DOI: 10.3745/KIPSTD.2009.16-D.4.583

레거시 시스템의 웹서비스화를 위한 마이그레이션 기법

박 옥 자[†] · 최 시 원[††] · 김 수 동[†††]

요 약

현재 기업이나 조직에서 적용할 수 있는 SOA(Service-Oriented Architecture, 이하 SOA)는 조직의 요구사항을 분석하여 비즈니스 모델을 설정하고 후보 서비스를 식별하여 필요한 서비스를 찾거나 개발하는 하향식 방법론(top-down methodology)이 대부분이다. 이 경우 기존의 시스템을 버리고 새로운 SOA를 도입하기에는 비용과 시간면에서 모험이 따를 수 있으므로 기존의 시스템을 최대한 이용하면서 점차적으로 SOA에 적합한 시스템으로 변환하기를 원한다. 본 논문에서는 기존의 레거시 시스템을 SOA에 적합한 웹서비스 형태로 만들기 위한 M-LSWS (A Method for Migration of Legacy System into Web Service)를 제안한다. M-LSWS는 레거시 시스템이 가지고 있는 디자인 명세 및 코드를 기반으로 비즈니스 프로세스를 분석하고 후보 서비스를 식별하여 재사용 가능한 웹서비스로 변환하는 절차를 정의하였으며 실제 SOA에 적합한 웹서비스 변환에 목적을 두고있다. 제안한 방법은 레거시 시스템 분석, 재사용 가능 서비스 도출 및 명세, 서비스 래핑, 서비스 등록의 네 단계로 이루어지며 단계별 프로세스와 가이드라인에 제시되고 도서관리 시스템에 적용함으로써 제안한 방법론의 타당성을 평가하여 본다.

키워드 : 레거시 시스템, 웹서비스, 서비스 지향 아키텍쳐, 마이그레이션

A Method for Migration of Legacy System into Web Service

Oak Cha Park[†] · Si Won Choi[††] · Soo Dong Kim[†††]

ABSTRACT

Most of the SOA solutions applicable to businesses and organizations are taking a top-down methodology. It starts with an analysis of an organization's requirements, followed by definition of business models and identification of candidate services and ends with finding or developing required services. Challenges in adopting SOA while abandoning legacy systems involve time and cost required during the process. Many businesses and organizations want to gradually migrate into SOA while making the most of the existing system.

In this paper, we propose A Method for Migration of Legacy System into Web Service(M-LSWS); it allows legacy system to be migrated into web service accessible by SOA and be used as data repositories. M-LSWS defines procedures for migration into reusable web services through analysis of business processes and identification of candidate services based on design specification and code of legacy system. M-LSWS aims to migrate of legacy system into web service appropriate for SOA. The proposed method consists of four steps: analysis of legacy system, elicitation of reusable service and its specification, service wrapping and service registration. Each step has its own process and guideline. The eligibility of the proposed method will be tested by applying the method to book management system.

Keywords : Legacy System, Web Service, Service-Oriented Architecture, SOA, Migration

1. 서 론

SOA[1]는 잘 정의된 인터페이스와 서비스(service)들 간 규약(contracts)를 통해 서비스라고 하는 애플리케이션의 다양한 기능 단위를 상호 연관시키는 컴포넌트 모델(component model)이다. 따라서 기업 인프라의 복잡성 및 유지비용을 최소화하고 기업의 생산성과 유연성을 극대화할 것으로 기대되어 산업계에서 꾸준한 사용이 증가하고 있다. 하지만 현재 조직에서 적용할 수 있는 SOA는 조직의 요구사항을 분석하여 비즈니스 모델을 설정하고 후보 서비스를 식별(identification)하여 필요한 서비스를 찾거나 개발하는 하향식 방법론이 대부분이다. 이 경우 기업 또는 조직에서는 기존의 시스템을 버리고 새로운 SOA를 도입하기에는 비용과 시간면에서 모험이 따를 수 있다. 따라서, 기존의 레거시 시스템이 가지고 있는 핵심 내용을 변경할 필요없이 그것들이 가지고 있는 기능(functionality)만 노출시킴으로써 서비스로 변환해 갈 수 있는 마이그레이션(migration)[1] 방법을 필요

※ 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-005-J03803).
† 정 회 원 : 숭실대학교부속 정보미디어기술연구소 전임연구원
†† 정 회 원 : 숭실대학교부속 정보미디어기술연구소 연구교수
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2009년 6월 9일
심사완료 : 2009년 7월 7일

로 한다.

본 논문에서는 조직의 레거시 시스템을 웹서비스(web service)로 변환하기 위한 M-LSWS (A Process for Migration of Legacy System into Web Services)을 제안한다. 웹서비스는 SOA를 구현하기 위해 선호되는 표준 기반 방법으로서 네트워크상에서 상호운용 가능한 기계와 기계 사이의 상호작용을 지원하기 위해 설계된 소프트웨어 시스템이다[2]. SOA는 일반적으로 웹서비스를 통해 실현되므로 레거시 시스템을 SOA로 마이그레이션하기 위해서는 먼저 SOA에 적합한 웹서비스를 변환해야 한다. 따라서 본 논문은 초기 연구에서 웹서비스 변환에 목적을 둔 M-LSWS를 제안하고 차후에 확장하여 SOA 마이그레이션 절차를 수행하고자 한다. M-LSWS는 레거시 시스템의 디자인 명세 및 코드를 기반으로 비즈니스 프로세스를 분석한 후 후보 서비스를 식별(identification)하여 재사용 가능한 웹서비스로 래핑(wrapping)하는 단계로 이루어져있다. 기본적인 원칙은 기존 시스템의 핵심 내용을 변경할 필요없이 그것들이 가지고 있는 기능(function)만 노출시킴으로써 하나의 웹서비스로 변환하는 것이다. 제안한 방법은 레거시 시스템 분석, 재사용 가능 서비스 도출 및 명세, 서비스 래핑, 서비스 등록의 네 단계로 이루어지며 단계별 상세 절차와 가이드라인을 정의하고 도서관리 시스템에 적용함으로써 제안한 방법론의 타당성을 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 살펴보고 3장에서는 M-LSWS의 방법론에 대하여 기술한다. 4장과 5장에서는 제안한 방법론을 실제 시스템에 적용하여 평가하여 보고 6장에서 결론을 맺는다.

2. 관련 연구

Kulkarani[3]은 레거시 시스템 마이그레이션을 위하여 아키텍쳐 중심의 InSOAP을 제안하였다. InSOAP은 레거시 시스템에서 비즈니스 프로세스를 적절한 서비스의 개념적 크기에 맞게 분류하도록 제안하였으며 제안한 비즈니스 서비스로 실체화(realization)하기 위하여 서비스, 데이터 흐름, 컴포넌트와 같은 서비스 및 기능의 흐름등을 결정하는 지침(guideline)을 정의하였다.

Belushi[4]와 Sneed[5]는 레거시 애플리케이션으로부터 추출한 여러 비즈니스 로직을 웹서비스로 변환하기 위해 필요한 래핑방법에 초점을 제안하였다. 래핑 방법을 세션 기반(session-based), 트랜잭션 기반(transaction-based), 데이터 기반(data-based)의 세 가지 유형으로 분류하였으며 세 가지 유형의 장점을 결합한 하이브리드(hybrid) 방법론을 제안하였다. 프로세스는 서비스 추출, 웹서비스 구축, 웹서비스 래퍼(wrapper) 작성의 절차로 이루어져있으며 서비스 추출 단계에서 추출한 서비스를 웹서비스 구축 단계에서 각 서비스의 유형에 적합한 래퍼를 정의하고 이를 웹서비스 래퍼 작성단계에서 실체화하는 절차로 이루어진다. 하지만 레거시 시스템으로부터 서비스를 추출하는 기준을 명확하게 정의하지 않았고 실제 웹서비스 구축 부분만 기술하였기 때문에 초기 적용단계에서 어려움이 있다. Sneed[5]는 SOA의 상향식 방법과 하향식 방법(bottom-up methodology)을 결합한 하이브리드 방식(hybrid method)의 웹서비스 마이그레이션을 제안하였다. 특히 레거시 애플리케이션이 실제 가지고 있는 블랙박스(blackbox)화된 기능을 웹서비스 형태로 이용하기 위하여 래퍼(wrappers), 어댑터(adaptors), 프록시(proxies)와 같은 패턴을 사용하였다.

Wang[6]은 레거시 시스템은 각 모듈간의 비즈니스 로직이 강한 결합력을 가지고 있으므로 SOA에 적합한 서비스가 되도록 비즈니스 로직과 데이터를 노출시키는 방법을 제안하였으며 레거시 시스템이 가지고 있는 데이터 정의와 SOA 규약 간의 갭(gap)을 줄이기 위하여 브리지(bridge) 적용 방법을 제안하였다. Lewis[7]에서는 SMART(The Service Migration and Reuse Technique)를 제안하였다 이 중 레거시 시스템을 SOA 환경에 적합한 프로그램으로 마이그레이션하는 초기 결정단계부터 개발단계까지의 SMART-MP (Migration Planning)를 네 항목으로 분류하였다. SMART MP는 레거시 시스템을 분석하여 후보 서비스를 추출하고 대상 SOA 환경을 기술하여 현 레거시 환경과 SOA 환경간의 갭을 조절하여 마이그레이션 전략을 세우는 절차로 이루어져있다.

위와 같이 기존의 방법론은 레거시 애플리케이션에서 서비스 가능한 비즈니스 로직의 크기를 결정하는 방법 또는 추출한 후보서비스의 래핑방법에 대해 초점을 맞추었다. 이외에도 마이그레이션을 프로세스로 나누어 각 단계에서 이루어지는 업무나 특징에 대해서 정의한 방법론도 있지만 상세화된 액티비티(activity)나 지침이 부족하여 실제 적용하는 데는 어려움이 있다. 본 논문에서는 이와 같은 문제점을 보완하는 마이그레이션 기법을 제안하고자 한다.

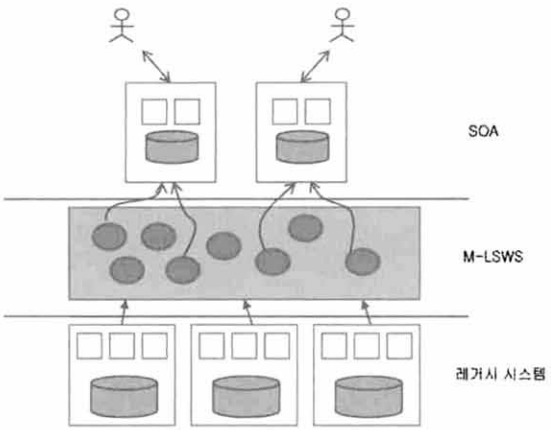
3. M-LSWS : 웹서비스를 위한 마이그레이션 프로세스

본 장에서는 본 논문에서 제안한 M-LSWS에 대하여 기술한다. 3.1절에서는 M-LSWS 시스템 아키텍쳐에 대하여 정의하고 3.2절에서는 M-LSWS의 실제 프로세스를 단계별로 정의한다.

3.1 시스템 아키텍쳐

(그림 1)은 M-LSWS의 시스템 아키텍쳐이다. 레거시 시스템을 분석하여 서비스 후보를 추출하고 이를 웹서비스로 변환하여 SOA를 이용한 애플리케이션 구축을 가능하게 한다. 서비스 후보는 레거시 시스템이 가지고 있는 다자인 명세와 소스코드와 같은 가용산출물(artifacts)을 분석하여 이들로부터 하나 또는 그 이상의 서비스 후보를 추출하여 웹서비스로 변환하는 절차로 이루어진다.

(그림 2)는 본 논문에서 M-LSWS의 DFD(date-flow iagram)를 나타낸다. 레거시 시스템 분석 단계에서는 디자



(그림 1) M-LSWS 시스템 아키텍처

인 명세, 소스코드, 실행파일과 같이 레거시 시스템이 관리하고 있는 가용 산출물을 분석하여 비즈니스 프로세스(business process)를 찾아내어 명세서를 작성한다. 재사용 가능한 서비스 도출 및 명세 단계에서는 후보서비스(candidate service)를 추출하고 각 서비스의 명세서(specification)를 작성한다. 서비스 래핑단계에서는 후보 서비스를 실제 실행가능한 웹서비스로 변환하기 위한 래핑을 수행한다. 래핑은 후보 서비스와 관련된 레거시 코드를 클러스터링(clustering)한 후 외부 인터페이스를 작성하여 관련된 레거시 코드를 접근할 수 있도록 작성한다. 이 후 래핑이 끝난 후보서비스는 실제 서비스로 동작할 수 있는 WSDL(Web Service Description Language)을 작성하여 UDDI(Universal Description Discovery, and Integration)등록하게 된다[2]. M-LSWS 네 단계의 상세 프로세스는 다음과 같다.

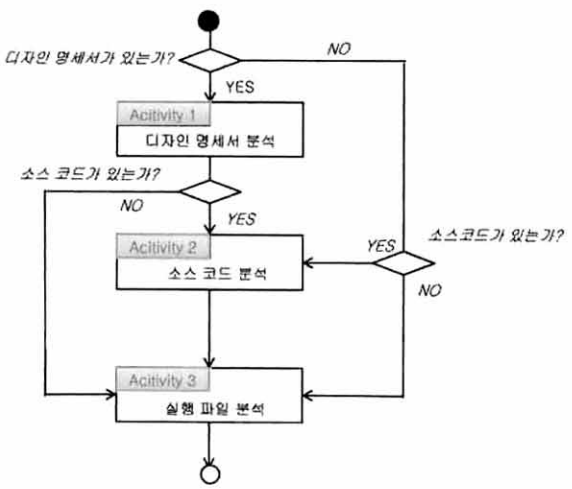
3.2 M-LSWS : 상세 프로세스

3.2.1 레거시 시스템 분석

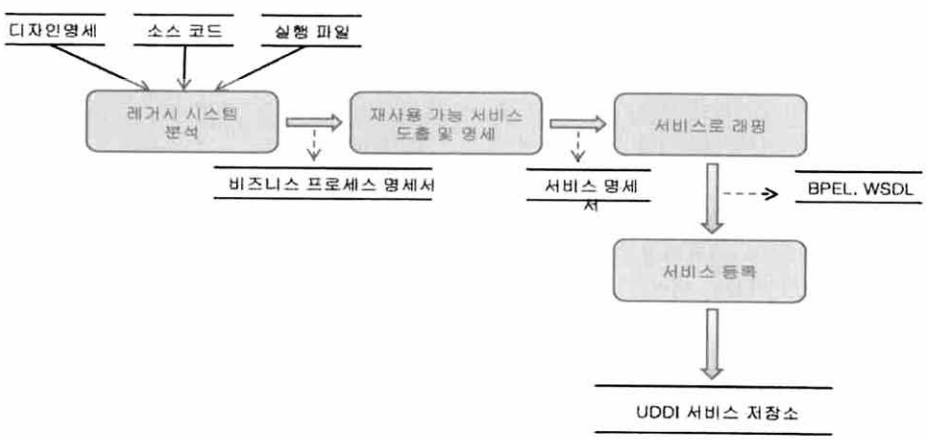
이 절에서는 레거시 시스템을 분석하는 단계이다. 본 논문에서는 레거시시스템의 가용산출물을 디자인 명세서, 소스코드, 실행파일과 같이 크게 세 가지로 구분하였다. 디자인명세서(Documentation)는 요구사항 명세서(Requirement Specification), 데이터 흐름도(Data Flow Diagram, 이하 DFD), ER 다이어그램(Entity-Relation Diagram), 클래스 다이어그램(Class Diagram)등 디자인 단계에서 생성되는 전반적인 모든 자료를 포함한다. 소스코드는 디자인 명세서를 기반으로 작성된 코드이며 실행파일은 실제 동작하는 프로그램으로 일반적으로 배치파일(batch file) 형태로 제공되거나 GUI 형태로 제공된다. 이와 같은 가용산출물은 기존의 레거시 시스템이 모두 관리해야 하지만 그렇지 않고 소스코드와 실행파일 또는 디자인 명세와 실행파일과 같이 일부부만 관리될 수 있기 때문에 본 논문에서는 가용 산출물에 따른 분석과정을 (그림 3)과 같이 액티비티(activity)로 분리하였다.

(그림 3)은 레거시 시스템의 가용산출물의 유형에 따른 분석 액티비티를 보여준다. 하나의 액티비티는 분석자료에 따라 이루어지는 활동과정으로서 디자인명세서(Documentation, 이하 *Doc*), 소스코드(Source code, 이하 *Soc*), 그리고 실행파일(Execution file, 이하 *Exe*)에 따라 액티비티의 흐름을 나타낼 수 있으며 이를 <표 1>과 같이 *CASE* 별로 정의하였다.

CASE 1 은 *Doc*와 *Soc* 모두 가지고 있는 경우이다. *Doc*



(그림 3) 레거시 시스템의 가용산출물에 따른 프로세스 유형



(그림 2) M-LSWS DFD

〈표 1〉 레거시 시스템의 가용 산출물에 따른 분석 *CASE*

마이그레이션 CASE	레거시 시스템의 가용 산출물	액티비티 흐름	분석율
CASE 1	*Doc* ∩ *Soc*	*Activity* 1 -> *Activity* 2 -> *Activity* 3	높음
CASE 2	*Only Soc*	*Activity* 2 -> *Activity* 3	중간
CASE 3	*Only Doc*	*Activity* 1 -> *Activity* 3	낮음
CASE 4	*Only Exe*	*Only Activity* 3	낮음

Doc : 초기분석자료
Soc : 소스코드
Exe : 실행파일

는 시스템 전체의 분석자료이기 때문에 프로그램에 익숙하지 않은 사용자라도 *Doc*만으로 시스템의 요구사항 또는 비즈니스 프로세스를 분석할 수 있다. *Soc*는 *Doc*에서 추출한 비즈니스 로직에 따라 실제 래핑단계에서 적용되기 때문에 서비스 코드로 변환하기 위하여 사용된다. *Soc*가 *Doc*를 명세에 따라 명확하게 구현된 프로그램이라면 시스템 분석과정에서 시간과 비용을 줄일 수 있으며 <표 1>에서와 같이 분석율이 높아지게 된다. *CASE* 2는 *Doc*는 없고 *Soc*만 있는 경우이다. *CASE* 1에 비하여 레거시 시스템 분석에 더 많은 시간이 소요될 수 있다. 하지만 소스코드 분석만으로도 비즈니스 프로세스를 추출할 수 있으므로 마이그레이션은 가능하지만 *CASE* 1에 비해서는 분석율이 낮아질 수 있다. *CASE* 3는 *Doc*만 있고 *Soc*가 없는 경우로 디자인 명세와 실제 실행되는 프로그램만 가지고 레거시 시스템을 분석하여야하므로 비즈니스 핵심 기능이나 비즈니스 프로세스를 추출은 가능하다. 하지만, 이 경우 소스코드가 없으므로 코드를 웹서비스로 변환하는 것은 불가능하고 비즈니스 프로세스에 적합한 서비스로 교체(replacement)하는 방법이 효율적이므로 본 논문에서는 다루지 않는다. *CASE* 4는 오직 Exe만 있는 경우이다. *CASE* 3와 마찬가지로 실행파일을 가지고 레거시 시스템을 분석하는 것은 매우 어렵다. 따라서 실행파일만을 기반으로 하는 마이그레이션은 SOA로 대체하는 방법보다 오히려 시간과 비용이 낭비되는 문제를 초래할 수 있게 있게 되므로 *CASE* 3과 같이 실행단계에서 이루어지는 비즈니스 프로세스만 추출하여 관련된 후보 서비스를 명시한 후 이를 다른 서비스로 교체 또는 조합하는 과정을 거쳐 새로운 서비스로 전환하는 방법이 훨씬 효율적이므로 본 논문에서는 다루지 않고 M-LSWS는 위의 네 가지 유형 중 CASE 1과 CASE 2에 적합한 프로세스를 정의한다.

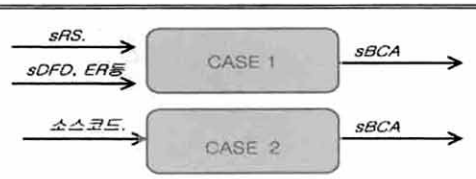
다음은 *Doc*와 *Soc*를 자료를 분석하여 레거시 시스템으로부터 서비스 후보를 추출하기 위한 절차이다. 이 단계에서는 <표 1>에서 제시한 *CASE* 중 마이그레이션 가능성이 높은 *CASE* 1과 *CASE* 2의 단계별 절차를 수행한다. *CASE* 1과 *CASE* 2는 별도의 프로세스로 유지되면서 병행적으로 수행할 수 있다. 상세 프로세스 이해를 위하여 다음과 같은 용어를 정의한다.

RS (Requirement Specification): 요구사항 명세서
sRS : RS 집합
DFD(Data Flow Diagram): 데이터 흐름도
sDFD : DFD 집합
BCA(Business Core Asset) : 비즈니스 핵심 기능
sBCA(set of Business Core Asset) : 비즈니스 핵심 기능 집합
ER(Entity-Diagram) : 엔티티 다이어그램
BP(Business Process) : 비즈니스 프로세스
SBP(Specification of Business Process) : 비즈니스 프로세스 명세서
CS(Candidate Service) : 후보 서비스
sCS(set of Candidate Service) : 후보 서비스 집합

Step 1 : 비즈니스 핵심 기능 명시

디자인 명세로부터 비즈니스 핵심 기능을 명시하는 단계이다. SOA는 각 단위 시스템 또는 모듈을 비즈니스 서비스(Business Service)로 부르고 있으므로 레거시 시스템을 웹서비스로 변환하기 위해서는 상호 연동되고 있는 커다란 시스템을 비즈니스 서비스 단위로 분류할 수 있어야 한다. 비즈니스 핵심 기능은 시스템이 가지고 있는 전체적인 기능을 포함하고 있으므로 먼저 비즈니스 핵심 기능을 명시한 후 이를 상세화하면서 비즈니스 서비스 단위를 추출할 수 있게 된다.

서비스 가능한 비즈니스 핵심 기능 추출을 위하여 절차는 다음과 같다. 요구사항 명세 집합 *sRS*는 RS_1에서 RS_n으로 이루어진다. 마찬가지로 *sRS*를 기반으로 작성된 *sDFD*는 DFD_1에서 DFD_n으로 구성된다. 이 중 RS_1은 하나의 *BCA* 매핑될 수 있다. 또한, *sDFD*중 하나의 *DFD*는 *BCA*로 매핑될 수 있다. 따라서 sRS와 *sDFD*를 분석하여 *sBCA*을 추출한다.

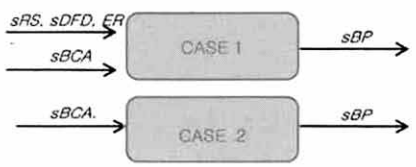
Step 1 : 비즈니스 핵심 기능 명시	
입력물과 산출물	
지침	ST101 : 디자인 명세에서 하나의 *RSx* 또는 *DFDx*는 하나의 *BCAx*와 매핑할 수 있다. 단, *DFDx*는 *RSx*를 데이터흐름도로 변환한 결과이다. ST102 : 소스코드에서 main()함수를 분석하여 main() 함수 내에 나열된 각 sub함수를 하나의 *BCA*와 매핑할 수 있다. ST102 : ST101에서 추출한 *BCA*들을 모아 *SBCA*로 명세한다.

CASE 2의 경우 디자인 명세가 없으므로 소스코드에서 비즈니스 핵심 기능을 추출해야 한다. 코드를 직접 분석하는 것보다 프로그램을 실행하여 나타나는 함수의 호출관계 또는 GUI 상태에서의 초기 화면의 메뉴의 유형에 따라 비즈니스 핵심 기능을 명시할 수 있다.

Step 2 : 비즈니스 핵심 기능으로부터 비즈니스 프로세스를 추출

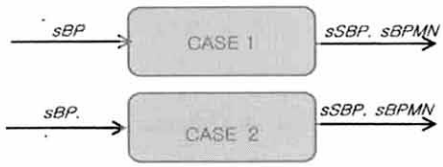
비즈니스 핵심기능(*BCA*)으로부터 비즈니스 프로세스(*BP*)를 추출하는 단계이다. *BP*는 비즈니스 핵심 기능에 대한 업무 흐름을 보여주므로 업무 흐름을 세분화하여 후보서비스를 추출할 수 있게 된다. 먼저 *sBCA*에서 각각의 *BCA*에 해당하는 *BP*를 추출하는 단계이다. *BP*는 *BCA*의 개념적 크기 정도에 따라 소규모(fine-grained) 서비스가 되기도 하고 대규모(coarse-grained) 서비스가 될 수 있다. *BP*를 추출하는 이유는 *BP*를 분석하여 흐름에 따른 새로운 *BP*를 찾아거나 조건분기에 따른 새로운 BP를 찾아 상세화하기 위해서이다. 하나의 *BCAx*로부터 하나의 *BPx*를 유도할 수 있다. *BCAx*는 *RS* 또는 *DFD*를 기반으로 작성되었기 때문에 *RS*와 *DFD*의 흐름을 *BP*로 표현하게 된다. *BP*는 조건에 따라 다른 *BP*로 분기할 수 있고 *BP*에 의하여 호출당할 수 있다. 이 경우 비즈니스 프로세스 명세화 단계에서 분기 또는 호출되는 *BP*는 다시 새로운 *SBP*로 명세한다.

CASE 2의 경우에서는 main() 함수에서 분기되는 각 함수 또는 호출되는 함수의 세부 실행 절차를 하나의 *BP*로 매핑하게 된다. *BCA*와 매핑되는 메뉴를 선택하여 메뉴 실행과정에서 발생하는 데이터 입력에 따른 데이터 흐름이나 다른 기능의 호출 및 분기과정을 분석하면 하나의 *BP*를 추출할 수 있게 된다.

Step 2 : 비즈니스 핵심 기능으로부터 비즈니스 서비스 추출	
입력물과 산출물	
지침	ST201 : *SRx*과 *DFDx*의 흐름을 분석하여 하나의 BPx를 찾아낸다. ST202 : main() 함수에서 호출 또는 분기되는 각 함수의 흐름을 BP로 명시한다. ST203 : *sBP*는 ST201과 ST202로부터 추출된 BP를 분석하여 작성한다.

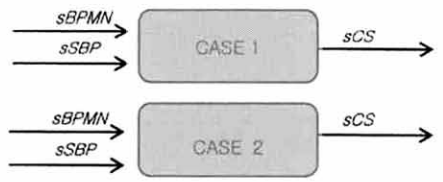
Step 3 : 비즈니스 프로세스 명세

이전 단계에서 추출한 *BP*를 기반으로 하여 *SBP*를 작성한다. *SBP*는 *BP*에 해당하는 비즈니스 프로세스명과 특징, 해당 *BP*의 비지니스 요구사항, 입력데이터, 출력데이터, 예외처리, 그리고 해당 *BP*에 해당하는 요구사항 시나리오등으로 구성된다. 이 단계에서는 BP를 기반으로 BPMN을 작성한다. *BPMN(Business Process Modeling Notation)*[9]은 *BP*를 도식화하여 업무 흐름의 이해를 도와주는 비즈니스 프로세스 분석 방법중의 하나이다. 서비스 재사용 과점에서는 대형크기의 서비스보다 소형크기의 서비스가 더 효율적이기 때문에 *BPMN*을 이용하여 후보서비스를 추출하도록 한다. 하나의 *BP*는 후보 서비스가 될 수 있지만 이를 도식화한 *BPMN*에서 조건에 따라 발생하는 새로운 *BP*는 후보서비스로 명시한다.

Step 3 : 비즈니스 프로세스 명세	
입력물과 산출물	
지침	ST301 : 하나의 *BP*는 하나의 *SBP*와 매핑된다. ST301_1 : 입력데이터는 *BP*를 수행하기 위한 매개변수(parameter)이다. ST301_2 : 출력데이터는 *BP*를 수행한 후의 결과값이다. ST301_3 : 예외사항은 *BP* 프로세스내에서 발생하는 예외적인 경우에 해당 한다. ST302 : *SBP*를 기반으로 하여 *BPMN*을 작성한다.

Step 4 : 후보 서비스 명시

*SBP*와 *BPMN*으로부터 후보 서비스를 추출한다. 후보서비스는 서비스가 될 수 있는 조건을 참조하여 가능한 최소한의 기능을 수행할 수 있는 후보 서비스로 상세화한다[1]. 넓은 관점에서 하나의 *BPMN*은 하나의 후보 서비스가 될 수 있다. 하지만, *BPMN*에서 조건에 따라 새로운 기능을 수행하게되는 모듈을 호출할 경우 호출된 기능은 새로운 후보 서비스가 될 수 있다.

Step 4 : 후보 서비스 명시	
입력물과 산출물	
지침	ST401 : *BPMNx*가 오직 하나의 기능만 수행한다면 후보 서비스 CSx가 된다. ST402 : *BPMNx*에서 조건에 해당하는 분기가 발생할 때를 $Bpmn_y$이라 하자. $Bpmn_y$가 *BPMNx*와 유사 기능을 수행할 경우 $Bpmn_y$는 *BPMNx*에 포함된다. 그렇지 않고 새로운 기능을 호출하게 될 경우 $Bpmn_y$는 또 다른 후보서비스가 될 수 있다. $Bpmn_y$은 후보 서비스 CSy가 될 수 있다. ST403 : 추가된 *CSy*는 이에 해당하는 *SBPy*를 작성한다.

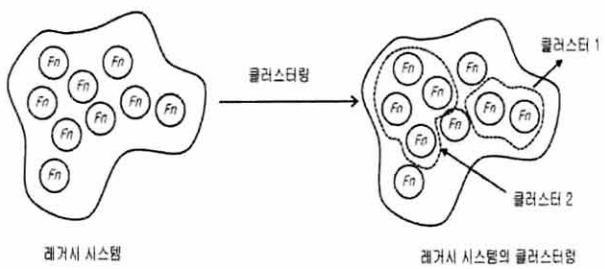
Step 5 : 소스코드 클러스터링

*sCS*와 이에 해당하는 *BP*를 기반으로 후보 서비스에 해당하는 소스코드를 추출하여 클러스터링한다. 함수는 여러 모듈로 이루어진 블록에 산재되어있는 경우가 많다. 이 경우 하나의 함수가 여러 핵심 기능을 처리하는 일대다 또는 다대다의 관계를 가지고 있으므로 데이터 흐름을 분석하여 해당 *BP*에 맞도록 소스를 클러스터링하여야 한다. 이후 클러스터링된 모듈은 고유 인터페이스와 함께 개별모듈로 변환하여 서비스로 래핑하게 된다.

Step 4에서 추출한 후보 서비스 *sCS*는 *sSBP*와 *sBPMN*와 같은 디자인 명세를 기반으로 얻어진 결과이다. Step 5에서는 하나의 *CS*에 해당하는 코드의 흐름을 분석하여 상호 호출 및 참조 관계에 따라 유사한 기능들을 클러스터화한다.

(그림 4)는 *Fn*을 하나의 클래스 또는 함수라 할 때 레거시 코드에서 각각의 기능 *Fn*을 지침 ST502_1에서 ST502_3에 모듈로 클러스터링하는 과정을 보여주다. 하나의 클러스터내에서 이루어지는 지침 ST502와 같이 *CS*에 해당하는 *SBP*와 일치되어야 한다.

Step 5 : 소스코드 클러스터링	
입력물과 산출물	*sCS, sSBP* → Source Code → *sClu*
지침	ST501 : *sCS*와 *sSBP*를 기반으로 *sCS*에 해당하는 코드를 클러스터링 한다. ST502 : *CSx*는 *SBPx*를 참조하여 Clu_x와 일치되는지 검증한다. 다음과 같은 조건에 의하여 *SBPx*와 맞지 않을 경우 *CSx*는 클러스터링 된 코드집합 Clu_x에 맞게 재작성한다. ST502_1 : 하나의 기능에서 다른 기능을 호출할 경우 하나의 클러스터에 포함되어야 한다. ST502_2: 서로 다른 기능이 같은 기능을 호출할 경우 각 기능은 하나의 클러스터에 포함되어야 한다. ST502_3: 여러 개의 기능이 같은 데이터페이스를 사용할 경우 하나의 클러스터에 포함한다.



(그림 4) 레거시 코드의 기능들을 클러스터링 하는 과정

3.2.2 재사용 서비스 도출 및 명세

이 절에서는 3.2.1의 레거시 시스템 분석 단계에서 얻은 여러 후보 서비스들중 중복성을 배제한 실제 사용할 서비스를 명세한다. 레거시 시스템으로부터 추출한 후보서비스는 디자인 명세 단계에서 추출한 비즈니스 기능으로부터 분석한 후보서비스들이기 때문에 다른 비즈니스 기능에 의하여 중복선언될 수 있다. 따라서 이와 같이 추출한 후보서비스 집합에서 중복성을 배제하고 재사용 가능한 서비스를 도출하여 서비스 명세를 작성한다. 서비스 명세에는 서비스의 특징 및 기능, 입력사항등의 내용을 명세화한다. 후보서비스 명세에는 서비스 이름, 서비스 특징, 인터페이스 이름, 데이터 타입, 입력값, 출력값, 선행조건, 후행조건등이 명시된다. 서비스 명세에 대한 템플릿은 <표 2>와 같다.

〈표 2〉 서비스 명세 템플릿

서비스 명세서	
서비스 이름	후보서비스 이름이다.
특징	서비스가 가지고 있는 주 기능
래퍼클래스 이름	클러스터링된 레거시 코드에 접근하기 위하여 작성한 클래스 이름
메소드 이름	레거시 코드에 실제 접근하기 위한 메소드이름
WSDL	외부 API에 제공될 WSDL
입력값	메소드에 접근하기 위한 입력매개변수
출력값	실행후의 결과값
선행조건	서비스 수행하기 위한 선행조건
후행조건	서비스 수행후 발생하는 결과

3.2.3 서비스로 래핑

명세한 후보 서비스를 실제 서비스화할 수 있도록 래핑하는 단계이다. 래핑은 추출된 CS에 대한 래퍼클래스를 작성하여 외부 API에 의해 호출될 WSDL를 작성한다. 일반적으로 서비스를 요구하는 표준 API는 레거시 코드에 직접 접근할 수 없기 때문에 래퍼클래스와 같이 공용 인터페이스를 구현하여 서비스를 제공하도록 한다. 래퍼클래스는 실제 표준 API가 레거시 코드에 접근할 수 있도록 레거시 코드의 엔트리(entry)는 메소드로 변형하고 입력값은 매개변수(parameter)로 구현한다.

한편 이 단계에서는 실제 API가 필요로 하는 입력 파라미터와 일치하지 않을 경우 표준 API의 유형에 맞게 매개변수를 변경해주어야 한다. 그런데, 서비스를 요청하는 표준 API가 제공되는 인터페이스의 매개변수 유형 또는 파라미터가 맞지 않을 경우 어댑터(adaptor)나 조정자(mediator)를 통해 표준 API에 적합하게 인터페이스를 제공하도록 하여야 한다[9]. 이 경우 어댑터 패턴과 조정자 패턴을 이용하게 되는데 패턴의 사용방법은 <표 3>과 같다.

〈표 3〉 표준 API와 인터페이스간의 매개변수에 따른 패턴의 사용

일치도 / 서비스와 인터페이스 접근	인터페이스 일치	인터페이스 불일치
1:1	없음	어댑터 사용
1:n	중재자 사용	중재자와 어댑터 사용

표준 API와 인터페이스가 일치하는 경우 다른 변환없이 직접 래퍼클래스에 의하여 레거시 코드를 호출할 수 있다. 인터페이스가 일치하지 않는 경우 어댑터를 사용하여 표준 API 호출에 적합한 인터페이스로 변환해준다. 제공되는 서비스가 두 가지 이상의 컴포넌트로 이루어지는 경우 중재자를 사용하여 연결해주는 역할을 수행한다. 마지막으로 두 개 이상의 모듈내에서 인터페이스 까지 불일치 할 경우 어댑터를 사용하여 변환해주어야 한다. 실제 레거시 코드를 래퍼 클래스로 작성할 때에는 위와 같이 표준 API 특성맞는 패턴을 사용하여야 하며 패턴의 사용 방법은 [4][9]에서 제안한 방법을 사용하면 된다.

3.2.4 서비스 등록

실제 서비스 제공을 위하여 래퍼클래스에 해당하는 WSDL을 작성하고 서비스를 제공할 수 있는 UDDI(Universal Description Discovery and Integration)에 등록하는 단계이다[2]. UDDI는 웹서비스를 등록하고 이를 실시간으로 검색할 수 있는 공용 디렉토리로서 SOA 시스템을 구축하기 위해서 여기에 등록된 웹서비스를 검색하여 조립하여 새로운 애플리케이션을 구축하게 된다.

4. 사례 연구

본 논문에서 제안한 M-LSWS 방법론의 유효성과 실용성을 보여주기 위하여 C로 구축된 도서관리시스템(Book Management System, 이하 BMS)에 적용하여 사례 연구를 수행한다. 사례 연구에서는 제안한 방법론을 적용하여 웹서비스로 변환하는 과정을 살펴보며 이를 토대로 평가하여 본다.

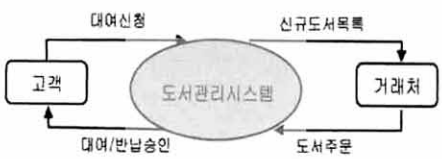
4.1 레거시 시스템 분석

(그림 5)는 BMS에서 회원과 거래처의 핵심기능을 수행하는 관계도를 보여주고 있다. 회원은 도서 대여를 신청하고 거래처는 신규 도서를 제공한다. 이와 같은 기능을 수행하기 위해서 BMS는 회원관리, 도서대여, 도서관리등의 기본적인 기능을 제공해야 한다.

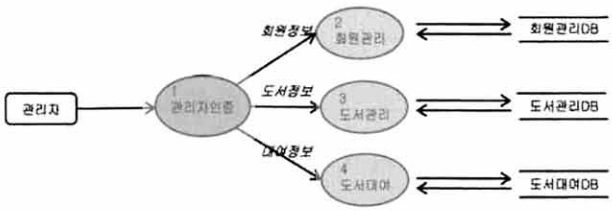
BMS에서 제공되는 디자인 명세서에는 요구사항 명세서, DFD, 클래스 다이어그램(class diagram)등이 있다. (그림 6)은 BMS에 대한 상위흐름도로서 비즈니스 핵심 기능으로는 '회원관리', '도서관리', '도서대여'로 분류할 수 있다.

Step 1 : 디자인 명세로부터 비즈니스 핵심 기능 명시

3.2.1절에서 정의한 Step 1의 지침 ST101에서 하나의 *RS*



(그림 5) BMS 구성도

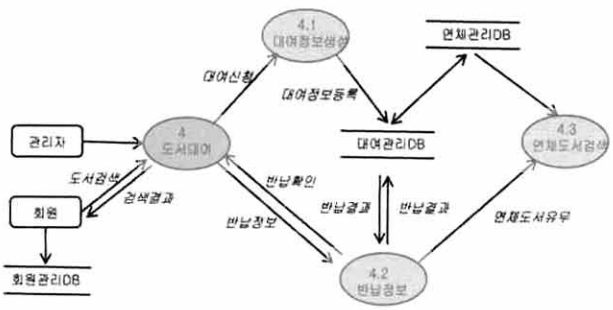


(그림 6) BMS 상위 DFD

또는 *DFD*는 하나의 *BCA*로 매핑될 수 있다고 정의하였다. (그림 6)은 BMS의 상위 *DFD*이므로 각각의 '회원관리', '도서관리', '도서대여'는 다시 상세화된 *DFD*가 된다. 따라서 각각의 *DFD*를 전체 시스템의 *BCA*로 명시할 수 있다.

Step 2 : 비즈니스 핵심 기능으로부터 비즈니스 프로세스 추출

세 가지 유형의 *BCA*중 (그림 7)은 '도서대여'에 해당하는 *DFD*로서 데이터 흐름을 보여준다. 도서대여 *DFD*는 하나의 비즈니스 프로세스로 매핑되지만 내부적으로 '도서연체'를 관리하거나 '도서반납'을 관리하는 추가적인 *BP*를 포함하므로 조건별로 분기하는 *BP*는 새로운 후보 서비스가 될 가능성을 되므로 다음 단계인 명세과정에서 이를 명시한다.



(그림 7) 도서대여 DFD

Step 3 : 비즈니스 프로세스 명세

<표 4>는 *Step 2*에서 추출한 '도서대여'에 관한 *SBP*이다. *SBP*는 *RS*, *DFD*를 참조하여 다음과 같은 명세서를 추출할 수 있게 된다.

(그림 8)는 '도서대여' *SBP*를 기반으로 한 *BPMN*이다. *BPMN*은 *BP*를 도식화하여 업무 흐름의 이해를 도와 *BP* 내에 포함되어 있는 새로운 후보서비스를 추출하고자 사용된다. Step 2에서 언급하였듯이 '도서대여'는 조건에 맞는 도서를 검색하여 대출을 여부를 결정하지만 먼저 관리자에게 가져가 '회원인증'을 받은 후 '도서대여'를 한다. 이 중 연체여부에 따라 대여가 가능할 수도 불가능할 수도 있게 된다. 따라서 '도서대여' *BPMN*으로부터 조건에 맞는 '도서검색', '도서 대출', '회원인증', '연체관리'와 같은 세부 기능으로 상세화할 수 있다. 이 경우 Step 3의 지침 ST402에 따라 각각의 세부 기능은 하나의 후보서비스가 될 수 있게 된다. '도서대여' 외에 '회원관리', '도서관리' *BPMN*에서도 같은 방법으로 후보서비스를 추출하였다.

〈표 4〉 도서관리 시스템중 대출관리에 해당하는 비즈니스 명세서

비즈니스 프로세스 명세서	
비즈니스 프로세스명	도서대여
설명	도서를 대출하는데 필요한 비즈니스 프로세스에 대한 기술
비즈니스 요구사항	회원은 대여하고자 하는 도서를 검색하고 대출을 위하여 아이디와 패스워드를 입력한다. 관리자는 아이디와 패스워드를 확인하고 대출을 수행한다.
입력데이터	회원의 아이디와 패스워드
출력데이터	도서대여 목록리스트
예외사항	회원의 아이디와 패스워드가 일치하지 않으면 대출은 종료된다. 회원이 연체상태라면 대출은 종료된다.
비즈니스 시나리오	1. 회원이 자신이 원하는 책의 도서명, 저장명 등 원하는 정보를 검색한다. 2. 책을 가지고 사서에게로 간다. 3. 사서는 회원과 아이디를 확인한다. 일치하지 않으면 에러메시지를 출력한다. 4. 대출할 책의 바코드를 찍는다. 만약 학생이 연체중이면 연체정보를 확인한다. 5. 사서는 학생에게 책을 대여해준다. 6. 대출 종료
관련 비즈니스 프로세스	도서검색, 연체 관리

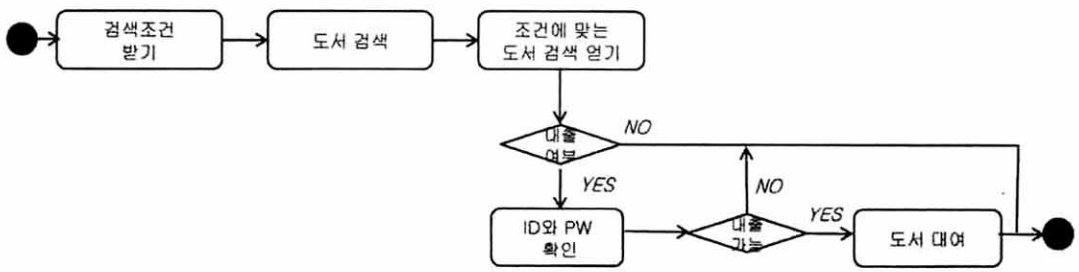
Step 4 : 후보서비스 명시

이 단계에서는 Step 3에서 추출한 후보 서비스들의 중복성을 없애고 실제 웹서비스로 변환하기 위한 후보서비스를 명시한다. Step 1에서 명시한 *BCA*인 '회원관리', '도서대여', '도서관리' *BP*로부터 추출한 후보서비스들은 다른 *BCA*에 있는 기능들과 중복되므로 이를 배제하고 오른쪽 그림과 같이 나타낸다. *BCA*로부터 중복성을 배제한 추출한 후보 서비스는 (그림 9)와 다음과 같다.

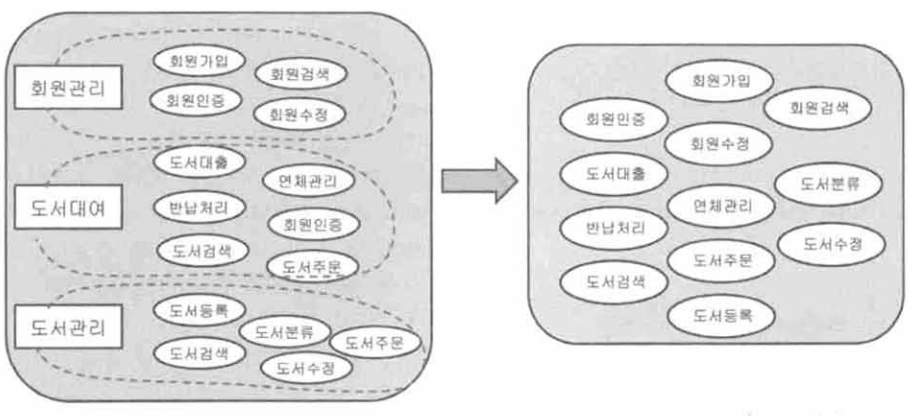
Step 5 : 소스 클러스터링

(그림 10)은 BMS의 함수 호출관계를 도식화한 결과로서 소스를 클러스터링한 결과이다. *Step 4*에서 정의한 지침 ST502_1에서 ST502_3에 따라 서비스로 변환할 수 있도록 클러스터링 하였다.

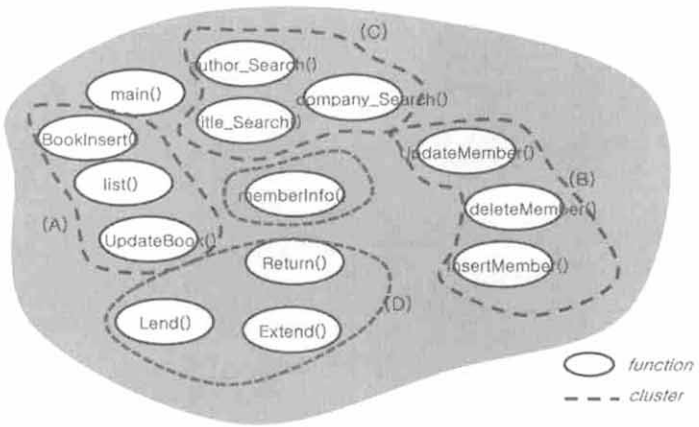
(그림 10)은 BMS의 코드를 서비스로 변환하기 위하여 클러스터링한 결과이다. main() 함수로부터 메뉴를 클릭하면 원하는 메뉴에 따라 호출되는 함수의 기능이 달라지게 되는데 *Step 4*에서 정의한 지침에 따라 서비스로 변환할 수 있도록 클러스터링 하였다. 클러스터 (A)는 '도서입력'과 목록을 보여주는 클러스터로 레거시 코드상에서 '도서관리' DB와 연관되어있으므로 ST501_3에 따라 하나의 모듈로 묶었다. 이외에도 클러스터 (B)는 '회원관리'에 관한 클러스터이며 클러스터 (C)는 '도서검색'에 관한 클러스터인데 각각의 회원관리 DB와 도서관리 DB와 독립적으로 연결되어있으므로 각각의 클러스터로 분리하였다. *memberInfo*는 회원정보를 관리하여 회원관리 DB만을 사용하므로 단일 클러스터로 하였고 (D) *Return, Extend, Lend*는 '도서반납', '도서연장', '도서대여'는 모두 도서대여 DB와 연관되어 있으므로 하나의 클러스터로 묶는다.



(그림 8) BPMN 표기법을 이용한 도서검색을 통한 도서대여 시나리오



(그림 9) 도서관리 시스템에서 추출한 후보 서비스 목록



(그림 10) BMS 레거시 코드의 클러스터링

4.2 재사용 서비스 도출 및 명세

<표 5>는 4.1에서 제시된 후보 서비스 중 도서검색에 관한 서비스 명세이다.

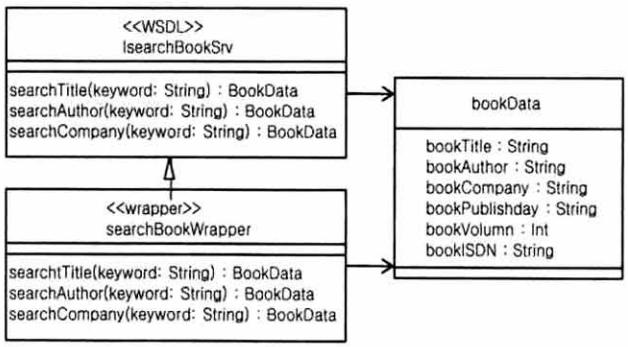
〈표 5〉 BMS에서 searchBookSrv에 해당하는 서비스 명세서

서비스 명세서	
서비스 이름	*searchBookSrv*
특징	도서를 검색하기 위하여 조건을 입력하면 해당하는 검색결과를 보여준다. 검색 방법은 bookTitle, bookAuthor, bookComp와 같이 세 가지 유형이 존재한다.
래퍼클래스 이름	*searchBookWrapper*
인터페이스 이름	*IsearchBookSrv*
메소드 이름	*searchTitle*(String *bookTitle*) *searchAuthor*(String *bookAuthor*) *searchCompany*(String *bookCompany*)
입력값	*bookTitle, bookAuthor, bookCompany*
출력값	도서대여 목록리스트
선행조건	검색할 도서명이 입력되어야 한다.
후행조건	검색결과가 목록으로 보여진다.

4.3 서비스로 래핑

4.2에서 작성한 서비스 명세를 기반으로 실제 레거시 코드를 호출할 수 있는 Java 클래스 파일과 이를 연결할 래핑 클래스를 작성한다. C로 작성된 레거시 코드를 동적으로 연결해주며 Java에서 C를 직접적으로 호출할 수 없기 때문에 Java에서 네이티브 코드를 호출을 제공하는 JNI(Java Native Interface) [10]를 이용하여 레거시코드를 동적 라이브러리로 변환한 후 래핑 클래스를 작성하였다.

(그림 11)은 래퍼클래스와 인터페이스 그리고 결과값을 리턴해주는 클래스의 관계도를 보여준다. 래퍼 클래스 이름은 <표 5>의 명세서 *serchBookSrv*에 기술된 래퍼클래스 이름 *searchBookWrapper*로 명시한다. *searchBookWrapper* 클래스는 (그림 11)과 같이 WSDL로 작성되는 *IsearchBookSrv*에 의하여 호출되며 코드는 (그림 12)와 같다.

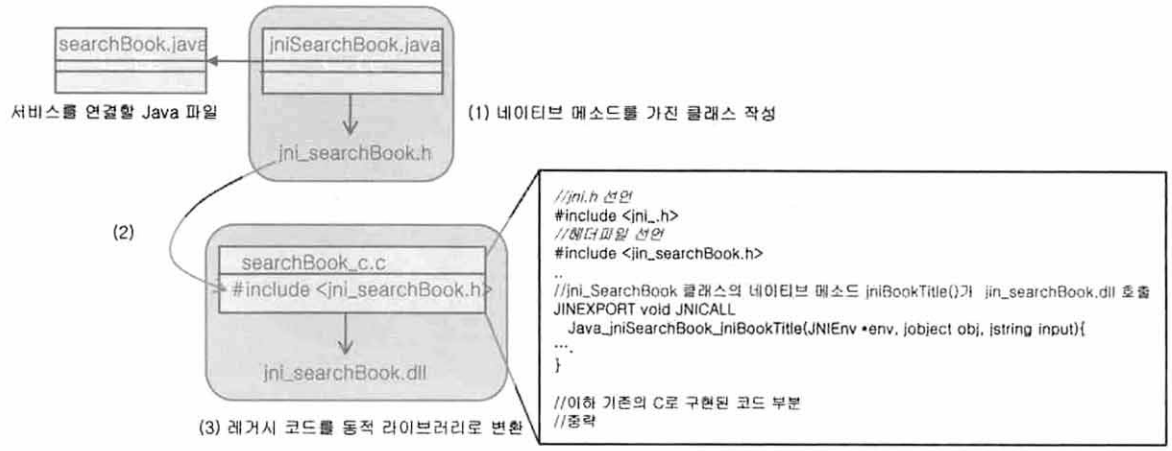


(그림 11) *searchBookWrapper* 클래스와의 관계도

```
class searchBookWrapper{
        public BookData searchTitle(String bookTitle) {
                return SearchTitle().search(bookTitle);
        }
        public BookData searchAuthor(String bookAuthor){
                return SearchAuthor().search(bookAuthor);
  }
        public BookData searchCompany(String bookComp) {
                return SearchCompany().search(bookComp);
        }
}
```

(그림 12) searchBook을 호출할 래퍼클래스 searchBookWapper.java

(그림 13)은 레거시 코드를 Java에서 호출할 수 있는 동적 라이브러리고 변환하는 과정으로 *searchBook* 코드의 일부이다. 서비스를 연결할 *searchBook* 클래스는 실제 레거시 코드를 접근기 위하여 *searchBookWrapper*를 호출한다. *searchBook* 클래스가 레거시 코드를 호출하는 원리는 Java에서 네이티브 코드 호출을 지원하는 JNI(Java Native Interface)를 이용하였다. 실제 동작원리는 C로 작성된 레거시코드를 동적 라이브러리 *jni_searchBook*.dll로 변환한 후 이 라이브러리를 Java 메소드인 *loadLibrary*()가 호출하는 원리로 이루어진다. (1) 서비스와 레거시 코드를 연결할 브리지 역할을 수행하는 *seachBook*을 작성하며 이 코드내에



(그림 13) JNI에 의하여 레거시 코드를 동적 라이브러리로 변환

*jniSearchBook.h*의 네이티브 메소드를 호출하는 메소드를 선언한다. *jniSearchBook.java*는 네이티브 코드를 호출하기 위한 메소드를 선언하고 실제 몸체는 *loadLibrary*() 메소드를 이용하여 (3)의 라이브러리를 호출하도록 한 후 헤더파일로 변경한다. (3)은 레거시 코드에 *jniSearchBook.h*을 포함하고 Java와 레거시 코드를 포함하는 선언부를 선언한 후 동적 라이브러리로 변경한다. 본 논문에서는 레거시 코드를 Win32에서 생성된 *jni_searchBook.dll*로 변환하여 사용하였다.

4.4 서비스 등록

서비스를 등록하기 위한 WSDL 생성단계이다. 실제 서비스에 접근할 수 있는 래퍼클래스 *searchBookWrapper.java*에 해당하는 WSDL을 생성하고 4.3에서 기술한 서비스 명세를 따르게 된다. WSDL 생성은 명세에 따라 직접 작성해도 되지만 지원되는 툴에 따라서 자동생성되기도 한다. 생성된 WSDL[8]을 UDDI에 등록하게 되면 실제 웹서비스의 역할을 수행할 수 있게 된다. 등록된 서비스는 이를 필요로 하는 사용자로부터 표준 API를 통해 호출되어 사용되며 이때 래퍼클래스에 의해 래핑된 레거시 코드의 기능을 사용하게 된다.

5. 평가 및 토의

본 논문에서 제안한 프로세스의 검증을 위하여 다음과 같은 일곱가지의 평가 기준을 기반으로 비교평가하였다. 평가기준은 [11][12]에서 제안한 항목을 기반으로 추출하였으며 기존의 방법론과 비교분석하여 평가기준에 적합한 지원을 하는지 여부에 따라서 *support*(O), *semi-support*(△), *not support*(×)으로 분류하였다.

*프로세스의 체계적 구성*은 제안된 프로세스가 단계, 절차, 세부단계와 같이 일정한 업무단위에 따라 이루어지도록 작성되었는지를 평가한다. *지침의 상세화*는 제안된 지침과 활동이 다음 단계에 일관성있게 전달되도록 상세히 기술되어야 하는지를 평가하며 *산출물의 명세*는 잘 정의된 산출물은 인도자에게 명확한 목적과 비젼을 제시하게 되므로 산출물에는 핵심 항목과 템플릿, 예제 그리고 관련된 가공물과 활동내역을 상세히 기술하도록 한다. *추적성*은 산출물간의 항목들은 추적가능해야 하고 그에 따른 활동이 정의되어야 함으로 평가한다. *적용성*은 잘 정의된 프로세스는 실제 프로젝트에 적용할 때 사용자가 이해하기 쉽고 타당성있게 정의되는지 여부를 평가하며 *간결성*은 프로세스가 불필요한 산출물과 활동들을 만들지 않고 오직 최소한의 노력으로 바람직한 결과를 얻도록 작성되어야 하는지를 나타낸다.

본 논문에서 제안한 M-LSWS와 타 연구와의 비교 평가 결과는 <표 6>과 같다. M-LSWS는 레거시 시스템 분석부터 실제 웹서비스가 변환하기 까지 단계, 절차 및 세부 지침까지 상세히 정의하였기 때문에 *프로세스의 체계적 구성*과 *지침의 상세화 항목*에 '*support*'로 명시하였다. 또한 각 세부 절차에 따라 얻어지는 산출물은 다음 단계의 입력자료로 사용될 수 있도록 명확한 목적과 비전을 제시하였으며 또한 추적자료로 사용될 수 있으므로 *산출물 명세*와 *추적성* 항목에 '*support*'를 명시하였다. *적용성*은 상위 항목의 평가 결과에 근거하여 '*support*'를 명시하였으며 *간결성*은 프로세스내에서 이루어지는 단계의 상세함이 오히려 사용자에 따

〈표 6〉 다른연구와의 비교 평가

기존의 방법론 / 평가 기준	Belushi	Sneed	Wang	M-LSWS
프로세스의 체계적 구성	△	△	△	O
지침의 상세화	×	×	×	O
산출물 명세	×	×	△	O
추적성	△	△	△	O
적용성	O	O	O	O
간결성	O	O	O	△
SOA 적합성	O	O	△	△

O : *support*
△ : *semi-support*
× : *not support*

라 복잡하게 여겨질 수 있어 '*semi-support*'를 명시하였다. 마지막으로 *SOA 적합성*은 본 논문에서 완전한 검증을 거치지 않았기 때문에 '*semi-support*'를 명시하였다.

6. 결론 및 향후 연구

SOA는 기업의 인프라의 복잡성 및 유지비용을 최소화하고 기업의 생산성과 유연성을 극대화할 것으로 기대되어 산업계에서 꾸준한 사용이 증가하고 있다. 기업은 SOA로 변환하기 위하여 기존의 시스템을 버리고 새로운 SOA를 도입하기에는 비용과 시간면에서 많은 위험이 있기 때문에 기존의 레거시 시스템을 최대한 이용하면서 점차적으로 SOA로 변환하기를 원하고 있다. 기존의 레거시 시스템이 가지고 있는 핵심 내용을 변경할 필요없이 그것들이 가지고 있는 기능만 노출시킴으로써 SOA로 마이그레이션하는 방법이 필요하다.

본 논문에서는 조직의 레거시 시스템을 SOA에 적합한 웹서비스로 변환하기 위한 M-LSWS를 제안하였다. M-LSWS는 레거시 시스템이 가지고 있는 디자인 명세 및 코드를 기반으로 비즈니스 프로세스를 분석한 후 후보 서비스를 식별하여 재사용 가능한 웹서비스로 래핑하여 서비스를 등록하는 절차로 이루어져있다. 레거시 시스템 분석 단계에서 서비스 등록까지 네 단계로 이루어진 프로세스를 정의하였으며 각 단계에서는 마이그레이션에 필요한 상세한 활동과 지침을 정의하였고 산출되는 명세 및 가공물에 대한 템플릿을 제안하였다. 각 단계에서 발생하는 산출물은 다음 단계의 입력자료로 사용되어 산출물을 기반으로 데이터 흐름이 이루어지도록 정의하였다. 레거시 코드를 서비스로 변환하기 위하여 클러스터링 기법을 제안하였으며 클러스터화된 코드를 서비스로 구현하기 위한 래핑방법을 정의하였다. 제안한 방법론을 BMS에 적용하여 검증하였고 일곱가지 프로세스 품질평가을 선정하여 타 연구와 비교 평가하였다. 본 논문에서 제안한 M-LSWS는 기업이 SOA로 변환하고자 할 때 기존의 레거시 시스템을 안정적으로 SOA로 변환하도록 유도함으로써 시간과 비용에 대한 위험성을 줄여주게 되며 SOA에 적합한 유지보수 및 확장가능성을 높여줄 것으로 기대된다.

참 고 문 헌

[1] Zhuopeng Zhang, Ruimin Liu, Hongji Yang, "Service Identification and Packaging in Service Oriented Reengineering," pp.620-625, SEKE, 2005.

[2] Thomas Erl, Service-Oriented Architecture, Prentice Hall PTR, 2004

[3] Kulkarni, N., Dwivedi, V., "The Role of Service Granularity in a Successful SOA Realization A Case Study" SERVICES '08. IEEE 6-11 pp.423 - 430, July, 2008.

[4] Al Belushi, W., Baghdadi, Y., "An Approach to Wrap Legacy Applications into Web Services," Service Systems and Service Management, 2007 International Conference, June, 2007.

[5] Sneed, H., "Integrating Legacy Software into a Service Oriented Architecture," In Proceedings of the 10th European Conference on Software Maintenance (CSMR 2006), March, 22-24, IEEE Computer Society Press, 2006.

[6] Xiaofeng Wang, Hu., S.X.K., Haq, E., Garton, H., "Integrating Legacy Systems within The Service-oriented Architecture," Power Engineering Society General Meeting 2007, IEEE 24-28, June, 2007.

[7] Lewis, C., Morris, E., Smith, D., Simanta, S., SMART: Analyzing the Reuse Potential of Legacy Component in a Service-Oriented Architecture Environment, CMU/SEI-2008-TN-008, Software Engineering Institute, May, 2008.

[8] W3C, Web Services Description Language(WSDL) Version 2.0 Part 1: Core Language, "http://www.w3c.org/TR/2007/REC-wsdl20-20070626/, 2007.

[9] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.

[10] Java Native Interface, http://java.sun.com

[11] IEEE Computer Society and ACM, Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society, 2004.

[12] Pressman, R., Software Engineering: A Practitioner's Approach 6th edition, McGraw-Hill, 2005.

박 옥 자

e-mail : ojpark@otlab.ssu.ac.kr

1999년 전북대학교 전산통계학과(이학석사)

2008년 전북대학교 전산통계학과(이학박사)

2004년~2006년 백석대학 IT 프로그래밍 교수

2008년~현 재 숭실대학교부속 정보미디어기술연구소 전임연구원

관심분야 : 서비스 지향 아키텍처(SOA), 관점 지향 프로그래밍(AOP), 컴포넌트 기반 개발(CBD), S/W 유지보수 및 추적기법

최 시 원

e-mail : swchoi@otlab.ssu.ac.kr
2000년 삼육대학교 컴퓨터학과(공학사)
2002년 숭실대학교 컴퓨터학과(공학석사)
2008년 숭실대학교 컴퓨터학과(공학박사)
2008년~현 재 숭실대학교부속 정보미디어기술연구소 연구교수
관심분야 : 품질모델(Quality Model), 서비스 지향 아키텍처(SOA), 서비스 품질 관리

김 수 동

e-mail : sdkim@ssu.ac.kr
1984년 Northeast Missouri State University 전산학(학사)
1988년~1991년 The University of Iowa 전산학(석사/박사)
1991년~1993년 한국통신 연구개발단 선임연구원
1994년~1995년 현대전자 소프트웨어연구소 책임연구원
1995년~현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 서비스 지향 아키텍처(SOA), 객체지향 S/W공학, 컴포넌트 기반 개발 (CBD), 소프트웨어 아키텍처