논문 2008-45SD-2-20

서비스 지향 아키텍처 기반의 모바일 서비스 모델링

(Mobile Service Modeling Based on Service Oriented Architecture)

장 영 원*, 유 철 중**, 노 혜 민***

(Young-Won Chang, Cheol-Jung Yoo, and Hye-Min Noh)

요 약

서비스 기반 아키텍처(Service-Oriented Architecture;SOA) 등장 이후 서비스와 애플리케이션을 상호 연결하는 측면에서 여러 가지 접근 방법들이 개발되고 있다. 모바일과 같은 정보기기는 일반적인 접근법이나 전통적인 접근법과는 달리 메모리와 프로세스 등 단말기 자체의 제약과 스크린 및 입출력 및 사용자 인터페이스의 제약이 많아 여러 가지가 고려되어 설계되어야 한다. 본 연구는 서비스를 기반으로 한 구조에서 많은 제약을 갖는 모바일의 효율성을 높이기 위해 모바일 애플리케이션 요구사항을 분석하고 명세한 후 설계 방법을 최적화하고, 서비스 유스케이스 테스팅을 위한 확장된 유스케이스 명세를 제공하고, 서비스 명세로부터 서비스간 상호운영 테스트를 한다. 또한 모바일 애플리케이션의 성능을 확장하고, 제약을 최소화 할 수 있는 과정을 제시하며 유스케이스 테스팅을 위한 명세 방법과 서비스 상호간 상호운용성 보장 테스팅을 수행한다. 본 연구는 서비스 지향 아키텍처를 기반으로 모바일 서비스 명세 방법과 애플리케이션 테스팅 방법에 대한 아이디어를 제공한다는데 그 의의가 있다.

Abstract

Recently, the need for accessing information from anywhere at any time has been a driving force for a variety of mobile applications. As the number of mobile applications increases rapidly, there has been a growing demand for the use of Service Oriented Architectures(SOA) for various applications. Mobile based SOA offers a systematic way to classify and assess technical realizations of business processes. But mobile has severly restricted range of utilizing services in computing environment and more, a mobile computer is envisioned to be equipped with more powerful capabilities, including the storage of a small database, the capacity of data processing, a narrow user input and small size of display. This paper present mobile adaption method based on SOA to overcome mobile restriction. To improve mobile efficient we analyzing mobile application requirement, writing service specification, optimizing design, providing extended use case specification which test use case testing and testing service test case which derived from service specification. We discuss an mobile application testing that uses a SOA as a model for deploying discovering, specifying, integrating, implementing, testing, and invoking services. Such a service use case specification and testing technique including some idea could help the mobile application to develop cost efficient and dependable mobile services.

Keywords : SOA, UML, Modeling, Mobile

I. 서 론

SOA의 기본적인 아이디어는 가볍고 안정된 컴포넌트 형식의 서비스를 구현하고 이를 제공 시 독립적으로 또는 컴포넌트 간 결합을 통해 비즈니스 서비스를 제공하는 것이다[1]. 이러한 서비스의 활용 및 설계방식은 엔터프라이즈 관점에서 비즈니스 프로세스를 실현하고 소프트웨어 아키텍처의 관점에서 서비스를 설계하는데 용이하며 XML을 이용하여 비즈니스와 서비스간의 뷰를 통합하고 구현할 수 있다는 점으로 인해 양자간에 주목을 받고 있다. 현재 SOA는 주로 대형 벤더나 소프트웨어 개발자들, 특히 아키텍처 설계자들에 의해 다뤄지고 있으며, 이러한 서비스를 수반한 아키텍처를 기반으로

* 학생회원, ** 정회원-교신저자, *** 정회원,
전북대학교 컴퓨터과학과
(Department of Computer Science, Chonbuk National University)

※ 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2006-521-D00419)
접수일자: 2007년11월13일, 수정완료일 : 2008년2월5일

하여 컴퓨터 설계자 중심, 비즈니스 프로세스 아키텍처 중심의 아이디어들이 많이 제시되고 있다. SOA는 주로 엔터프라이즈의 비즈니스 프로세스에 대한 요청과 소프트웨어 아키텍처의 요청으로 주로 개발이 되고 있지만, 실제 최근 활용도가 가시적으로 나타나는 유비쿼터스 환경 및 많은 정보이용의 수단이 되고 있는 모바일 및 PDA 활용에 관한 연구는 미약하다. 모바일 및 PDA는 그 자체적으로 이동성, 편의성 등을 가지고 있어 정보 활용에 있어 많은 장점을 가지고 있지만 반면에 설계의 복잡성과 프로세스 실행영역의 제한성에 있어서는 전자와의 트레이드오프를 갖는다. 모바일이 계속해서 빠른 발전을 보이고 있다고 할지라도 모바일 자체가 갖는 제약은 항상 존재한다. 첫째, 편의성과 이동성을 고려한 모바일 기기 자체의 메모리나 프로세스 등 제약을 가지고 있으며, 둘째, 사용자 입력과 화면 제약을 가지고 있고, 셋째, 애플리케이션 구현을 위한 복잡성 및 응용의 한계를 가지고 있다. 실제 이러한 제약사항을 없애기 위한 많은 아이디어들이 제안되고 있으며 본 논문에서는 모바일을 기반으로 하여 비즈니스 서비스를 설계하고, 제안된 설계를 위해 서비스별 명세를 작성하고, 유스케이스 테스팅을 위한 확장 명세를 제시하며, 서비스 상호운영을 위한 테스팅을 실시하고 그 결과를 설명한다. 또한 명세로부터 서비스를 최적화하기 위한 기법을 정의하고 모바일 설계를 최적화 할 수 있는 방안에 대해 제시한다. 이러한 구조는 모바일 애플리케이션 설계 시 최적화된 구조를 제공하고 실제 활용 시 서비스 접근 및 실행시간을 단축하여 사용자 환경에서부터 최종 모바일 정보를 얻기 위한 단계까지 불필요한 서비스의 구현을 막을 수 있고 모바일을 위한 강력한 서비스를 구축하는 장점을 갖는다.

본 연구는 크게 V장으로 구성되며 II장 관련연구에서는 SOA와 모바일에 관련된 연구가 리뷰되고, III장에서는 SOA기반의 모바일 서비스 애플리케이션 테스팅을 위한 방법론에 대해 기술하고, IV장에서는 실제 적용된 사례를 분석하고 테스팅하며 마지막으로 V장에서는 본 연구의 결과 및 의의를 설명한다.

II. 관련연구

기존의 컴포넌트 방식은 시스템에 제약적인 방식으로 컴포넌트간 결합방식을 제공하였으나, SOA 서비스 방식은 사용자 인터페이스로부터 메시징 채널까지 독립된 서비스간의 약 결합(loosly-coupled)을 통해 제공된다. 따라서 SOA는 컴포넌트 방식에 비해 비즈니스 서비스를 실행하고 서비스 컴포넌트를 운용하는데 있어 보다 넓고 확장 가능한 아키텍처를 제공한다. 이러한 이유로 SOA를 기반으로 하여 모바일 서비스 환경을 적용한 모바일 애플리케이션 테스팅에 관해 연구한다.

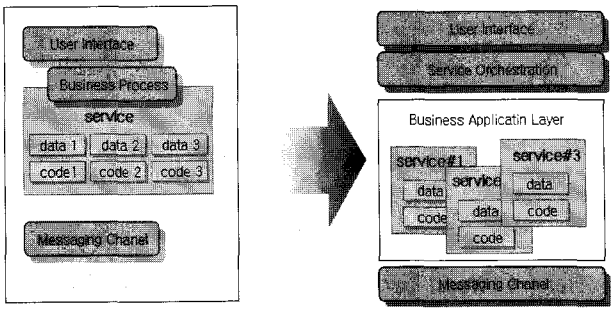


그림 1. SOA 구조
Fig. 1. SOA architecture.

본 연구를 위해 UML을 기반으로 한 SOA의 서비스 명세 방법에 대해 리뷰하고 모바일 적용가능성 및 서비스 테스팅에 관한 연구를 고찰해 본다.

1. SOA 서비스 명세

본 연구에서 사용되는 서비스 명세는 요구공학(requirement engineering)과 밀접하게 관련되어 있으며, 모델로부터 소프트웨어를 개발하는 모델기반 아키텍처(Model Driven Architecture)[2]와 UML모델링 툴을 사용하여 비즈니스 서비스를 명세 한다. SOA 서비스 명세는 비즈니스로부터 유도된 명세(business derived specification)와 시스템으로부터 유도된 명세(system derived specification)를 중심으로 연구가 진행되고 있다[3~8]. 본 연구에서는 두 가지 관점을 통합하고 명세하기 위한 도구로 객체지향 방법론에서 통합프로세스를 모델링하고 MDA의 모델링 도구로 많이 활용되는 UML을 사용한다.

2. 모바일 SOA 설계

SOA를 이용하여 모바일 서비스를 적용하기 위한 방법으로는 [9, 10]의 연구가 있었으며, [9]의 연구에서는 서비스 기반 인터페이스 설계와 서비스 기반 임베디드 디바이스 설계 시 필요한 요건들에 대해 제안하였으며, [10]의 연구는 모바일 서비스를 SOA에 매핑 시키기 위한 추상적 방법을 제안한다. 이러한 두 가지 연구는 체계적이고 근본적인 모바일의 기능적, 비 기능적 제약을 분석, 설계 시 기능 및 역할을 정하는데 있어 보다 국소적인 제약을 가지며 모바일의 전체적인 특성을 나타내

기에 범위가 추상적이고 미약하다고 볼 수 있다. 따라서 서비스를 모바일에 적용하기 위한 요구사항으로부터 설계 테스팅에 이르기 까지 전체적인 기능성 및 비 기능적 분석, 설계 그리고 테스트 프레임워크가 필요하다.

3. SOA를 위한 테스팅 방법

Hans-Gerhard Gross[11]의 연구에서도 언급했듯이 서비스의 테스팅에 관한 문제는 접근법 및 테스팅 방법론에 있어 많은 문제점을 가지고 있다. 지금까지 진행된 테스팅에 관한 연구는 컴포넌트 기반 상호운용성 테스팅과[12], 유스케이스 테스팅[13]의 연구가 진행되고 있으며 본 연구와 관련해서 모바일 서비스 구현 시 [14] 연구의 적용기법 적용 및 유스케이스 명세표를 이용한 테스트 케이스 추출 기법을 사용하여 각 각 명세표를 작성하고 테스트 케이스를 추출한다[14]. 테스트를 위한 테스트 케이스 명세(test case specification)는 일반적으로 테스트케이스 명세 식별자(test case specification identifier), 테스트 아이템(test items), 입력명세(input specification), 출력명세(output specification), 요구되는 환경(environment needs), 절차적 요구사항(special procedural requirements), 케이스간의 의존성(intercase dependencies)과 같은 정보를 포함한다[15~16]. 본 논문에서는 이와 같은 정보를 포함할 수 있도록 테스트 케이스 명세를 작성하고 서비스를 기반으로 수정하여 적용한다.

Ⅲ. 본 론

본 연구는 SOA를 기반으로 하는 모바일 서비스 애플리케이션 테스팅에 대한 연구이며 서비스별 명세와 설계 및 테스트를 위한 테스트 케이스를 추출하고 추출된 테스트 케이스를 이용하여 각각 유스케이스별 유스케이스 테스팅 명세와 서비스 상호 운영성 테스팅을 실시한다.

본론의 전체적인 순서는 다음과 같다.

1. 비즈니스 모델 분석 및 요구사항 분석
2. 모바일 서비스 식별 및 명세
3. 모바일 서비스 구현 및 테스팅

일반적으로 SOA 서비스 비즈니스 실행 구조는 다음 [그림 2]와 같다[17].

모바일 애플리케이션 서비스는 도메인 경계 설정과

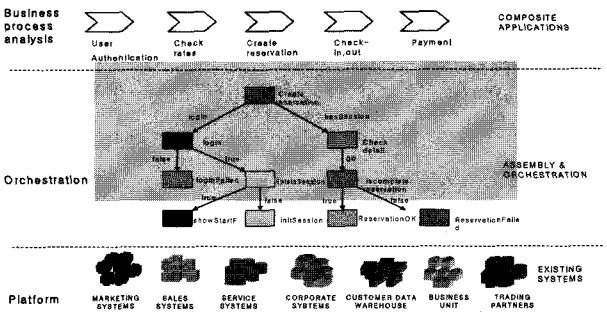


그림 2. SOA 서비스 비즈니스 실행 구조
Fig. 2. Business execution structure of SOA service.

고객에 대한 비즈니스 요구사항 분석으로부터 시작된다. 모바일 서비스 애플리케이션은 모바일을 위한 SOA 서비스 설계가 필요하고 모바일 상호운영을 위한 인터페이스가 요구된다. 일단 서비스가 시작되면 서비스는 약 결합된 서비스를 실행하며 상호간의 운영에 있어서 제어가 어렵고 관리가 복잡해진다. SOA는 서비스를 기반으로 약 결합된 컴포넌트 형식으로 정보를 효율적으로 전달하기 위한 구조이며 이러한 특성을 기반으로 비즈니스 요구사항으로부터 시스템의 요구사항까지 분석하여 각각의 유스케이스를 분석해야 보다 안정적인 시스템을 인도할 수 있다. 또한 모바일에서 요구되는 모바일의 제약사항과 특수요소를 포함한 요구사항을 분석 대상에 포함해야 한다. 본 논문에서는 첫째, 이러한 비즈니스 환경을 분석하고 둘째, 모바일 제약을 포함한 요구사항을 분석한다. 셋째, 모바일 서비스 구현을 위해 서비스를 명세하고 명세 된 명세서로부터 테스팅을 위한 유스케이스 명세와 상호운영 테스팅을 위한 테스트 케이스를 추출한다. 마지막으로 이러한 명세를 근거로 서비스를 구축하고 관련 문제를 보장하기 위한 테스팅을 수행한다.

1 비즈니스 모델 분석 및 요구사항 분석

비즈니스 목적은 비즈니스 가치와 고객의 가치를 영위하기 위한 근본적인 수단으로 설립되며 조직 운영과 고객 서비스 정보를 수단으로 제공한다. 이러한 비즈니스 목적을 달성하기위해 비즈니스 환경에 대한 이해를 바탕으로 정책 운용 모델이 설계된다. 비즈니스 모델은 정보기술 환경에서 서비스를 효율적으로 제공하기 위해 설계된다. 본 연구에서는 SOA를 도입하고 활용하기 위해 모바일 서비스 도메인에서 비즈니스 모델에 대한 환경 분석을 한다. 이를 위해 정보기술 분석가와 비즈니스 운용자 측면에서 기본적인 프로세스의 노출(expose)

과 소비(consume)가 분석된다.

본 단계의 분석은 모바일과의 상호 연동 및 관련성을 중심으로 기존의 시스템 또는 새로운 시스템을 설계할 때 제공되는 비즈니스 요구사항과 시스템의 요구사항 그리고 모바일의 요구사항 및 제약사항에 대한 정보기술과 비즈니스간의 뷰(view)를 일치시키기 위한 환경분석을 한다. 환경 분석의 산출물은 양 당사자간의 컨텍스트 다이어그램이 된다.

요구사항 분석 단계는 비즈니스 모델 분석으로부터 도출된 요구사항을 시스템 설계 관점에서 바라볼 수 있도록 보다 구체화 한다. 비즈니스 요구사항, 시스템 요구사항, 모바일 요구사항에 대한 기능적 요구와, 비 기능적 요구를 도출하고 최종 시스템 개발 및 배치 시 고려되어야 할 요구사항을 분석한다.

본 분석의 산출물은 보다 시스템레벨의 요구사항과 구현환경이 포함된 컨텍스트 다이어그램이 된다.

2 모바일 서비스 식별 및 명세

본 논문의 서비스 식별 및 명세는 유스케이스와, 기술적 요구사항, 현존 어셋 분석, 산업 표준 분석을 통해 서비스 식별 및 명세가 이루어진다. 이 과정은 모델링을 위한 기술인 서비스 기반 모델링 및 아키텍처(Service Oriented Modeling and Architecture;SOMA)에 의해 제안되었다[15].

본 논문은 서비스 식별을 위한 도메인 분해를 위해 상향식접근법 (top-down approach), 하향식 접근법 (bottom-up approach), 미들아웃접근법 (middle-out approach)중 모바일의 적용을 고려한 상향식 접근을 중심으로 접근한다. 그리고 도메인 분해를 통해 서브시스템을 식별하고 서브시스템 간 흐름과 상호 의존성을 분석한다.

서비스 명세는 추적가능성, 무상태성, 발견가능성, 재사용가능성, 노출가능성, 모바일 가능성을 중심으로 서비스 소비자에 의해 요구될 수 있는 요구사항을 탐색하고 서비스 및 규칙에 관련된 사항을 명세 한다. 또한 메시지와 이벤트에 대한 명세와 관리가 기술된다.

테스팅은 테스팅 기법과 방법을 정의하고 도출된 서비스 명세를 통해 테스트 케이스를 도출한다. 테스트케이스는 각각의 모바일 적용을 위한 서비스 유스케이스에 대한 기능, 입출력, 유스케이스간 관계, 제약사항에 대한 유스케이스 테스팅을 위한 명세와 서비스의 상호운영을 위한 상호운용성 테스팅을 위한 테스트 케이스 설계를 한다.

본 단계는 서비스 유스케이스 명세, 유스케이스 테스팅 명세, 상호운용성 테스팅 명세서가 산출된다.

3 모바일 서비스 구현 및 테스팅

구현 및 테스팅은 설계된 유스케이스를 이용해 서비스를 구현하는 단계이며 명세에 따라 최종적으로 시스템을 구현하는 과정을 포함한다. 서비스 구현은 모든 서비스를 위한 공간을 발견하고, 비즈니스 목표와 컴포넌트간의 추적성이 관리될 수 있도록 한다.

서비스 구현 시 중점적으로 다뤄지는 사항은 새로운 기능을 갖는 서비스 컴포넌트의 구축, 서비스가 노출되었을 때 코드 재사용성, 기존의 시스템과의 통합, 제3의 업체(third-party)의 제품과의 통합성 그리고 대안적 실현방안에 포커스를 맞춰 구현해야 한다.

서비스의 테스팅은 서비스 유스케이스 명세로부터 서비스의 기능을 테스팅 할 수 있는 유스케이스 테스팅과 서비스간 상호운용성을 보장하기 위한 상호운용성 테스팅으로 구분하여 테스팅된다. 첫째, 유스케이스 테스팅을 위해 유스케이스 명세로부터 기능의 보장을 위한 새로운 명세가 도출 될 수 있고, 도출된 명세를 이용하여 시스템의 기능성을 보장 할 수 있는 유스케이스 테스팅이 수행되어야 한다. 둘째, 상호운용성 테스팅은 서비스 유스케이스 명세로부터 유도되는 기능적 속성과 관계로부터 테스트케이스가 명세되고 최종적으로 서비스간 상호운영이 테스트 된다.

IV. 실 험

1 비즈니스 모델 분석

본 실험은 '자동차 렌탈 서비스'를 도메인으로 하며, 고객은 모바일, 유선 그리고 인터넷을 통해 서비스를 이용할 수 있다. 본 시스템은 전체적으로 SOA를 기반

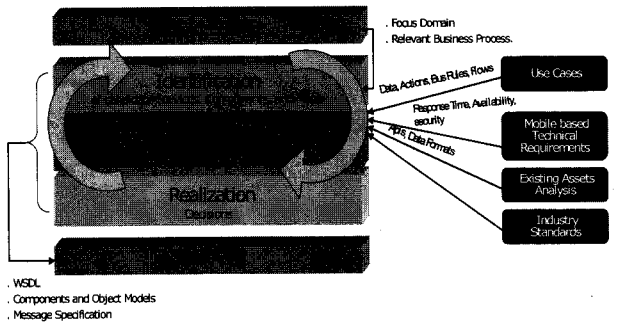


그림 3. 서비스 지향 모델링 및 아키텍처 분해 그림
Fig. 3. SOM and decomposition of architecture.

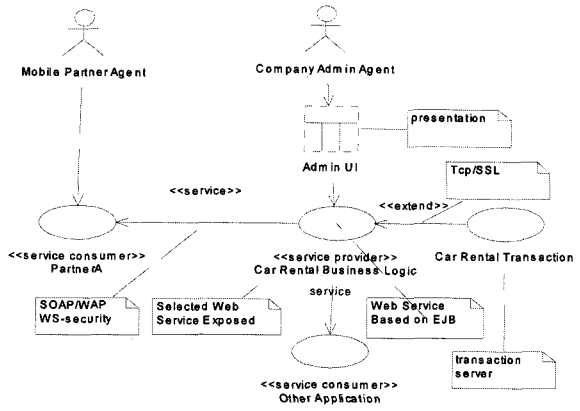


그림 4. 전체적인 비즈니스 분석
Fig. 4. Overall business analysis.

으로 분석, 설계, 구현 및 테스팅 되며, 특별히 모바일 사용자는 모바일을 통해 서비스를 이용할 수 있다. SOA기반 모바일 자동차 렌탈 서비스의 전체적인 비즈니스 모델은 분석은 [그림 4]와 같다.

서비스 프로바이더인 'Car Rental Busines Logic'은 자동차 대여에 관련하여 EJB를 기반으로 한 웹서비스를 제공하고 시스템 내부 또는 외부에 노출되어 다른 서비스 소비자에게 서비스를 제공한다. 'Mobile Partner Agent'는 서비스 소비자로서 서비스 제공자로부터 서비스를 제공받는다. 'Company Admin Agent'는 'Admin UI'를 통해 서비스를 관리한다.

2 요구사항 분석

본 연구에서 렌탈 서비스 도메인에 대한 비즈니스 요구사항 분석은 크게 기능적 요구사항 분석과 비기능적 요구사항 분석 2가지로 구분된다.

가. 기능적 요구사항 분석(상위수준)

[표 1]은 기능적 요구사항의 상위 수준의 기능적 요구사항을 나타내며, 주요 기능적 요구사항을 정리하였다.

표 1. 기능적 요구사항표(상위레벨)
Table 1. Functional requirement(high level).

요구사항	설명
사용자 인증을 처리한다.	로그인 처리시 사용자 인증을 처리해야 한다.
자동차의 대여 가능 정보를 체크한다.	자동차가 대여 가능한 상태인지 확인 가능해야 한다.
예약을 한다.	대여 가능한 자동차와 일정에 맞게 예약 기능을 제공한다.
렌터카를 대여를 한다.	렌터카를 대여하고 반납하는 기능이 포함되어야 한다.
결재를 한다.	반납 후 결재시스템과 연동하여 결재하는 기능이 포함되어야 한다.

나. 비기능적 요구사항분석

[표 2]는 모바일 제약성 및 보안에 관한 비기능적 요구사항을 나타낸다.

표 2. 비 기능적 요구사항표(상위레벨)
Table 2. Non-functional requirement(high level).

요구사항	설명
모바일 해상도에 맞춘다.	모바일 화면의 크기는 최소한 320*240의 해상도에 맞춘다.
모바일 입력을 최소화 한다.	모바일의 주요 키를 중심으로 설계되어야 한다.
전송되는 용량에 제한을 둔다.	페이지에 전송되는 패킷의 총 크기는 10KB를 넘지 않아야 한다
응답시간을 고려한다.	응답시간이 최대 60초를 초과해서는 안된다.
서비스간 보안 접속을 해야 한다.	보안을 위해 보안 프로토콜을 사용한다.

3. 서비스 식별 및 명세

가. 서비스 식별

[그림 5]는 비즈니스 레벨부터 유스케이스 레벨까지 서비스 식별을 위한 분해를 나타낸다.

서비스 도메인 분석에서는 서비스 식별을 위해 위의 그림과 같이 비즈니스 레벨로부터 분해된 시스템 및 서브 시스템을 서비스 식별을 위해 분해한다. 시스템 분해와 어셋(asset)분석, 모바일기반 기술적 환경분석 그리고 업계표준항목과 관련하여 서비스를 식별한다.

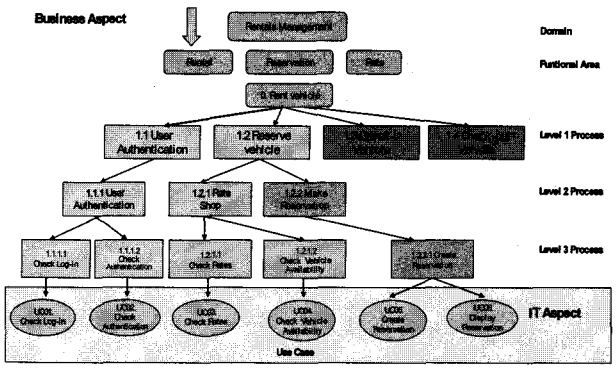


그림 5. 서비스 식별을 위한 분해
Fig. 5. Decomposition for service identification.

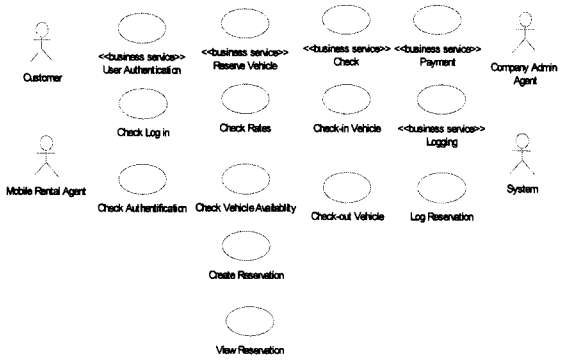


그림 6. 서비스 유스케이스 모델링
Fig. 6. Use case modeling of service.

표 3. 서비스 유스케이스와 액터 식별표
Table 3. Table of service use case and actor identification.

Use Case	Actor
UC1: User Authentification	
UC2: Check Rates	Customer
UC3: Check Vehicle Availability	
UC4: Create Reservation	Company
UC5: Display Reservation	Admin agent
UC6: Check-out Vehicle	
UC7: Check-in Vehicle	Mobile Partner
UC8: Process Payment	agent
UC9: Log Reservation Activity	System
UC10: View Reservation Activity	

식별된 서비스의 비즈니스 서비스 유스케이스와 서비스 유스케이스 액터는 [그림 6]과 같다. 비즈니스 서비스 유스케이스는 <<business service>> 스테레오타입으로 나타나고 관련 서비스 유스케이스는 각각 비즈니스 유스케이스 아래 나타난다.

[표 3]은 식별된 서비스 유스케이스와 액터를 나타낸다.

나. 서비스 명세

[표 4]는 서비스 'Display Reservation'에 대한 서술식 유스케이스 명세이다.

표 4. 'Display Reservation' 유스케이스에 대한 서술식 명세
Table 4. Narrative specification of 'Display Reservation' use case.

Use case	UC05. Display Reservation
Brief description	이 유스케이스는 Customer 예약을 마치고 예약에 대한 정보를 확인 할 때 보여준다.
Actors	Customer
Preconditions	Customer는 예약 확인 위치해 있어야 한다. 그 페이지는 대여 자동차 정보와 예약의 세부 정보를 보여준다.
Main flow	M1. 이 유스케이스는 예약에 대한 세부 정보가 스크린 상에 표시될 때 '확인' 기능을 선택함으로써 Customer에게 예약정보를 보여줄때 시작된다. : M1.3 예약을 확인해주기 위해 시스템은 Customer에게 모든 주문 세부 정보와 더불어 예약 번호 및 사용자 정보를 화면에 보여준다.
Alternative flow	
Postconditions	사용자정보와 세부 예약정보가 화면에 표시된다.

다. 유스케이스 테스팅을 위한 유스케이스 명세

모바일 제약 조건을 테스팅 하기위한 유스케이스 테스팅을 위한 명세는 [표 5]와 같다. 모바일 서비스 테스팅을 위한'Create Reservation'유스케이스를 확장하여 명세하였고 모바일 제한적인 상황을 테스팅 하기위해 화면정보 제한 및 응답시간과 전송패킷 사이즈를 제한하여 확장하였다. 본 확장된 유스케이스는 모바일 제약사항에 대한 스트레스 테스팅과 의사결정테이블 테스팅에 적용한다.

표 5. 모바일 테스팅을 위한 확장된 'Create Reservation' 유스케이스 명세
Table 5. Extended use case specification of 'Create Reservation' for mobile testing.

Use case	UC003. Create Reservation		
Brief Description	이 유스케이스는 Customer가 입력한 예약정보를 받아 예약을 한다.		
Actors	Customer		
Service Type	Provider		
Preconditions	Customer는 예약정보 확인 페이지에 위치해 있어야 한다. 그 페이지는 총 가격과 함께 예약된 세부 정보를 보여준다.		
Main flow	M1. 이 유스케이스는 주문에 대한 세부 정보가 스크린 상에 표시될 때 예약하기를 선택함으로써 구성된 자동차 예약을 Customer가 결정할 때 시작된다.:		
Extensions	Mobile Limitation List	Action	Alternative Action
	Mobile Display limitation	None	None
	Response Time	60seconds or less	Reload
	Packet Size limitation	10Kb or less	Confirm
Reference Service	Display Reservation		
Alternative flow	A1. Customer가 필수 정보들을 모두 입력하기 전에 Send 버튼을 활성화한다면 시스템은 에러 메시지를 출력하고 누락된 정보를 입력해 줄 것을 요청한다. A2.Customer가 예약 양식의 빈곳을 채우기 위해 Reset 기능을 선택하면 시스템은 Customer가 정보를 다시 재입력 하는 것을 허용한다.		
Postconditions	만약 유스케이스가 성공적이었다면 예약은 시스템의 데이터베이스에 저장되며 그렇지 않으면 시스템의 상태는 변하지 않는다.		

라. 서비스 상호운용성을 위한 상호운영 테스트 케이스 설계

SOA의 서비스는 서비스간 상호운용성을 보장해야 한다. 이를 위해 본 연구에서는 [13]의 연구를 확장하여 서비스 상호운영 테스팅에 적용하였다.

참고문헌 [13]의 상호운영을 위한 EFSM의 구성 요소는 다음과 같이 정의된다.

```
From-State
{Input}{Output}{Predicates}{Actions}{Color}To-State
```

상호운용성 테스트명세를 위해 서비스 컴포넌트를 구분하고 [표 6]과 같이 속성식별 테이블을 작성한다. 속성 식별 테이블에서 속성 값에 따른 행위를 명세하고 이를 기반으로 [표 7]과 같이 상태전이 테이블을 작성한다.

[그림 7]은 EFSM 명세의 한 예를 보이고 있으며, 규칙을 적용하여 EFSM 명세를 상태 전이 다이어그램(state transition diagram)으로 표현한 것이다.

EFSM 명세와 같이 전체전인 EFSM명세를 통합하여 기술한 다음 테스트 시퀀스를 생성한다.

S4 → S11

S4 → S13

위와 같은 상호운영 시퀀스 테스트를 위하여 상태전이 테이블을 작성한다.

[표 8]의 상태전이 테이블을 이용하여 최종 테스트를 위해 상호운용성을 위한 테스트 케이스를 명세한다.

표 6. 'Display Reservation'유스케이스로부터 속성식별 테이블

Table 6. Table of attribute identification about 'Display Reservation' use case.

Step No	Corres. Service	Behaviors	AC	Attributes
M1	C1	M1. 이 유스케이스는 예약에 대한 세부정보가 스크린 상에 표시될 때 '확인' 기능을 선택함으로써 Customer에게 예약정보를 보여 줄 때 시작된다.	없음	clickedReservation_C1 displayReservation_C1
:	:	:	:	:

표 7. 'Display Reservation' 속성에 따른 속성 값 테이블

Table 7. Table of attribute value about 'Display Reservation' attribute.

Service ID	Attributes	Attributes Values		
C1	clickedReservation_C1	T	F	F
	displayReservation_C1	F	T	F
:	:	:	:	:
State No.		S4	S13	S11

```
S4
    {}{예약정보를 화면에 표시함}
    {clickedReservation_C1=true;
reqResInfoToC7_C1=true;sendResInfoToC1_C7=true}
    {'Create Reservation '페이지 의 'Send'를 선택함}{black}
    S13
S13
    {}{예약정보를 화면이 표시됨}
    {dispReservationIdel_C1=true}
    {}{white}
    S11
```

그림 7. EFSM 명세의 예

Fig. 7. Example of EFSM specification.

표 8. 상태 전이 테이블의 예

Table 8. Example of state transition table.

No	From State	Precondition	Input	Postcondition	To State
T01	S1	dispLoginPageIdel_C1=true	String name String password	filledLoginInfo_C1=true	S2
T02	S2	filledLoginInfo_C1=true	String name String password	A_authSuccess=true filledLoginInfo_C1=true clickedSubmit_C1=true reqCustInfoToC2_C1=true checkIdPsw_C1=true sendCustInfoToC1_C2=true createReservation_C3	S3
…	…	…	…	…	…

표 9. 테스트 케이스 명세

Table 9. Test case specification.

Test Case Identifier	T05	
State Transition	S4→ S13	
Test Item	Use case ID.	UC03. Display Reservation
	Interoperability	C1과 C7간의 예약정보 요청 및 요청정보 제공
	Behavior	- 'Display Reservation' 페이지의 'send' 버튼을 클릭함
Input	없음	
Output	예약 정보를 화면에 표시	
Procedural requirements	고객은 예약정보 보기에 위치해 있어야 함	
Intercase dependencies	T01, T02, T03 (사용자인증 을 거쳐야 함)	

4. 모바일 서비스 구현 및 테스팅

가. 모바일 서비스 구현

[그림 8]은 'viewReservation'의 서비스 타입, 메시지 정보, 포트 타입 정의, 파트너 링크 타입 정의 정보를 담고 있는 wsdl파일이다.

[그림 9]는 실제 비즈니스 프로세스를 실행하는 bpel 파일을 보여준다. bpel파일은 프로세스를 실행하기 위한 네임스페이스, 비즈니스 파트너와 변수가 선언되고 오케스트레이션을 위한 로직이 설계된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ViewReservation"
targetNamespace="http://210.117.172.202/ViewReservation"...">
        <types>
:               </types>
    <message name="ViewReservationRequestMessage">
        <part name="payload"
element="client:ViewReservationProcessRequest"/>
    </message>
    <message name="ViewReservationResponseMessage">
        <part name="payload"
element="client:ViewReservationProcessResponse"/>
    </message>
    <portType name="ViewReservation">
        <operation name="initiate">
            <input
message="client:ViewReservationRequestMessage"/>
        </operation>
    </portType>
    <portType name="ViewReservationCallback">
        <operation name="onResult">
            <input
message="client:ViewReservationResponseMessage"/>
        </operation>
    </portType>
    <plnk:partnerLinkType name="ViewReservation">
        <plnk:role name="ViewReservationProvider">
            <plnk:portType
name="client:ViewReservation"/>
        </plnk:role>
        <plnk:role name="ViewReservationRequester">
            <plnk:portType
name="client:ViewReservationCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

그림 8. viewReservation.wsdl
Fig. 8. viewReservation.wsdl.

```
<process name="ViewReservation"
targetNamespace="http://210.117.172.202/ViewReservation"
... ">
  <partnerLinks>
    <partnerLink name="CreateReservation"
                 partnerLinkType="client:ViewReservation"
                 myRole="ViewReservationProvider"
partnerRole="ViewReservationRequester"/>
    <partnerLink myRole="ViewReservationProvider"
                 name="System"
                 partnerRole="ViewReservationRequester"
partnerLinkType="client:ViewReservation"/>
  </partnerLinks>
    <variable name="uid"
messageType="client:ViewReservationResponseMessage"/>
    <variable name="reservID" type="ns1:int"/>
    <variable name="reservCar" type="ns1:string"/>
:
<!-- ORCHESTRATION LOGIC   -->
:
     <assign name="asign_uid_to_requestUid">
      <copy>
        <from variable="uid"/>
        <to variable="requestUid"/>
      </copy>
    </assign>
:
</process>
```

그림 9. viewReservation.bpel
Fig. 9. viewReservation.bpel.

나. 모바일 화면 구현

[그림 10]은 모바일 환경을 구현한 화면으로 가상 시뮬레이션 툴을 사용하여 예약정보보기 (View Reservation)서비스를 구현하였다.

그림 10. 예약정보보기(View Reservation)서비스를 구현한 그림
Fig. 10. Screen of 'view Reservation' service.

다. 테스팅 결과

테스트 케이스의 품질을 완전성(completeness)과 테스트 케이스의 최적화(optimization) 정도를 통하여 정의한다. 테스트 케이스의 완전성은 주어진 테스트 케이스를 통하여 얼마나 많은 오류(error) 검출(detect)을 해낼 수 있는가 하는 척도(measurement)이며, 테스트 케이스(test case)의 최적화는 테스트 목적에 상관없는 테스트 케이스가 얼마나 많이 포함되지 않았는가를 나타낸다.

[표 10]은 측정한 결과로써 높은 수치의 완전성과 테스트 케이스의 최적화 정도를 보이고 있다.

표 10. 상호운용성 테스팅 결과
Table 10. Result of Interoperability testing.

완전성	*상호운용 메소드 수*	*테스트 케이스가 포함하는 메소드 수*
	27	24
테스트 케이스 최적화 정도	*알고리즘 적용 전 전이 수*	*알고리즘 적용 후 전이 수*
	22	11

V. 결 론

모바일기기의 성능 향상과 함께 이용자가 증가하고 확대됨에 따라 엔터프라이즈와 정보기술의 관심은 서비스를 기반으로 양자 간의 관심사를 통합 할 수 있는 구조에 많은 노력을 기울여 왔다. 이러한 노력은 컴포넌트 기반 개발을 넘어 SOA를 통해 결실을 맺을 수 있었고 SOA를 활용하여 엔터프라이즈와 정보기술의 성과를 향상시킬 수 있다고 볼 수 있다. 본 논문은 모바일의 이동성과 SOA의 편의성을 위해 모바일을 위한 SOA 서비스를 구현하고 테스팅 하였다.

본 연구의 구현 및 테스팅의 의의는 다음과 같다.

첫째, 비즈니스 레벨부터 시스템 영역까지 모바일을 적용하기 위한 설계의 과정을 제시하였고,

둘째, 이를 위해 모바일의 제약사항을 검토하고 서비스 명세에 반영하여 테스트를 위한 명세를 유도하였으며, 셋째, 유도된 테스트 케이스를 구현하여 최종 테스팅 함으로써 모바일을 위한 SOA 서비스 구현 및 테스팅에 대한 현실성 있는 활용 가능성에 대해 제시했다. 또한 유스케이스 테스팅에 있어 유스케이스 확장 명세를 통해 보다 구체적인 테스팅에 대한 아이디어를 제공하였고, 상호운용성 테스팅을 통해 보다 최적화되고 안정된 구조를 테스트할 수 있는 실험적 토대를 마련하였다는데 그 의의가 있다.

본 연구와 관련하여 향후에는 SOA 구조에서 특수화된 도메인을 위한 보다 정형적으로 명세 된 유스케이스 테스팅 스펙에 관한 연구가 필요하다고 본다.

참 고 문 헌

[1] Allam, A.; Arsanjani, A., "Service-Oriented Modeling and Architecture for Realization of an SOA", Services Computing, 2006. SCC '06. IEEE International Conference on pp. 521-521, September 2006.

[2] Alarcon, M.P.; Fuentes Fernandez, L.; Troya Linero, J.M., "Using MDA to develop Component and Aspect Based Applications", Latin America Transactions, IEEE (Revista IEEE America Latina), Volume 3, pp. 1-1, 2005.

[3] R.M. Dijkman, and S.M. Joosten, "Deriving Use Case Diagrams from Business Process Models", 2002.

[4] S. Stolfa, I Vondrak, "Using the Business Process for Use Case Model Creation", ISIM '03, 2003.

[5] A. Brown, "An introduction to Model Driven Architecture Part I: MDA and today's systems", IBM, January 2004.

[6] S. Mellor, K. Scott, A. Uhl, D. Weise, "MDA Distilled: Principles of Model-Driven Architecture", Addison-Wesley, 2004.

[7] Object Management Group, "Extensible Metadata Interchange", http://omg.org/technology/xml/

[8] T. Gardner "UML Modeling of Automated Business Processes with a Mapping to BPEL4WS", In Proceedings of the European Workshop on Web Services and Object orientation, ECOOP July 2003.

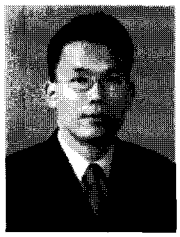
[9] Do Van Thanh; Jorstad, I., "A Service-Oriented Architecture Framework for Mobile Services", Telecommunications, 2005. Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop. AICT/SAPIR/ELETE 2005. Proceedings, pp. 65-70, July 2005.

[10] Glaschick, R.; Oesterdieckhoff, B.; Loeser, C., "Service Oriented Interface Design for Embedded Devices", Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on Volume 2, pp. 8-8, September 2005.

[11] Hans-Gerhard Gross, Component-Based Software

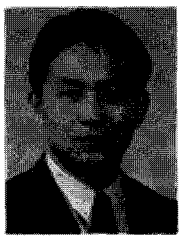
Testing with UML, Springer, 2004.
[12] N. Griffeth, R. Hao, D. Lee, R. K. Sinha, "Interoperability Testing of VoIP Systems", Global Telecommunications Conference, Vol. 3, pp. 1565-1570, 2000.
[13] Kwang Ik Seo; Eun Man Choi; "Comparison of Five Black-box Testing Methods for Object-Oriented Software", Software Engineering Research, Management and Applications, 2006. Fourth International Conference on pp. 213-220, August 2006.
[14] Hye-Min Noh, Ji-Hyun Lee, Cheol-Jung Yoo, and Ok-Bae Chang. "Behavior Modeling Technique Based on EFSM for Interoperability Testing", ICCSA 2005, LNCS 3482. pp. 878-885, 2005.
[15] Z. Gao. H.-S.J. Tsao, and Y. Wu. Testing and Quality Assurance for Component-Based Software Engineering, Artech House, 2003.
[16] IEEE. IEEE Standard for Software Test Documentation, IEE Std 829. 2000.
[17] Margaria, T.; Steffen, B. "Service Engineering: Linking Business and IT", Computer,Volume 39, pp. 45-55, October 2006.

저 자 소 개



장 영 원(학생회원)
1998년 전주대학교 회계학과 학사 졸업.
2003년 전북대학교 컴퓨터 정보학과 석사 졸업.
2006년 전북대학교 공과대학 컴퓨터공학과 박사과정.
<주관심분야 : 소프트웨어공학, 테스팅, 에이젼트, 인지공학, 로보틱스 등임>

노 혜 민(정회원)
2000년 전북대학교 컴퓨터과학과 학사 졸업.
2002년 전북대학교 대학원 전산통계학과 석사 졸업.
2006년 전북대학교 대학원 컴퓨터통계정보학과 박사 졸업
<주관심분야 : 컴포넌트 기반 소프트웨어 개발, 정형 명세 기법, 소프트웨어 테스팅, LBS 등임>



유 철 중(정회원)-교신저자
1982년 전북대학교 전산통계학과 학사 졸업.
1985년 전남대학교 대학원 계산통계학과 석사 졸업.
1994년 전북대학교 대학원 전산통계학과 박사 졸업.
<주관심분야 : 소프트웨어 개발 프로세스, 소프트웨어 품질, 컴포넌트 소프트웨어, 소프트웨어 메트릭스, 소프트웨어 에이전트, 인지공학 등임>