 요청의 경우는 SMART 서버에서 일정 시간마다 이루어지는 관리를 위해서 데이터를 전송하는 것을 의미하는데 이 경우에는 관리데이터와 관련된 이벤트의 발생과 무관하게 관련된 기능흐름이 수행된다.

위에서 보는 바와 같이 기능흐름은 피드백을 활용하여 자동적으로 수정되며, 특정 상황에 따라 SMART 에이전트를 자동 조정하고 실행되는 구조를 갖고 있다. 이와 같이 SMART 에이전트가 필요한 관리데이터를 보내면, SMART 서버에서는 관련 데이터를 처리할 수 있도록 추론 엔진을 구동한다. 추론 엔진을 구동하기 위한 추론 방법 및 저장소는 하나로 국한시키지 않는다. 즉, 대표적인 추론 기법으로는 패턴 인식 기법, 확률 기반 추론 기법, 모델 기반 추론 기법 등이 있는데, 하나의 추론 기법이 모든 상황에 알맞은 것이 아니기 때문에 플랫폼 차원에서 선택한 추론 기법에 따라 추론 엔진 및 저장소를 플러그 할 수 있는 확장 포인트를 제공한다.

본 논문에서는 결함 진단 프로세스를 상세히 다루지는 않으나, 기존에 있는 모델 기반 결함 진단 방법과 확률 기반 결함 진단 기법을 활용하여 원인을 분석할 수 있다[13, 26]. 또한, 원인에 맞는 해결 방법이 있는지 확인하기 위해서는 기존에 자율 관리 시스템에서 사용하는 ECA(Event-Condition-Action) 규칙을 활용할 수 있다[3]. 마지막으로 공용 저장소 업데이트의 경우, 확률 기반 결함 진단 기법에서 사용하는 업데이트 방법을 활용할 수 있다.

이런 진단 기법에 따라 파악한 원인을 해결하기 위하여서는 관련 계획과 원인을 해결할 수 있는 모듈이 필요하다. 본 논문에서 계획 수립과 원인을 해결할 수 있는 모듈에 대한 내용을 다루지는 않으나 기존의 연구 결과를 활용할 수 있다[4, 6]. 먼저 계획 수립에서는 원인을 해결할 수 있는 모듈을 찾고, 이를 실행하기 위한 환경 설정을 수행함으로써 실제 원인 해결 모듈을 실행시킬 수 있는 상태로 만든다. 그런 다음 원인 해결 모듈을 실행시킨다.

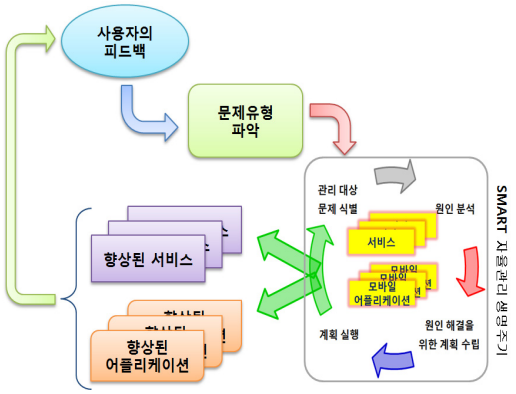
이 일련의 과정이 자율 관리 생명주기에 따라 에이전트와 서버가 관련 기능을 자율적으로 수행하게 된다. 즉, 관리자는 문제가 발생할 때마다 수작업으로 문제를 해결하여야 하는 일들의 대부분을 에이전트와 서버에서 처리해주게 되어서, 관리자는 관리를 위한 정책과 관리 결과에 따른 문서만 확인하면 되므로, 관리에 있어서의 효율성이 향상된다.

4.4 사용자 중심의 효율적인 운영

본 절에서는 사용자 중심의 효율적인 운영 기법을 제시한다. 사용자가 서비스나 어플리케이션 이용 시, 실시간으로 발생되는 문제점을 해결하기 위한 운영 시스템이 요구된다. SMART 운영 서브시스템은 기존 시스템과는 다르게, 독립적인 시스템이 아닌 사용자 중심의 효율적인 운영시스템을 위해 다른 시스템(관리 서브시스템, 배치, 배포 서브시스템)과 상호운영(Inter-operation)된다. 이러한 운영 시스템은 사용자에게 다음과 같은 기능들

을 제공한다.

사용자로부터 받은 피드백을 적용하여, 품질평가 후 이에 대한 문제유형과 이 문제해결을 위해 관리 서브 시스템으로 전송한다. 운영 서브 시스템은 배치와 배포 서브 시스템으로부터 서비스의 배치 정보와 어플리케이션의 배포 정보를 획득하여 사용자들이 원하는 서비스나 어플리케이션을 정확하고, 빠르게 발견할 수 있게 한다. 또한 사용자에게 이미 배치된 서비스나 배포된 어플리케이션을 관리하는 관리 서브 시스템으로부터 서버의 자원현황, 문제점 발생정보 등을 획득하여 품질평가에 포함한다. 사용자 중심의 효율적인 운영 절차는 다음 [그림 7]과 같다.

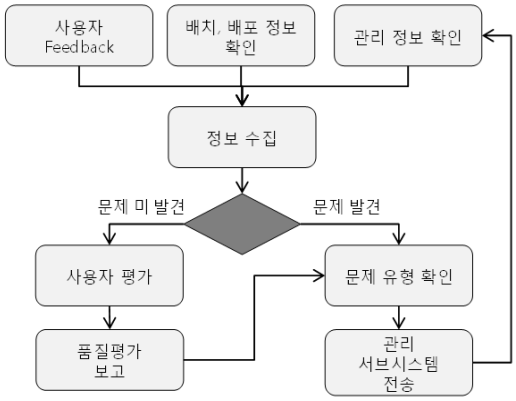


[그림 7] 사용자 중심의 효율적인 운영

[그림 7]에서와 같이 사용자로부터 받은 피드백을 기반으로 문제의 유형을 파악하고, 파악된 문제유형을 관리 서브 시스템으로 전송한다. 관리 서브 시스템으로부터 문제해결이 완료된 후에는 보다 향상된 서비스와 어플리케이션을 사용자들에게 제공할 수 있다.

효율적인 운영을 위해 SMART 환경에서는 다음과 같은 기능들을 제공하며, 제공되는 기능들은 [그림 8]과 같은 기능 흐름을 가진다.

1. 사용자 피드백, 배치, 배포 서브 시스템과 관리 서브 시스템으로부터 정보를 수집 및 분석한다. 분석된 정보를 기반으로 문제가 발견되었을 경우와 그렇지 않은 경우로 나뉜다. 문제에 대한 판단 기준은 운영 정책에 따라 달라질 수 있다.
2. 문제 발견 시 발견된 문제에 대한 유형을 분류하고, 분류된 문제의 유형에 대한 원인을 확인한다. 확인된 문제 원인은 관리 서브 시스템으로 전송된다.
3. 문제 미 발견 시 품질평가를 보고하며, 문제 유형을 파악할 수 있도록 한다.



[그림 8] 운영 서브시스템의 운영 기능 절차

위의 기능 흐름을 기반으로 운영되기 때문에 사용자들은 SMART에서 보다 고품질 서비스와 어플리케이션을 이용할 수 있다.

5. SMART 구현

본 논문에서 제안하고 있는 SMART는 2011년부터 개발하고 있는 슈퍼 모바일 컴퓨팅 플랫폼으로 2013년에 완료될 예정이다. 본 장에서는 현재 구현되어 있는 SMART의 각 서브 시스템을 보여주고, 구현되지 않은 부분에 대해서는 구현하기 위해 필요한 기술 및 주요 메커니즘에 대해서 설명한다.

5.1 개발 시스템

개발 시스템은 서비스 기반 모바일 어플리케이션 설계 지원 클라우드 서비스와 SMART 아키텍트

도구로 구성된다. 이 중 클라우드 서비스를 먼저 구현하기 시작하였으며 다음과 같은 플랫폼들을 활용하였다.

- HTML5, JavaScript, Cascading Style Sheets (CSS)을 활용하여 사용자 인터페이스를 설계하고 구현하였다. HTML 기반의 웹 인터페이스로 설계하고 구현한 이유는 사용자에게 일관성 있는 사용자 경험을 제공하기 위해서이다. 즉, 사용자가 클라우드 서비스를 활용할 때 사용하는 장비(모바일 장치나 개인용 컴퓨터)에 무관하게 필요한 기능을 수행할 수 있게 해주기 때문이다[8].
- Java Servlet Page(JSP)을 활용하여 데이터를 다루는 계층과 사용자 인터페이스 계층을 연결해 주었다. 즉, 사용자 인터페이스 계층은 HTML 페이지로 구현되고 데이터를 다루는 계층은 Database Management System(DBMS)로 이루어진다. HTML 자체에는 동적으로 데이터의 값을 할당하는 등의 기능을 수행하지 못하기 때문에 JSP를 활용하여 이를 지원한다.
- Java를 활용하여 데이터베이스를 접근하거나 데이터베이스에 특정 데이터를 활용하기 위한 질의를 처리할 수 있게 구현하였다.
- MySQL DBMS를 활용하여 사용자가 검토를 위해 업로드 할 수 있는 산출물을 관리한다.

[그림 9]는 클라우드 서비스를 활용하기 위하여 구현한 HTML 페이지 중 설계 지침의 일부를 재 정의하는 화면을 보여주고 있다. 이 페이지의 왼쪽 측면은 선택된 방법론의 전반적인 구조를 트리 형식으로 보여주고, 오른쪽은 그 중 사용자가 선택한 특정 단계나 활동의 자세한 정보를 보여준다. 이를 활용하여 사용자는 설계 지침에 있는 내용을 검색하거나, 일부 수정할 수 있다. 또한, 검토하기 원하는 산출물을 업로드 할 수도 있다. 이런 클라우드 서비스의 기능은 사용자의 사용 환경에 따라 제공된다. 개인용 컴퓨터를 이용하여 클라우드 서비스를 사용하는 경우는 모든 기능을 사용할 수 있지만, 모바일 장치를 이용하여 클라우드 서비스를

[그림 9] 서비스 기반 모바일 어플리케이션 설계 지원 클라우드 서비스

사용하는 경우는 내용 검색만을 사용할 수 있다.

5.2 배치와 배포 시스템

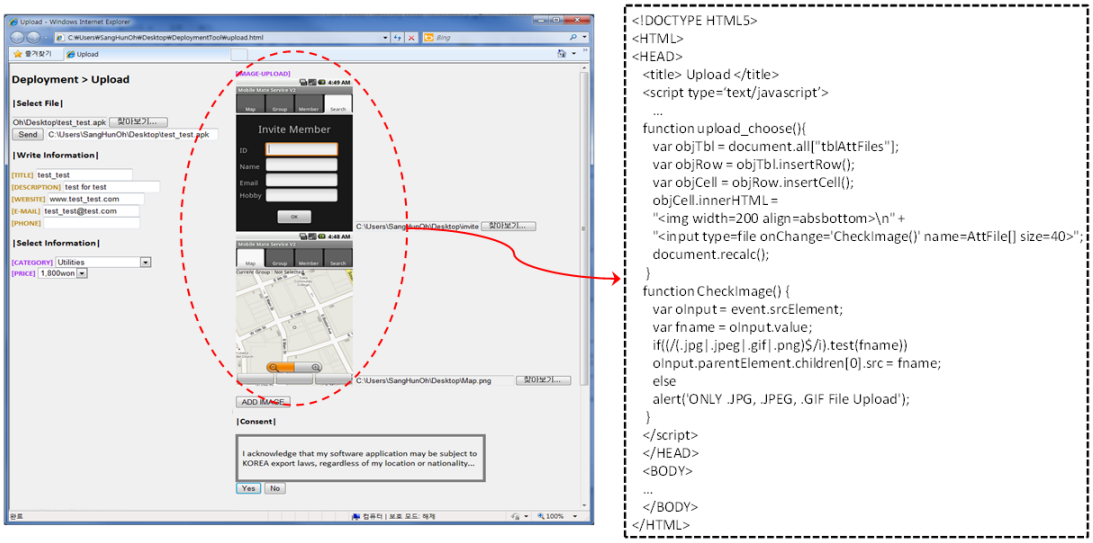
본 절에서는 SMART에서 제공하는 배치, 배포 시스템을 [그림 10]과 같이 HTML5 기반으로 구현하였으며, 이 중 어플리케이션 배포에 대한 실행화면 Screen-shot과 업로드 하는 기능에 대한 소스코드를 포함하였다. 배치, 배포 시스템의 중요 기능들은 기존의 애플 플랫폼이나 안드로이드 플랫폼과 유사하다. 개발된 어플리케이션 파일을 업로드 할 수 있는 파일 선택, 개발자의 정보를 입력할 수 있는 정보입력, 배포할 어플리케이션의 카테고리나 가격을 선택할 수 있는 정보선택, 그리고 어플리케이션의 실제 실행 화면을 그림파일로 업로드 할 수 있다. 마지막으로 어플리케이션에 대한 법률적인 사항도 포함되어 있다.

5.3 관리 시스템

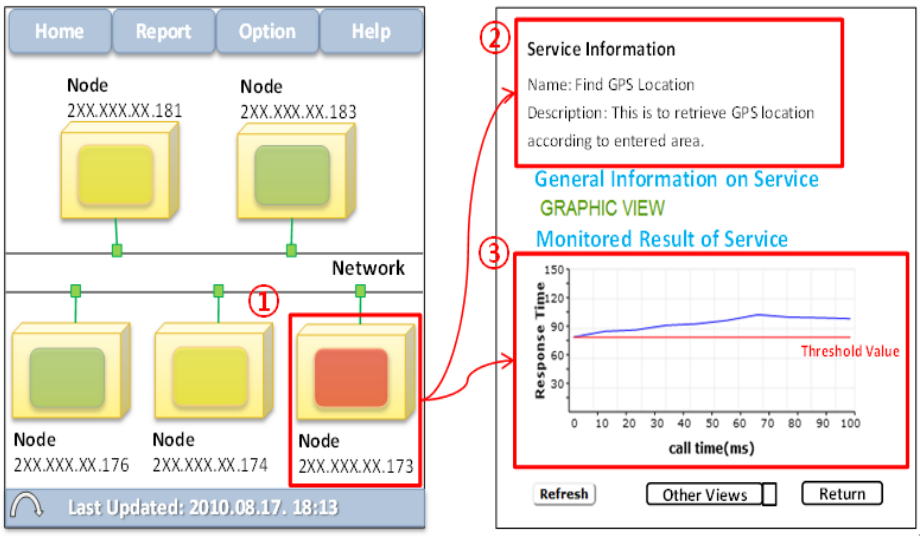
자율 관리 시스템을 구현하기 위해서는 3개의 주요 활동 중 모니터링이 먼저 구현되어야 한다. 모니터링 결과에 따라 결함을 진단하거나, 모바일 어플리케이션 및 서비스의 품질을 평가할 수 있고, 이를 기반으로 자율적인 관리가 실행되기 때문이다.

우리는 개발 관리 시스템에서 사용하고 있는 플랫폼을 활용하여 모니터를 구현하기 위해서 HTML 기반의 사용자 인터페이스, JSP 기반의 제어 컴포넌트, Java 기반의 서버 프로그램, MySQL 기반의 DBMS를 사용하였다. 또한, 모니터링 데이터를 전송하기 위한 인터페이스로 REST(Representational State Transfer)와 WSDL(Web Service Description Language) 모두를 사용하였다. [그림 12]는 모니터를 사용하기 위하여 제공하는 HTML 기반의 사용자 인터페이스 구현 결과를 보여준다.

상단에 4개의 버튼은 모니터의 주요 기능을 사용할 수 있도록 제공하는 것이다. Home은 메인 페이지로 돌아가기 위한 버튼으로 그림에서 보는 것과 같은 페이지로 돌아갈 수 있게 해준다. Report는 현재 모니터 되고 있는 모바일 어플리케이션 및 서비스의 상태를 요약하는 보고서를 만들어 주는 기능을 한다. Option은 모니터링 결과를 보여주기 위한 설정을 관리하는 기능을 한다. Help는 모니터링을 사용하는 데 필요한 도움말을 제공한다. 중단에 노드와 네트워크 화면은 모니터링



[그림 10] SMART 배치, 배포 시스템



[그림 11] 에이전트 기반 관리 중 모니터링 기능

결과를 한 눈에 보여주는 역할을 한다. 여기서 노드와 같이 표현되는 녹색, 황색, 적색은 각각 관리 대상의 현재 상태를 보여준다. 즉, 녹색은 문제없음을 의미하고, 황색은 주의를 의미하며, 적색은 위험을 의미한다. 예를 들어, 노드 2XX.XXX.XX.173의 색이 적색이므로 현재 위험함을 의미한다.

이런 모니터는 다음 시나리오를 바탕으로 한다. 먼저, 모바일 어플리케이션 및 서비스 관리자는 브라우저를 통해 모니터를 연다. 관리자는 관리 대상 중 문제가 발생한 대상을 식별한다. 이 때, 관리 대상의 색깔은 빨강색이다. 3) 관리자는 특정 문제 있는 관리 대상을 선택하면 모니터 된 데이터와 그래프를 가져온다.

5.4 운영 시스템

현재 운영 시스템은 개발 중에 있으며, 다음 [그림 12]와 같은 형태로 운영 서브시스템의 구성은 서비스 구독이나 어플리케이션 구매, 사용량 측정, 청구서 발부, 지불, 수익분배, 비즈니스 요약 보고서 생성으로 되어 있다.

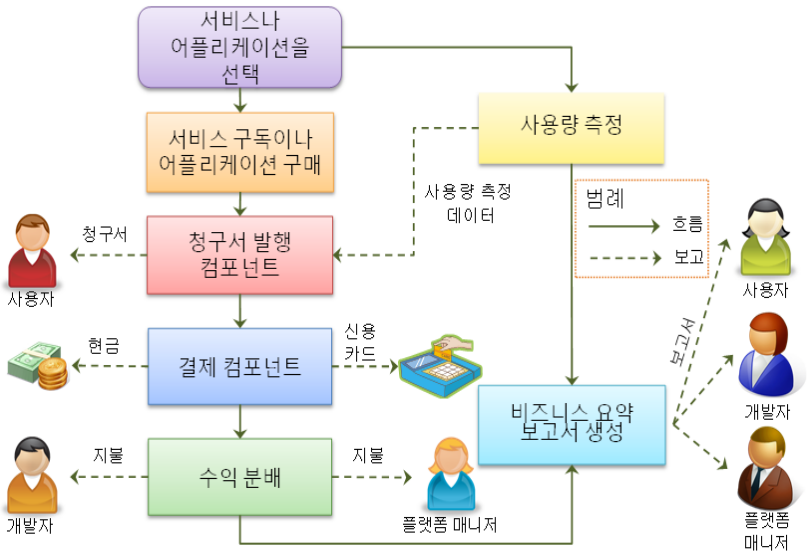
개발 진행 중인 운영 서브시스템은 다음과 같은 형태로 운영된다. 1) 사용자가 원하는 서비스나 어플리케이션을 선택하면, 2) 사용한 만큼의 금액을 계산하여 사용자에게 요청한다. 3) 요청된 금액에 대해서 신용카드나 현금으로 결제할 수 있으며, 4) 발생된 수익을 운영방침에 의해 개발자와 플랫폼 매니저가 분배를 한다. 5) 마지막으로 위의 내용을 사용자와 개발자, 플랫폼 매니저에게 구매 리스트, 결제수단, 결제금액, 분배된 금액, 사용량 측정에 관한 내용을 사용자, 개발자, 플랫폼 매니저에게 맞는 내용을 보고 한다.

6. 평 가

본 장에서는 제안한 시스템의 우수성을 보이기 위하여 각 서브시스템 별로 정성적으로 평가를 한다. 현재 타 시스템 중에서 본 논문에서 제안한 시스템의 모든 기능을 지원하는 시스템은 없다. 따라서 제안한 시스템의 각 서브시스템과 타 시스템을 비교하였다.

6.1 개발 서브시스템 기능성 평가

현재 타 시스템에서는 서비스 기반 모바일 어플리케이션 설계 및 개발을 명시적으로 지원하고 있



[그림 12] SMART 운영 시스템

지는 않다. 서비스를 설계하고 개발할 수 있는 지원 도구를 제공하거나, 모바일 어플리케이션을 개발할 때, 서비스를 활용하여 개발할 수 있는 도구를 제공하고 있다.

먼저, 서비스를 설계하고 개발할 수 있는 지원도구 중 대표적인 것은 IBM WebSphere 툴킷, Eclipse WTP 등이 있다. 이 도구에서는 서비스를 구현을 위하여 공통적으로 WSDL2Java, Java2WSDL 도구를 포함하고 있다. WSDL2Java는 명세된 서비스 인터페이스를 기반으로 Java 구현의 골격을 자동으로 생성해준다. Java로 구현한 기능성을 기반으로 서비스 인터페이스의 명세를 자동으로 생성해준다. 이 밖에 자바를 서비스화 하기 위한 패키지 파일을 자동으로 생성해주는 기능을 제공하고 있다. 또한, 이 도구는 공통적으로 UML 모델링을 지원할 수 있는 도구를 포함한다. UML 모델링을 통하여 서비스를 설계할 수 있다. 그러나 이 도구들은 서비스를 어떻게 설계할 것인지에 대한 지침이 있는 것이 아니다. 이 부분은 방법론이나 기법에서 제공해주는 것이다. 반면, SMART에서는 재사용 가능한 서비스를 설계하기 위해 필요한 활동, 활동 지침 등을 인터넷 브라우저에서 볼 수 있도록 클라우드 서비스 형태로 제공한다. 그러므로 SMART에서 제공하는 기능성은 기존에 있는 서비스 설계 및 개발 도구의 부족한 점을 보완한다고 할 수 있다.

모바일 어플리케이션 개발 도구 역시 비슷하다. 모바일 어플리케이션을 개발하기 위해서는 각 플랫폼에 특화된 개발 도구를 사용한다. 예를 들어 iOS에서 실행되는 모바일 어플리케이션을 개발하기 위해서는 XCode를 사용하여야 하고, Android에서 실행되는 모바일 어플리케이션을 개발하기 위해서는 Android 개발을 위한 Eclipse를 사용하여야 한다. 또한, 각 플랫폼에서 제공하고 있는 기능을 API 명세서, 개발 지침서 등을 통해서 제공하고 있다. 그러나 이 도구들 역시 모바일 어플리케이션을 어떻게 설계할 것인지에 대한 지침을 포함하고 있지 않다. 반면, SMART에서는 배포되어 사용할 수 있는 서비스를 활용하여 모바일 어플리케이션을 위한 아키텍처 설계에 필요한 활동 및 활동 지침 등을 클라우드 서비스 형태로 제공한다. 또한, 아키텍처 설계 후, 네트워크 부하량, 서비스

〈표 2〉 저장소 가용성 성능평가에 대한 환경설정

부하(Load) : 주어진 기간 안에 발생하는 서비스 요청 개수이며, 본 실험에는 500ms의 기간을 고정하고, 서비스 요청 개수를 10개에서 1500 개 사이는 100개의 단위로 증가하였고, 1500과 2000의 사이는 500개를 증가하였다. 2000개 이후에는 1000개를 증가하여 서비스의 사용요청 개수를 3000까지 증가시킨다.
응답시간(Response Time) : 서비스 요청(Request)이 서비스 제공자의 노드에 도착한 시간에서부터, 서비스 실행이 종료되어 그 결과 값을 서비스 사용자에게 반환하는 순간까지의 결과 시간을 의미한다. 즉, 본 서비스 사용자가 서비스 사용을 위해 GRM에게 메시지를 전달했을 때, GRM이 메시지를 전달받은 시간과 GRM에서 서비스 사용자에게 메시지를 반환한 시간과의 차이를 나타낸다. 본 실험에서는 오차를 줄이기 위해 평균값을 사용한다. 평균값은 같은 환경의 실험을 5회씩 반복 수행하여 계산한다. 본 실험에서는 응답시간을 측정하기 위해 다음과 같이 두 개의 메트릭을 정의하였다.

메트릭 1	메트릭 2
$Response\,Time = Processing\,Time + Waiting\ \ Time + Migration\,Time + LoadDistribution\,Time$	$Average\,Response\,Time = \frac{\sum_{i=1}^{n} Response\,Time_i}{n}$

기반 모바일 어플리케이션의 전체 성능 등을 시뮬레이션 할 수 있는 기능을 추가적인 도구로 제공한다. 그러므로 SMART에서 제공하는 기능성은 기존에 있는 모바일 어플리케이션 도구의 부족한 점을 보완한다고 할 수 있다.

결론적으로 SMART에서 개발 서브 시스템은 타 시스템에서 제공하고 있는 기능성을 대체하기 위해 만들어진 것이 아니라, 타 시스템에서 제공하고 있는 기능성을 체계적으로 활용할 수 있는 기법을 제공하기 위해 만들어졌다.

6.2 배치 및 배포 서브 시스템 성능 평가

배치 및 배포 서브 시스템의 평가는 타 시스템의 저장소를 물리적으로 접근하여, 성능을 측정할 수 없기 때문에, SMART 저장소 가용성에 대한 성능 평가를 정량적으로 수행하였다. 저장소 가용성에 대한 성능평가 환경은 <표 2>와 같이 구성된다.

위의 성능평가 환경을 기반으로 17번의 실험을 수행하였으며 사용자 요청도 동일하게 수행하였다. 서비스 이주를 위한 실험에서 서비스는 보다 가까운 거리에 있는 서비스 노드에게로 이주 하여 서비스를 제공하는 시나리오를 가진다. 응답시간은 비례하여 증가한다고 하고 응답시간이 빠른 노드로 서비스를 이주 한다. 다음 <표 3>에서 이주 후의 평균 응답 시간은 문제가 없는 단일 노드에서는 비슷한 응답시간을 보이고 있음을 나타낸다. 표 3에서 이주 전과 후의 응답시간의 차이는 시간이 사용자 요청이 증가 할수록 함께 증가하는 것을 확인 할 수 있다.

본 실험에서는 서비스의 SLA(Service Level Agreement)에서 응답시간의 임계값을 45ms로 정의한다. 그러므로 서비스는 45ms이내의 응답시간을 유지해야 한다. 따라서 [그림 13]에서 이주 전 응답시간을 나타내는 녹색 선은 500개의 사용자 요청이 발생하였을 때 SLA를 넘어선다. 반면 이주를 완료한 서비스는 SLA에 명시한 응답시간보다 빠른 응답시간을 보이며 더 많은 사용자 요청을 처리 하고 있다.

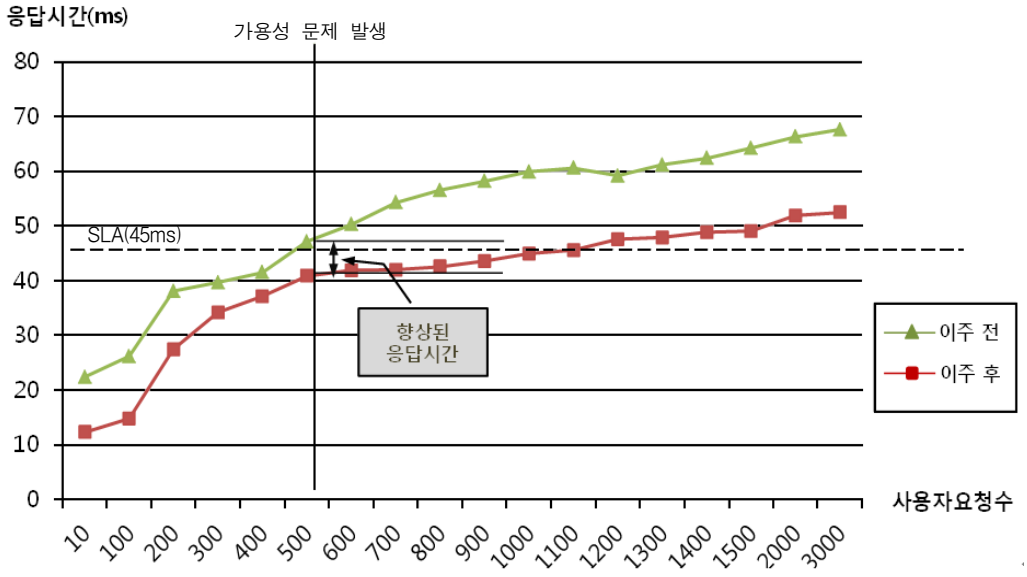
따라서 [그림 13]에서 500개의 사용자 요청이 발생하였을 때, 가용성 문제가 발생하였고 이주 기법을 적용함으로써, 응답시간을 SLA에 명시한 값 이하로 감소시키고 있음을 나타내고 있다.

6.3 관리 서브시스템 성숙도 평가

현재 출시되어 있는 타 시스템은 대부분 서비스를 관리하기 위한 목적으로 만들어졌다. 모바일 어플리케이션의 경우는 실시간으로 관리하는 것이 아니라, 운영 시 발생된 문제를 식별하고, 식별된

〈표 3〉 SMART 저장소에서 서비스 이주 실험 결과

실험ID	서비스 부하	이주 전 평균응답시간(ms)	이주 후 평균응답시간(ms)	향상된 시간(ms)
MI-01	10	22.4	12.29	3.33
MI-02	100	26.154	14.75	3
MI-03	200	38.1	27.452	3.234
MI-04	300	39.665	34.154	4.1226
MI-05	400	41.5	37.124	5.8025
MI-06	500	47.14	40.846	8.21
MI-07	600	50.24	41.8564	9.0959
MI-08	700	54.246	41.954	9.33
MI-09	800	56.54	42.554	9.065
MI-10	900	58.154	43.5124	10.7501
MI-11	1000	59.854	44.876	9.98
MI-12	1100	60.56	45.514	12.456
MI-13	1200	59.156	47.547	11.592
MI-13	1300	61.134	47.8984	13.6032
MI-14	1400	62.354	48.8452	13.3108
MI-15	1500	64.17	48.99	13.908
MI-16	2000	66.249	51.846	11.011
MI-17	3000	67.6	52.44	10.56



[그림 13] 서비스 이주 실험의 응답시간 그래프

문제를 수정하여, 새로운 버전의 모바일 어플리케이션을 사용자에게 제공하는 방법으로 관리를 수행하고 있다. 그러므로 본 절에서는 서비스 관리를 위한 타 시스템과 SMART의 관리 서브 시스템을 자율 컴퓨팅 적용(Adoption) 모델에 따라 성숙도를 비교하고 결과를 평가한다[23].

먼저 서비스를 관리를 위한 도구로는 IBM에서 제공하고 있는 Tivoli Autonomic Computing Toolkit과 Oracle의 IT Service Management Suit이 있다. IBM Tivoli Autonomic Computing Toolkit의 경우, 실시간으로 서비스를 관찰하고, 문제가 발생하는 경우 이 문제와 문제에 대한 해결책을 관리자가 XML로 기술한 후 저장한다. 그러면 같은 문제가 발생하는 경우 해결책에 따라 에이전트가 관련 기능을 수행한다. 이 도구는 기능성 측면에서 평가하면 4번째 단계인 Closed Loop이고, 지원 범위에서 평가하면 4번째 단계인 Multiple of different types라 할 수 있다. 단, 지원 범위의 경우, 관리자가 문제 및 해결책을 기술하기 위한 능력의 성숙도에 따라 달라질 수 있다. 그러나 IBM의 경우 문제 및 해결책을 기술하기 위한 별도의 명세를 제공하고 있으므로 문제 및 해결책을 좀 더 체계적으로 기술할 수 있다.

Oracle의 IT Service Management Suit의 경우, 서비스 관리자가 특정 상황에 대하여 미리 문제로 정의하여야 한다. 그런 다음 운영 시 같은 상황이 발생하면, 문제가 발생했다는 경고를 관리자에게 알리는 방식을 취하고 있다. 이 도구는 기능성 측면에서 평가하면 2번째 단계인 Invest and Monitor이고, 지원 범위 측면에서 평가하면 2번째 단계인 Simple Instance이다.

SMART의 관리 서브시스템의 경우, IBM에서 제공하고 있는 Tivoli Autonomic Computing Toolkit과 같은 단계를 지원한다고 할 수 있다. 먼저, 기능성 측면에서는 자율 관리를 위한 생명주기에 따라서 에이전트의 실행 패턴이 정의되어 있고, 이에 따라 Closed Loop으로 실행되기 때문이다. 지원 범위 측면에서는 문제 및 해결책을 기술하기 위하여 기존에 연구되어 있던 모델 기반 진단 기법 및 확률 기반 진단 기법에서 각 상황을 명세하는 방법을 활용하고 있기 때문에 Multiple of diferet types를 지원한다 할 수 있다.

결론적으로, SMART의 관리 서브 시스템은 관리 대상이 서비스 및 모바일 어플리케이션이므로 타 시스템의 관리 대상을 포함하고 있다. 또한, IBM에서 제안하고 있는 자율 컴퓨팅 적용모델에 따라 성숙도를 평가해본 결과, IBM의 Tivoli Autonomic Computing Toolkit과 동일하고, Oracle의 Service Management Suit 보다는 높은 것으로 결론을 내릴 수 있었다.

6.4 운영 서브 시스템 평가

운영 서브시스템은 기존의 안드로이드 마켓, 애플 앱 스토어, 아마존 앱 스토어 등과 비교했을 때, 기능적으로는 큰 차이가 없지만, SMART의 운영 서브 시스템은 사용자 중심의 효율적인 운영시스템이라는 점이 차이점이다. 즉, 사용자로부터 받은 피드백을 기반으로 문제의 유형을 파악하고, 파악된 문제유형을 관리 서브 시스템으로 전송한다. 관리 서브 시스템으로부터 문제해결이 완료된 후에는 보다 향상된 서비스와 어플리케이션을 사용자들에게 제공하는 방식으로 수행되고 있다.

결론적으로 기존의 다른 운영 시스템들도 SMART 운영 서브 시스템에서 제공하는 기능들과 같거나 유사한 기능들이 존재한다. 그러나 SMART 운영 서브 시스템은 기존 시스템과는 다르게, 독립적인 시스템이 아닌 사용자 중심의 효율적인 운영시스템을 위해 다른 시스템(관리 서브 시스템, 배치, 배포 서브 시스템)과 상호 운영된다. 이점이 기존의 운영 시스템과 비교하였을 때, 다르다는 결론을 내릴 수 있다.

7. 결 론

모바일 장치들은 커뮤니케이션 장치 뿐만 아니라 컴퓨팅 장치로 최근 각광을 받고 있다. 그러나 모

바일 장치가 가진 핵심 문제는 크기가 작은 요인 때문에 자원에 한계가 있다는 점이다. 따라서 복잡한 로직과 대용량의 자원을 소비하는 어플리케이션들은 개발하기 어렵다. 또한, 모바일 장치의 엔터프라이즈 컴퓨팅은 많은 양의 자원들을 요구하기 때문에 어려움이 많다. 이를 해결하기 위해 서비스 기반 모바일 어플리케이션 및 서비스를 개발, 관리, 배포 및 배치, 운영하기 위한 모든 기능을 하나의 플랫폼에서 제공하기 위한 SMART를 제안하였다. SMART를 이용하여 얻을 수 있는 장점은 다음과 같다.

- 다중 네트워크와 다중 OS를 지원하고, 여러 장치에서 재사용할 수 있는 디자인을 제공한다.
- 네트워크 전송에 대한 과부하 발생 시 이와 관련된 것들을 줄여준다.
- 모바일 디바이스를 위한 사용자 인터페이스를 제공한다.
- 서브시스템에서 각 요소들을 재사용하기 위한 통합 비용과 테스트를 피하거나 줄일 수 있다.
- 효율적인 방법으로 SMART의 전체를 관리 한다.

개발에서부터 테스팅 그리고 배치까지 일관성 있고, 효율적으로 제공하며, SMART는 도구(Tool)와 방법론 모두를 제공한다. 현재 유용한 플랫폼들은 독립적인 모바일 어플리케이션을 위한 디자인이며, 모바일 어플리케이션 기반의 서비스를 위한 것은 아니다. 특정 상표가 부착되어 있고, 저장소와 관리가 중앙집권화 되어 있기 때문에 가용성의 한계가 있다. 서비스 보다는 어플리케이션 배포 및 유통을 위해 집중되어 있다. SMART와 같이 유사한 플랫폼은 존재하나 모바일 어플리케이션 혹은 모바일 서비스 개발부터 운영까지 총괄하는 통합 플랫폼은 미흡한 실정이다. 따라서 SMART와 같은 통합 플랫폼이 요구된다. 현재 SMART는 2011년부터 개발하고 있는 슈퍼 모바일 컴퓨팅 플랫폼으로 2013년에 완료될 예정이다.

참 고 문 헌

[1] 신동천, 장효선, “상황인식 컴퓨팅 환경에서 동적 서비스 컴포지션 기술 동향 분석”, 「한국IT서비스학회지」, 제9권, 제3호(2010), pp.163-178.

[2] 최재현, 박제원, 이남용, “JBI기반 ESB환경에서 효과적인 메시지 추적을 위한 메시지모니터링 프레임워크”, 「한국IT서비스학회지」, 제9권, 제2호(2010), pp.179-192.

[3] Bailey, J., A. Poulovassilis, and P. T. Wood, “An Event-Condition-Action Language for XML”, *Proc. WWW, Hawaii,* 2002.

[4] Brachman, R. J. and H. J. Levesque, *Knowledge Representation and Reasoning*, Morgan Kaufmann, 2004.

[5] Chang, S. and S. Kim, “A Variability Modeling Method for Adaptable Services in SOC”, *Proc. the 11th International Software Product Line Conference(SPLC)*, (2007), pp. 261-268.

[6] Cheun, D. and S. Kim, “An Engineering Process for Autonomous Fault Management in Service-Oriented Systems”, *Proc. 9th IEEE/ACIS International Conference on Computer and Information Science(ICIS)*, (2010), pp.901-906.

[7] Clark, W. and M. King, “Magic Quadrant for Mobile Enterprise Application Platforms”, *Gartner RAS Core Resaerch Note G001727 28*, 2009.

[8] David, M., *HTML5 : Designing Rich Internet Applications*, Focal Press, 2010.

[9] Franke, D. and C. Weise, “Providing a Software Quality Framework for Testing of Mobile Applications”, *Proc. IEEE Fourth International Conference on Software Testing, Verification and Validation(ICST)*, (2011), pp.431-434.

[10] Gammage, B., D. Plummer, E. Thompson, L. Fiering, H. LeHong, F. Karamouzis, C. Rold, K. Collins, W. Clark, N. Jones, C.

Smulders, M. Escherich, M. Reynolds, and M. Basso, "Gartner's Top Predictions for IT Organizations and Users, 2010 and Beyond : A New Balance", *Gartner Press release*, 2009.

[11] Greenfield, J. and K. Short, "Software factories : assembling applications with patterns, models, frameworks and tools", *Proc. ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, (2003), pp.16-27.

[12] Grønmo, R., D. Skogan, I. Solheim, and J. Oldevik, "Model-driven Web Services Development", *Proc. IEEE International Conference on e-Technology, e-Commerce and e-Service(EEE)*, (2004), pp.42-45.

[13] ISO/IEC, ISO/IEC 12207 : *Information Technology-Software life cycle processes*, International Organization for Standardization, 1995.

[14] Kassinen, O., E. Harjula, T. Koskela, and M. Ulianttila, "Guidelines for the Implementation of Cross-platform Mobile Middleware", *International Journal of Software Engineering and Its Applications*, Vol.4, No.3(2010), pp.43-58.

[15] Kephart, J. and D. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, Vol.36, No.1(2003), pp.41-50.

[16] Kim, C., E. Cho, and S. Kim, "A Variability Design Techniques For Enhancing Component Reusability", *International Journal of Software Engineering and Knowledge Engineering(IJSEKE)*, Vol.16, No.3(2006), pp. 425-470.

[17] Kim, S., J Her, and S. Chang, "A Theoretical Foundation of Variability in Component-Based Development", *Elsevier Information and Software Technology(IST)*, Vol.47(2005), pp.663-673.

[18] König-Ries, B. and F. Jena, "Challenges in Mobile Application Development", *it-Information Technology*, Vol.52, No.2(2009), pp. 69-71.

[19] Kranenburg, H. and H. Eertink, "Processing Heterogeneous Context Information", *Proc. The 2005 Symposium on Applications and the Internet Workshops*, (2005), pp.140-143.

[20] La, H. and S. Kim, "A Service-based Approach to Developing Android Mobile Internet Device(MID) Applications", *Proc. IEEE International Conference on Service-Oriented Computing Application(SOCA)*, (2009), pp.196-202.

[21] Lindquist, D., H. Madduri, C. J. Paul, and B. Rajaraman, "IBM Service Management Architecture", *IBM Systems Journal*, Vol.46, No.3(2007), pp.423-440.

[22] Michael, B. and M. John, "A simple model of advertising and subscription fees", *Elsevier Economic Letters*, Vol.69(2000), pp.345-351.

[23] Miller, B., "The autonomic computing edge : The role of the human in autonomic systems", *IBM developerWorks*, 2005.

[24] Najafi, M. and K. Sartipi, "A framework for context-aware services using service customizer", *Proc. of The 12th International Conference on Advanced Communication Technology(ICACT)*, Vol.2(2010), pp.1339-1344.

[25] Peischl, B. and F. Wotawa, "Model-Based Diagnosis or Reasoning from First Principles", *Intelligent System*, Vol.18(2003), pp. 32-37.

[26] Pressman, R. S., *Software Engineering : A Practitioner's Approach, 7th ed.*, McGraw-

Hill, 2009.

[27] Satyanarayanan, M., "Mobile computing : the next decade", *Proc. 1st ACM Workshop on Mobile Cloud Computing and Services : Social Networks and Beyond*, Vol.5(2010).

[28] Unhelkar, B. and S. Murugesan, "The Enterprise Mobile Applications Development Framework", *IT Professional*, Vol.12, No.3 (2010), pp.33-39.

[29] Welke, R., R. Hirschheim, and A. Schwarz, "Service-Oriented Architecture Maturity", *IEEE computer*, Vol.44, No.2(2011), pp.61-67.

◈ 저 자 소 개 ◈

오 상 헌 (sanghun1004@gmail.com)

건양대학교 정보전산학과에서 학사, 숭실대학교 컴퓨터학과에서 전산한 석사를 취득하고, 숭실대학교 컴퓨터학과에서 박사 과정 중이다. 박사 전공은 소프트웨어 공학이며, 현재 소프트웨어 테스팅 및 품질을 실용화하기 위한 내용을 연구 중이다. 관심분야는 소프트웨어 품질 모델, 소프트웨어 테스팅 기법 및 도구 등이다.

천 두 완 (raphael.cheun@gmail.com)

숭실대학교 컴퓨터학부에서 학사, 숭실대학교 컴퓨터학과에서 전산한 석사를 취득하고, 숭실대학교 컴퓨터학과에서 박사 과정 중이다. 박사 전공은 소프트웨어 공학이며, 현재 모바일 컴퓨팅, 컨텍스트 인지 컴퓨팅을 소프트웨어 공학에 접목하는 것을 연구 중이다. 관심분야는 지능형 모바일 어플리케이션 개발과 이를 지원하는 플랫폼 개발, 클라우드 컴퓨팅 등이다.

김 수 동 (sdkim777@gmail.com)

Northeast Missouri State University 전산학 학사, The University of Iowa 전산학 석사, 박사를 취득하였고, 현재 숭실대학교 컴퓨터학교 교수로 재직 중이다. 관심분야는 서비스 지향 아키텍처(SOA), 클라우드 컴퓨팅(Cloud Computing), 모바일 서비스(Mobile Service), 객체지향 S/W공학, 컴포넌트 기반 개발 (CBD), 소프트웨어 아키텍처이다.