DOI: 10.3745/KIPSTD.2010.17D.1.25

객체지향 시스템에서 SOA서비스로의 전이 기법

김 지 원[†] · 라 현 정[††] · 김 수 동[†††]

요 약

서비스 지향 아키텍처 (Service-Oriented Architecture, SOA)는 독립적으로 실행가능하며, 외부 인터페이스를 통해서만 접근 가능한 서비스를 재사용하는 개발 패러다임이다. 서비스는 기존의 객체 또는 컴포넌트와 다른 특성을 보유하고 있고, 다수의 서비스 소비자들이 재사용할 수 있는 공통적인 기능을 제공해야 한다. 그러므로, 이런 서비스를 개발하기 위한 기법 연구가 필요하다. 대부분의 SOA 방법론은 서비스 요구사항에서부터 서비스를 새롭게 개발하는 기법인 Top-down 접근법을 제안하고 있어 서비스 개발에 비교적 많은 비용과 노력이 요구된다. 그리고, 많은 기업들은 객체 지향 시스템을 보유하고 있기 때문에, 기존의 객체지향 시스템에서 SOA 서비스를 효과적으로 도출하는 기법에 대한 수요가 크다. 객체 지향 시스템은 단일/특정 요구사항을 고려하여 개발되는 반면에, SOA 서비스는 다수의 소비자들의 공통성을 고려하여 개발되어야 하기 때문에, 간단한 매핑 과정을 통하여 객체 지향 시스템을 SOA 서비스로 전이하는 것은 어렵다. 따라서 본 논문에서는 객체지향 시스템의 다양한 산출물과 SOA 서비스의 주요 산출물간의 매핑 관계를 정의하고, 재사용성을 고려하여 객체지향 시스템을 구성하는 다양한 모델들이 서비스의 주요 산출물로 전이되는 체계적인 기법을 제안한다. 각 기법은 입/출력 산출물 간의 관계와 순서화된 상세 절차로 구성되어 있으므로, 보다 쉽게 객체지향 시스템을 서비스로 전이할 수 있도록 도와준다.

키워드 : 서비스 지향 아키텍처, SOA, 객체지향 시스템, 서비스 전이, 개발 방법론

A Method for Migrating Object-Oriented Systems into SOA Services

Ji Won Kim[†] · Hyun Jung La[††] · Soo Dong Kim[†††]

ABSTRACT

Service-Oriented Architecture (SOA) is a development paradigm for reusing services as an independent reuse unit. A service delivers a cohesive functionality through its external interface. Since services have unique characteristics which are not typically presented in conventional development approaches, there is a demand for effective approaches to developing services. Most of the current SOA methodologies presenta process where services are designed and developed from the requirements rather than reusing existing assets, which demands high cost and effort. Hence, a desirable approach is to be able to develop services by migrating from their existing legacy systems such as object-oriented system. A difficulty in this migration is that objects in object-orientedsystems reveal characteristics which differ considerably from those of services. That is, objects are designed without considering commonalities among several consumers. In this paper, we first define mapping relationships between key artifacts in object-oriented system and those in SOA services. By these relationships and considering commonalities among several applications in a domain, we propose three systematic methods to migrate from object-oriented system to SOA services. Each method consists of a list of input and output artifacts and detailed guidelines which are performed in order. Through these methods, service developers can easily develop services with less effort.

Keywords : Service Oriented Architecture, Object Oriented System, Service Migration, Development Method

1. 서 론

서비스 지향 아키텍처 (Service-Oriented Architecture, SOA)는 독립적으로 실행가능하며, 외부 인터페이스를 통해서만 접근 가능한 서비스를 재사용 단위로 하는 재사용 개발 패러다임 중 하나이다 [1]. SOA에서 서비스 제공자는 잠재적인 여러 서비스 사용자를 위해 범용적이고 재사용 가능한 서비스를 개발하고, 이를 서비스 저장소에 배포한다. 서비스 소비자는 서비스 저장소에서 적절한 서비스를 검색하고 발견하며, 서비스 어플리케이션의 일부로 서비스를 구독한다. SOA 서비스는 느슨한 결합성(Loosely Coupling), 비즈니스 요구사항 일치성(Business Alginment), 블랙 박스

※ 이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임.(No. 2009-0076392).
† 준 회 원 : 숭실대학교 컴퓨터학과 석사과정
†† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정(교신저자)
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2009년 11월 16일
수 정 일 : 1차 2010년 1월 14일, 2차 2010년 1월 30일
심사완료 : 2010년 2월 8일

형태(Black-box form)과 같은 기존의 개발 접근 방식인 객체지향 개발 방식과 컴포넌트 기반 개발 방식과는 구별되는 특징을 가지므로, 기존의 개발 방식을 이용해서 서비스를 개발하기 어렵다. 그러므로, 많은 문헌들은 고유한 특징을 가지고 있는 서비스를 효과적으로 개발하기 위해 다양한 개발 방법론을 제안하고 있다 [2, 3].

대부분의 개발 방법론은 Top-Down 접근 방법을 사용하여, 서비스 요구사항 정의에서부터 서비스 분석, 서비스 설계, 서비스 구현, 서비스 배포 등의 과정을 설명하고 있다. 이 방법을 이용하면, 고품질의 서비스를 개발할 수 있게 되지만, 다소 많은 노력이 소요된다. 이런 노력을 줄이는 한 방법으로, 현재 산업계에서는 각 기업이 보유하고 있는 객체지향 시스템과 같은 전통적인 시스템을 기반으로 서비스를 개발하려는 시도가 행해지고 있다. 기존 시스템을 이용하여 서비스를 개발하게 될 경우, 많은 기능이 이미 구현이 되어 있으므로 보다 적은 노력으로 기존의 기능을 서비스화 시킬 수 있게 된다.

그러나, 객체지향 시스템의 객체는 SOA의 서비스와 다른 특징을 가지고 있다. 서비스는 여러 사용자에 의해 공유될 수 있는 재사용 가능한 기능성 단위이지만, 객체지향 시스템의 객체는 특정 어플리케이션에 국한되어 사용되는 기능성 단위이다. 즉, 객체는 다양한 사용자의 재사용성을 고려하지 않고 설계된다. 이런 차이로 인해, 객체가 서비스로 비교적 수월하게 전이되지 않는다. 또한, 서비스 개발 방법론에 대한 연구는 진행되고 있지만, 객체지향 시스템을 이용하여 체계적으로 서비스를 개발하는 방법은 아직 널리 연구되고 있지 않고 있다.

그러므로, 본 논문에서는 객체지향 시스템의 다양한 산출물과 SOA 서비스의 주요 산출물간의 매핑 관계를 정의하고, 재사용성을 고려하여 객체지향 시스템을 SOA 서비스로 전이하는 체계적인 방법을 제안한다. 2장에서 레거시 시스템을 이용하여 서비스를 개발하는 방법에 대한 연구를 간략히 요약하며, 3장에서 전이 대상이 되는 객체지향 시스템과 SOA의 메타 모델을 알아본다. 4장에서는 전이하여 개발된 서비스가 만족해야 하는 품질 조건을 정의한다. 5장에서는 객체지향 시스템을 구성하는 다양한 모델들이 서비스의 주요 산출물로 전이되는 체계적인 기법을 제안하고, 6장에서는 제안된 기법을 적용한 사례연구의 결과를 보여준다. 각 기법은 입/출력 산출물 간의 관계와 순서화된 상세 절차로 구성되어 있으므로, 보다 효과적으로 객체지향 시스템을 서비스로 전이할 수 있도록 도와준다.

2. 관련 연구

Kim[4] 연구에서는 유즈케이스에 대해 다섯 가지의 리팩토링방법을 적용하여 후보 서비스를 추출 하는 방법을 제시하고 있다. 첫 번째 기법은 복잡한 기능을 가진 유즈케이스에 UC_1대해 기능을 상세화 하는 분해과정을 통해 새로운 UC_1'를 생성한다. 두 번째는 동일한 기능성을 보유하고 있는 두 개의 유즈케이스 UC_1, UC_2가 있다면 UC_2를 UC_1로 치환한다. 세 번째는 두 개의 유즈케이스 UC_1, UC_2 가 있고, 둘의 근본적인 기능이 동일 할 경우 그 기능을 가지는 새로운 유즈케이스를 UC_3를 생성하여UC_1과 UC_2 이UC_3를 상속 하도록 한다. 네 번째는 두 유즈케이스 UC_1과UC_2에 대해, UC_2이 UC_1에 Association의 관계라면 UC_1에 UC_2를 합병함으로 둘 간의 관계를 제거하고 합병된 유즈케이스는 둘 모두를 대치할 수 있다. 마지막으로 UC_1는 여타의 유즈케이스 또는 액터와 아무런 관계가 없을 경우UC_1를 제거한다. 제시한 다섯 가지의 리팩토링방법은 적절한 크기의 후보 서비스를 산정하고, 서비스들 간의 비슷한 추상화 정도를 유지 할 수 있도록 돕는다. 하지만, 서비스의 특성중에 하나인 재사용성에 대한 고려가 추가 된다면, 특정 조직에만 유용하지 않고 좀더 많은 서비스 사용자가 사용할 수 있는 서비스는 설계 할 수 있다.

Kulkarrani[5] 연구에서는 레거시 시스템 전이를 위하여 아키텍쳐 중심의 InSOAP 기법을 제안한다. InSOAP은 레거시 시스템에서 비즈니스 프로세스를 적절한 서비스의 개념적 크기에 맞게 분류하도록 제안하고, 제안한 비즈니스 서비스의 실체화를 위해 서비스, 데이터 흐름, 컴포넌트와 같은 서비스 및 기능의 흐름등을 결정하는 지침을 정의하고 있다.

Belushi[6]와 Sneed[7]는 레거시 애플리케이션으로부터 추출한 여러 비즈니스 로직을 웹서비스로 변환하기 위해 필요한 래핑방법에 초점을 두고 제안하고 있다. 래핑 방법을 세 가지 유형으로 분류하고 세 방법의 장점만을 모은 하이브리드 방법론을 제안하였다. 하지만, 레거시 시스템으로부터 서비스를 추출하는 기준제시가 다소 불분명하고 실제 웹서비스 구축 부분만 기술하였기 때문에 초기 적용 단계에서부터 서비스 추출에 대한 어려움을 안고 있다.

Wang[8]의 연구에서는 레거시 시스템은 각 모듈간의 비즈니스 로직이 강한 결합력을 가지고 있으므로 SOA적합한 서비스가 되도록 비즈니스 로직과 데이터 노출에 대한 방법을 제안하고 있다. Lewis[9]는 SMART(Service Migration and Reuse Technique)를 제안하고 있다. 이는 레거시 시스템을 분석하여 후보 서비스를 추출하고 대상 SOA환경을 기술하여 현 레거시 환경과 SOA환경간의 차이를 조절하여 전이 전략을 세우는 절차로 이루어져 있다.

이처럼 기존의 방법론은 레거시 시스템에서 후보 서비스의 래핑방법에 대해 초점을 두고 있다. 또는 마이그레이션을 프로세서로 나누어 각 단계에서 이루어지는 업무나 특징에 대해서 정의한 방법론도 있지만 상세화된 활동이나 지침이 부족하여 실제 적용하는데 어려움이 있다.

기존 시스템의 핵심 내용을 변경할 필요 없이 그것들이 가지고 있는 기능만 노출시킴으로써 하나의 웹서비스로 변환하고자 했던 연구는[10] 마이그레이션을 위해 여러 단계로 활동을 나누고 각 단계별 절차와 가이드 라인을 제시하고 있다. 래핑을 목적으로 하는 이 방법은 기존 시스템에서 제공하는 기능을 혼합하거나 또는 보다 작은 기능 단위로 분

해하여 새로운 기능을 제공하기 어렵다. 그리고 기존의 시스템이 보유 하고 있는 산출물을 충분히 활용하지 못 하기 때문에, 분석에 있어서 난해함을 가져 오거나 보다 많은 시간을 요구 할 수 있게 된다.

유즈케이스 기반으로 서비스 추출을 시도한 연구에서는 [11, 12] 기존 시스템의 산출물을 충분히 활용하여 서비스를 위한 산출물로 변환하고자 하였다. 유즈케이스를 분석하여 서비스를 식별하고 다시 정제 함으로써 적절한 입도로 서비스를 구성 하도록 하고, 산출물간의 관계를 통해 보다 실현 가능한 방법을 제시하고 있다. 하지만, 서비스를 생성하기에는 기존 시스템의 컴포넌트와의 관계에 대한 정보가 부족하여 서비스를 추출 후 서비스를 구현하기에는 부족함이 있다.

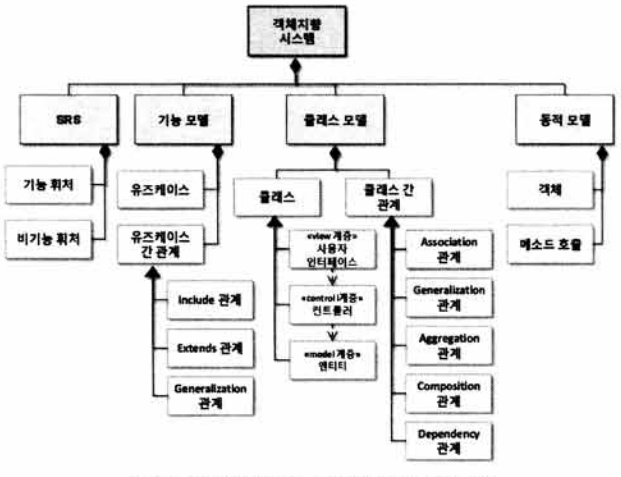
레거시 시스템을 웹 서비스에 통합하기 위한 연구에서는 [13] 기존 시스템을 웹서비스로 변환하기 위해 필요한 구성 요소와고려해야 하는 점을 제안하고 있다. 마이그레이션을 위해 고려할 점들을 나열 하였지만, 실질적인 산출물에 대한 변환 과정이 제시되지 않았기 때문에 실질적인 변환에 적용하기는 어려움이 있다. 따라서 본 논문에서는 이와 같은 문제점을 보완할 수 있는 보다 상세한 기법과 절차를 제안한다.

3. 메타 모델

3.1 객체지향 시스템 메타 모델

본 장에서는 객체지향 시스템에 대한 여러 문헌들을 조사 분석하여, SOA 서비스로 전이되는 대상인 객체지향 시스템의 구성 요소를 메타 모델로 정의한다[14, 15]. (그림 1)은 객체지향 시스템의 구성 요소를 보여준 것으로, 본 논문에서는 SOA 서비스로 전이되는 대상을 소스 코드에 국한 시킨 것이 아니라, 객체지향 시스템을 개발하는데 생성되는 다양한 중간 산출물도 활용한다.

객체지향 시스템은 크게 네 가지 종류인 소프트웨어요구사항 명세서 (Software Requirement Specification, SRS), 기능 모델, 클래스 모델, 동적 모델로 구성된다. SRS는 사용자의 요구사항을 기술한 것으로, 기능 요구사항을 기술한 기능 휘처 (Feature)와 비 기능 요구사항 및 기타 소프트웨어 제약사항을기술한 비 기능 휘처로 이루어진다.



(그림 1) 객체지향 시스템의 메타 모델

기능 모델은 SRS를 기능 관점에서 분석하여 작성한 산출물로, 객체지향 시스템이 제공하는 기능들을 보여주고, 대개 유즈케이스 모델과 유즈케이스 명세서로 기술된다. 유즈케이스 모델과 유즈케이스 명세서는 기능 단위를 나타내는 다수의 유즈케이스와 Include, Extends, Generalization 의 유즈케이스 간의 관계로 구성된다.

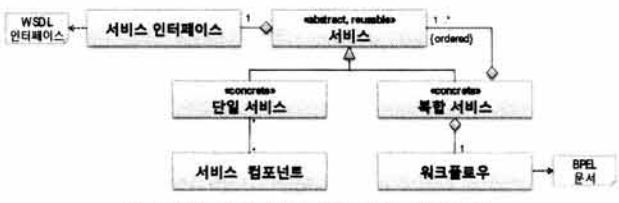
클래스 모델은 객체지향 시스템의 정적인 구조를 보여주는 산출물로, 대개 클래스 다이어그램으로 명세 된다. 클래스 모델에는 객체지향 시스템을 구성하고 있는 다수의 클래스와 각 클래스 간의 관계로 구성된다. 객체지향 시스템을 구성하는 클래스는 가장 보편적으로 사용되는 모델-뷰-컨트롤러 (Model-View-Controller, MVC) 아키텍처에 따라 크게 뷰 계층에 속하는 사용자 인터페이스 클래스, 컨트롤러 계층에 속하는 컨트롤러 클래스, 모델 계층에 속하는 엔티티 클래스로 분류된다[16]. 그리고, 여러 클래스들은 UML에서 정의한 Association, Generalization, Aggregation, Composition, Dependency 관계로 연결되어 있다.

동적 모델은 객체지향 시스템이 실시간에 작동하는 방법을 나타내어 동적인 구조를 보여주는 산출물로, 대개 시퀀스 다이어그램으로 명세 된다. 동적 모델은 객체지향 시스템이 제공하는 기능을 수행하는데 참여하는 다수의 객체와 각 객체 간의 메소드 호출 관계로 표현된다.

위와 같이 정의한 객체지향 시스템 메타 모델의 각 구성 요소는 SOA 서비스로 직/간접적으로 전이되거나 SOA 서비스로 전이되는 데 필요한 정보를 제공하는데 필요하게 된다.

3.2 SOA 메타 모델

본 장에서는 SOA에 대한 여러 문헌들을 조사 분석하여, SOA 서비스로 전이 목표 대상인 SOA 서비스의 구성 요소를 메타 모델로 정의한다[1, 17]. (그림 2)는 SOA 서비스의 구성 요소를 보여준다.



(그림 2) SOA 서비스에 대한 메타모델

서비스는 소비자가 원하는 요구사항을 만족시킬 수 있는 하나 이상의 기능을 포함하는 단위이다. 소비자는 자신의 요구사항을 만족시키는 서비스를 호출하고, 제공자는 재사용성이 높아 이익을 창출할 수 있는 서비스를 제공한다. 서비스는 응집력이 높은 기능을 제공하는 단일 서비스와 이미 생성된 두 개 이상의 단일 서비스로 구성된 복합 서비스로

나뉘어진다. 서비스가 제공하는 기능은 서비스의 구현 내용을 숨긴 상태에서 오직 서비스 인터페이스를 통해 사용자에게 제공된다.

서비스 인터페이스는 소비자가 서비스와 상호 작용할 수 있는 수단을 제공하며, 프로토콜, 오퍼레이션, 입력 데이터 등 서비스를 호출하는데 필요한 정보를 제공한다. 일반적으로 인터페이스는 WSDL을 이용하여 명세 되고 서비스 저장소에 등록되며, UDDI와 같은 서비스 발견 메커니즘을 통해서 발견된다.

단일 서비스는 더 이상 분해되지 않는 기능 단위로, 일반적으로 하나의 스텝에 의해 수행되는 단순한 기능을 가진 태스크를 포함한다. 단일 서비스는 필요한 기능을 모두 포함하고 있으므로, 다른 서비스와 상호 작용하거나 다른 서비스에 의존하지 않고 독자적으로 기능을 수행한다. 서비스 컴포넌트는 단일 서비스의 구현체이다. 서비스 컴포넌트는 일반적으로 객체지향 프로그래밍의 객체 또는 컴포넌트 기반 개발의 컴포넌트로 구현된다.

복합 서비스는 하나 이상의 서비스로 구성된 기능 단위이다. 복합 서비스에 포함되는 서비스는 단일 또는 다른 복합 서비스가 될 수 있으며, 이는 재귀적 조립(Recursive Composition) 패턴으로 정의될 수 있다. 워크플로우는 복합 서비스에 포함되는 여러 서비스간의 제어 흐름을 나타내며, 중개자(Mediator) 패턴을 이용하여 설계 될 수 있다. 일반적으로 워크플로우는 BPEL 문서로 기술된다.

4. SOA 서비스의 품질 기준

본 장에서는 SOA 관련 다양한 문헌[1, 17] 들을 분석하여 전이된 SOA 서비스가 만족해야 하는 기준을 정의하며, 5장에서 제안되는 기법은 이 기준들을 만족하는 서비스가 도출될 수 있도록 설계되어야 한다.

비즈니스 공통성: SOA의 기본적인 사상은 다양한 어플리케이션들간에 공통적인 기능을 가지는 서비스를 개발하고 배포하는 것이다. 비즈니스 도메인에서 높은 공통성을 가지는 서비스는 더 높은 재사용성을 지니게 된다. 또한 서비스가 높은 재사용성을 가질 수록 높은 투자수익률(ROI)를 창출할 수 있다.

비즈니스 요구사항 일치성 (Business Alignment): 서비스는 비즈니스 활동이나 실제 업무의 특징을 잘 반영하도록설계되어야 한다. 서비스는 서비스 내부에 대한 상세 내용을 외부에 노출시키지 않으면서, 실제 비즈니스에서 사용되는 용어를 사용하여 정의되어야 한다. 즉, 서비스는 실제 사용될 비즈니스 도메인의 특징을 잘 나타내도록설계되어야 한다.

기능적 독립성: SOA에서 서비스는 기능적으로 잘 응집되어 있고, 잘 모듈화된 기능 단위이다. 높은 모듈성을 가지는 서비스는 조합성(Composability)을 높일 수 있고, 결국 재사용성도 증가시킬 수 있다.

잘 정의된 서비스 인터페이스: 서비스는 상세 내용을 외부에 노출시키지 않고, 서비스 소비자가 서비스의 기능을 명확하게 이해할 수 있도록 잘 정의된 인터페이스를 제공한다. 서비스의 인터페이스가 잘 정의되어 있으면, 서비스 발견성(Discoverability)이 향상되어 서비스 소비자에 의해 잘 발견될 수 있다.

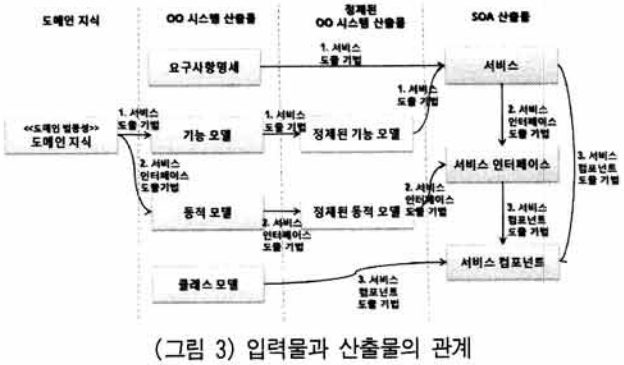
서비스 조립성 지원: 비즈니스 프로세스 상에서 사용자가 필요로 하는 기능은 대개 여러 서비스를 조립하여 수행된다. 즉, 하나의 비즈니스 프로세스에는 기능을 완전하게 수행할 수 있도록 하나 이상의 서비스가 오케스트레이션(Orchestration) 형태로 서로 조립 되어야 한다.

5. 전이 기법

본 장에서는 객체지향 시스템의 메타 모델 구성요소와 SOA 서비스 메타모델의 구성 요소를 매핑시킴으로써, 객체지향 시스템에서 SOA 서비스로 전이하는 체계적인 방법을 정의한다.

객체와 서비스 모두 응집된 기능성을 가진 기능 단위이지만, 객체지향 시스템은 대개 특정 사용자를 위해 개발되고, SOA 서비스는 여러 사용자를 고려하여 공통적인 기능을 포함하도록 개발된다는 차이점을 가지고 있다. 즉, 객체지향 시스템의 주요 산출물은 공통성을 반영하고 있지 않다. 그러므로, 객체지향 시스템에서 SOA 서비스로는 직접적으로 전이되는 것이 어렵기 때문에, 본 장에서는 객체지향 시스템과 SOA 서비스 간의 주요한 차이점을 해결하기 위한 정제된 객체지향 시스템 산출물을 도입한다. 정제된 객체지향 시스템 산출물은 기존의 객체지향 산출물과 도메인 지식을 기반으로 공통성 분석을 실시하여, 공통성 정보가 포함된 것이다.

(그림 3)는 객체지향 시스템 산출물로부터 SOA 서비스 산출물로 전이하기 위한 세 단계 및 두 산출물 간의 관계를 보여준다. 첫번째 단계인 서비스 도출 기법에서는 도메인 지식과 기능 모델을 이용하여 정제된 기능 모델을 만들고, 요구사항 명세, 정제된 기능 모델을 이용하여 재사용 가능한 서비스를 추출한다. 유즈케이스는 사용자가 인지할 수 있는 기능성을 가지는 반면에, 서비스는 독립적으로 실행되며 잘 모듈화된 기능성을 가지고 있다. 그러므로, 하나의 유즈케이스가 하나의 서비스로 전이될 수 있고, 서로 밀접한

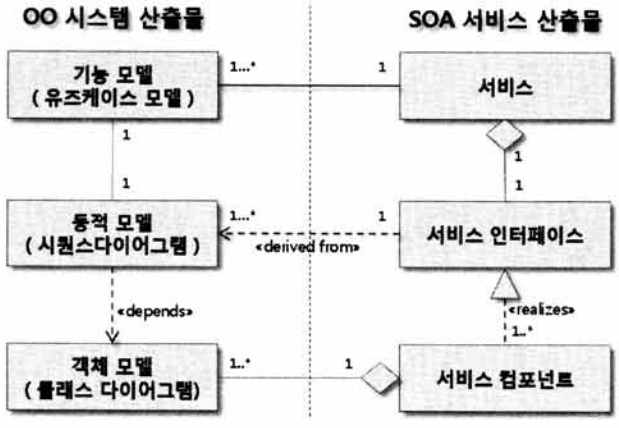


(그림 3) 입력물과 산출물의 관계

연관이 있는 여러 유즈케이스가 하나의 서비스로 전이될 수 있다. 즉, (그림 4)와 같이 기능 모델의 유즈케이스와 서비스 간에는 n : 1 관계가 성립된다.

두 번째 단계인 서비스 인터페이스 도출 기법에서는 도메인 지식과 동적 모델을 이용하여 정제된 동적 모델을 만들고, 이전 단계에서 추출된 서비스 목록과 정제된 동적 모델을 이용하여 서비스 인터페이스를 정의한다. 일반적으로 UML 모델링을 할 때, 유즈케이스 한 개별로 시퀀스 다이어그램을 그린다. 그리고, 하나의 서비스에는 하나의 서비스 인터페이스가 존재한다. 그러므로, 유즈케이스와 서비스 간의 관계가 n : 1 이기 때문에, (그림 4)와 같이 서비스 인터페이스와 동적 모델의 관계도 n : 1 이 된다.

세 번째 단계인 서비스 컴포넌트 도출 기법에서는이전 단계에서 정의한 서비스 목록, 서비스 인터페이스, 클래스 모델을 이용하여 서비스 컴포넌트를 정의한다. 일반적으로 서비스 컴포넌트는 응집된 기능성을 제공하기 위해 하나 이상의 클래스들이 존재하게 된다. 그러므로, 클래스 모델의 클래스와 서비스 컴포넌트 간의 관계는 (그림 4)와 같이 n : 1 이 된다.

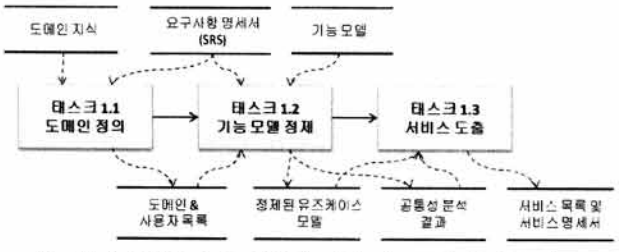


(그림 4) 객체지향 산출물과 서비스의 구조적 모형

5.1 서비스 도출 기법

개요: 서비스 도출 기법에서는 요구사항명세(SRS), 기능 모델, 도메인 지식을 이용하여 기존 객체지향 시스템에서 재사용 가능한 서비스 단위를 추출한다. (그림 5)는 서비스를 도출하기 위한 세부 태스크와 산출물 간의 관계를 보여준다.

객체지향 시스템의 유즈케이스는 도메인 전반적으로 사용



(그림 5) 서비스 도출 기법에 대한 태스크와 산출물 관계

될 수 있는 기능성 보다 시스템에 종속적인 기능적 요구사항을 해소하기 위한 기능성을 포함하고 있다. 따라서, 도메인 지식의 유즈케이스를 추가 함으로써 재사용성 있는 서비스를 도출해야 한다. 재사용성이 고려된 유즈케이스를 생성하기 위해 객체지향 시스템의 기능 모델을 정제하고, 이는 서비스 추출을 위한 기반으로 활용 된다.

(그림 6)은 서비스 도출 기법의 입/출력물과 주요 중간 산출물인 객체지향 시스템의 유즈케이스, 정제된 유즈케이스, 서비스 간의 관계를 보여준다. 공통성이 있는 유즈케이스는 하나 이상의 정제된 유즈케이스로 전이될 수 있고, 공통성 없는 유즈케이스는 정제된 유즈케이스에서 제외된다. 그리고, 서로 연관 있는 여러 개의 유즈케이스는 하나의 정제된 유즈케이스로 합쳐질 수 있고, 기능성이 큰 유즈케이스는 여러 개의 유즈케이스로 분리될 수 있으므로, 유즈케이스와 정제된 유즈케이스는 n : m 관계가 성립된다.

정제된 유즈케이스는 이미 공통성 분석과 유즈케이스 간의 연관 관계 분석을 거쳐서 생성된 것이므로, 서비스로 전이될 주요 후보자가 된다. 정제된 유즈케이스를 대상으로 재사용 및 여러 기준을 이용하여 서비스를 선정하게 되므로, 정제된 유즈케이스와 서비스 간에는 1 : 0…1 관계가 성립된다.

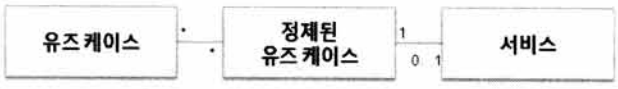
입/출력물: 객체지향 시스템 산출물 중 서비스와 동일하게 기능성을 표현하고 있는 SRS, 기능 모델인 유즈케이스 모델, 재사용성을 고려하기 위한 도메인 지식을 입력물로 사용한다. 그리고 본 장의 기법을 이용하여 재사용 가능한 기능성을 포함하는 서비스 목록과 각 서비스에 대한 명세서가 생성된다.

태스크 1. 도메인 정의: 객체지향 시스템은 단일 조직을 위해 개발된 것이기 때문에, 이 시스템에서 도출된 서비스는 재사용성을 보장할 수 없다. 태스크 1에서는 이런 한계점을 극복하여 재사용성이 높은 서비스를 도출하기 위해 서비스의 도메인 영역을 정의하고, 잠재적인 사용자들을 선정한다.

첫째, SRS 로부터 객체지향 시스템의 비즈니스 목적과 요구 기능의 특징을 분석하며, '도메인 지식'을 이용하여 객체지향 시스템의 해당 도메인을 확인한다. 본 논문에서 도메인은 객체지향 시스템이 속하는 비즈니스 영역으로 정의한다.

둘째, 도메인에 포함되는 잠재 어플리케이션을 선정한다. 객체지향 시스템에서 제공하는 기능성이 재사용될 가능성이 많은지를 측정하기 위해서는 하나 이상의 어플리케이션을 분석해야 한다. 그러므로, 잠재 어플리케이션은 도메인 지식을 이용하여 해당 도메인에서 접근이 용이하고 사용 빈도가 높은 어플리케이션으로 선정한다.

태스크 2. 유즈케이스 정제: 태스크 2에서는 단일 시스



(그림 6) 유즈케이스와 서비스 관계

템을 위해 생성된 유즈케이스 모델의 각 유즈케이스에 대하여 공통성 분석을 거쳐 재사용 가능한 기능만 포함한 유즈케이스 목록을 도출한다. 그리고, 유즈케이스의 높은 모듈성을 위해 연관관계를 고려하여 유즈케이스 모델을 정제한다.

첫째, 객체지향 시스템의 유즈케이스 모델에 포함된 모든 유즈케이스들을 대상으로 공통성 분석을 한다. 객체지향 시스템 외에 여러 어플리케이션을 고려해야 하므로 태스크 1에서 식별된 도메인의 주요 어플리케이션을 대상으로 하며, 공통성 분석표인 <표 1>을 이용한다.

'기능목록' 열에는 목표 객체지향 시스템의 모든 유즈케이스들을 나열하고, '어플리케이션 목록'에는 공통성 분석 조사 대상들을 나열한다. 특정 어플리케이션이 해당 유즈케이스를 제공한다면, √ 을 표시한다.

'재사용성' 열에는 얼마나 많은 어플리케이션들이 해당 유즈케이스를 제공하는지를 나타내는 것으로 다음의 수식을 이용한다.

$$\text{재사용성 정도} = \frac{\text{유즈케이스를 제공하는 어플리케이션수}}{\text{총 어플리케이션 수}}$$

결과값는 0과 1사이의 값을 가진다. 재사용성 정도 수치가 1에 가까울 수록 많은 어플리케이션들이 해당 유즈케이스를 제공하므로 재사용성이 높다는 것을 의미하며, 0에 가까울수록 재사용성이 낮다는 것을 의미한다.

둘째, 재사용성을 기반으로 정제된 유즈케이스 목록에 포함된 유즈케이스들을 선정한다. (그림 7)은 객체지향 시스템의 기능 영역과 SOA 서비스 도메인 영역 간의 관계를 보여준다.

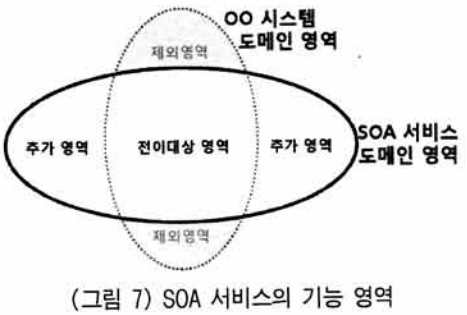
높은 재사용성을 제공하기 위해 객체지향 시스템의 도메인 영역을 (그림 7)와 같이 전이시켜야 한다. 객체지향 시스템의 도메인 영역 중 재사용성이 높은 영역을 SOA서비스의 영역으로 선정하고 ("전이대상 영역"), 도메인 지식과 사용자 요구사항을 이용하여 서비스를 가치를 높일 수 있도록 도메인 영역을 확장한다 ("추가 영역"). 반대로, 객체지향 시스템의 도메인 영역에서 재사용성이 낮거나, 서비스로서의 가치가 낮은 영역을 전이 대상 영역에 포함 하지 않는다 ("제외 영역"). "추가 영역"과 "제외 영역"을 결정하기 위해, <표 2>와 같은 기준을 제시한다.

위의 기준을 이용하여 SOA 서비스로 전이될 후보들을 결정하기 위하여 <표 3>을 사용된다. '유즈케이스 목록'은 목표 객체지향 시스템에서 제공되는 모든 유즈케이스들 뿐 아니라, UC_a와 UC_b과 같이 추가 영역의 유즈케이스 목록으로부터 도출된 목표 객체지향 시스템에서는 제공되지 않지만 다른 어플리케이션에서 제공되는 기능 이름도 포함한다. '재사용성 정도' 열은 <표 1>에서 측정된 값을 나타내며, 'SOA 서비스 전이 후보 여부'는 해당 유즈케이스가 SOA 서비스로 전이될 수 있는 후보인지를 명시한다. 재사용성 정도가 0에 근접한 값을 가지는 유즈케이스가 아닌 경우에만 '0'으로 표시 하고, '적용된 제외/추가 기준' 열은 SOA 서비스 전이 후보를 결정하는데 사용되는 기준을 기술한다.

셋째, <표 3>에서 결정된 SOA 서비스 전이 후보 대상이

〈표 1〉 공통성 분석표

유즈케이스 목록	어플리케이션 목록					재사용성정도
	목표 어플리케이션	잠재 어플리케이션 1	잠재 어플리케이션 2	…	잠재 어플리케이션 n	
UC_1	√	√	√	…	√	1
UC_2	√	√		…		0.9
…	…	…	…	…	…	…
UC_m	√	√	√	…	√	0.4



(그림 7) SOA 서비스의 기능 영역

〈표 2〉 서비스로 전이될 기능 영역 선정 기준

제외 기준	1) 해당 유즈케이스의 재사용성이 거의 낮아 잠재적인 ROI가 낮은 경우 (높은 재사용성을 갖는 서비스를 설계하려는 목적에 부합하지 않으므로) 2) 특정 플랫폼/벤더에 종속적인 기능인 경우(플랫폼에 대한 독립성이 떨어지고, 표준을 따르지 않음에 따라 이식성이 저하되므로) 3) 객체지향 시스템에서 제공했던 만큼의 좋은 품질의 컨텐츠를 제공할 수 없는 경우(전체적인 품질 저하를 유발시키므로) 4) 변화하는 비즈니스 환경에 대한 확장성이 낮은 경우 (서비스 전이로 인한 이점을 기대하기 어려움으로) 5) 복잡한 인터페이스를 필요로 하는 경우(자칫 특정 사용자에게 특성화된 서비스가 될 가능성이 있으므로)
추가 기준	1) 잠재적인 재사용성이 높아, 높은 ROI를 기대할 수 있는 경우(서비스로 전이할 경우 경제적인 수익을 창출할 수 있으므로) 2) 해당 기능에 대한 시장 가능성이 예상되는 경우(경제적 이점을 확보 할 수 있으므로) 3) 높은 품질의 기능 제공이 보장될 경우(전체적인 품질 향상을 유도시키므로)

〈표 3〉 SOA 서비스 전이 후보 결정 테이블

유즈케이스 목록	재사용성 정도	SOA 서비스 전이 후보 여부	적용된 제외/추가 기준
UC_1	1	0	
UC_2	0.9	0	
…	…	…	
UC_m	0.4	0	제외 기준 1)
UC_a		0	추가 기준 2)
UC_b		0	추가 기준 1)

되는 유즈케이스 간의 연관 관계를 고려하여, 정제된 유즈케이스 다이어그램을 작성한다.

서비스는 유즈케이스와는 달리 비교적 모듈성이 높은 기능을 제공한다. 그러므로, 각 유즈케이스 간의 연관 관계를 분석하여, 의존도가 높은 유즈케이스들은 하나의 유즈케이스로 병합하고 분리될 수 있는 기능성을 제공하는 유즈케이스를 여러 개의 유즈케이스로 분리한다.

유즈케이스 명세서(Description)은 유즈케이스의 기능성을 상세히 나타내기 때문에, 유즈케이스의 기능성을 파악하고 다른 유즈케이스와 공통성과 차이점을 분석 하는데 유용하다. 그러므로, 유즈케이스 정제를 위해 유즈케이스 명세서를 활용한다.

본 논문에서는 정제된 유즈케이스를 얻기 위해 (그림 8)과 같이 세 가지의 정제 형태를 제시하고 각 형태에 적용 가능한 기준을 다음과 같이 제시한다.

병합: 두 유즈케이스를 하나의 유즈케이스로 병합한다.

- 두 개 이상의 유즈케이스들이 상호 의존성이 높을 경우, 하나의 유즈케이스로 병합한다.
- UC_b와 UC_c가 «include»로 연결되어 있는 경우, 하나의 유즈케이스 실행 시 반드시 다른 하나의 유즈케이스가 실행되므로 의존도가 높다. 그러므로, 이런 유즈케이스들은 하나로 병합한다.
- UC_b와 UC_c가 «extends»로 연결되어 있는 경우, 하나의 유즈케이스가 실행을 완료하기 위해서 조건에 따라 다른 유즈케이스의 실행이 필요하므로 의존도가 다소 높다. 그러므로, 이런 유즈케이스들은 하나로 묶일 가능성이 높다.

분리: 하나의 유즈케이스를 두 개 이상의 유즈케이스로 분리한다.

- 해당 유즈케이스가 다소 큰 기능성을 제공하고 여러 개의 독립적인 기능성으로 분리될 수 있는 경우, 두 개 이상의 유즈케이스로 분리한다.

생성: 유즈케이스의 목록에 없는 새로운 유즈케이스를 생성한다.

- UC_b와 UC_c의 기능성 일부가 공통적인 기능인 UC_x을 포함하고, 공통적인 기능인 UC_x가 UC_b 또는 UC_c의 나머지 기능과 비교적 독립적인 경우 새로운 유즈케이스인 UC_x를 생성한다.

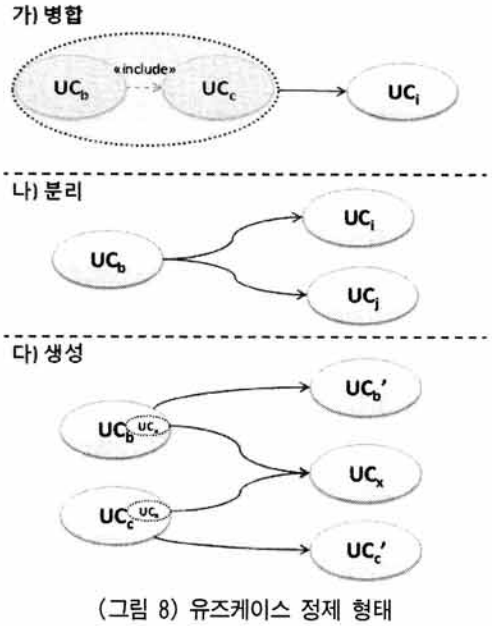
제시된 세 가지의 정제 형태를 적용하여 재사용성을 가지고 비교적 독립적인 기능을 수행하는 유즈케이스 목록을 획득하게 된다.

태스크 3. 서비스 도출: 잘 설계된 서비스는 높은 재사용성과 모듈성만 가지고 판단하기 힘들다. 그러므로, 이 두 가지 기준 외에 다른 기준을 적용하여 높은 품질을 보장할 수 있는 서비스를 도출한다.

첫째, 재사용성 외에 도메인에 특화된 다른 기준을 이용하여 정제된 유즈케이스를 평가한다. 재사용성 척도는 서비스를 평가하는 중요한 품질이지만, 4장에서 언급한 '비즈니스 요구사항 일치성'을 반영하지 않는다. 그러므로, 비즈니스 도메인의 특성을 잘 반영하도록 추가적인 평가 요인을 정의한다.

재사용성 외에 포함될 수 있는 평가 기준은 기능성 중요도, ROI, 시장성(Marketability)등이 될 수 있으며, <표 4>를 이용하여 각 정제된 유즈케이스가 서비스가 될 수 있는지를 평가한다. 각 평가 기준은 가중치를 설정하여 유즈케이스의 평가 값 계산에 활용 한다. 가중치는 도메인 전문가에 의해 각 유즈케이스의 중요도를 분석하여 설정 한다. 가중치에 대한 값은 1부터 0으로 한다.

'재사용성 정도'는 표 1에서 측정된 값을 이용하되 정제된 유즈케이스에 포함된 기존 유즈케이스의 수를 고려하여 다시 평가한다. 즉, 정제된 유즈케이스에 0.5와 0.7의 값을 가지는 유즈케이스가 포함되어 있다면, (0.5+0.7)/2를 하여 0.6으로 측정한다. '기능성 중요도'는 해당 유즈케이스가 도메인 내에서 얼마나 중요한 기능을 제공하는지를 나타내는 척도로 0과 1사이의 값을 도메인 지식에 의해 지정한다. 'ROI'는 해당 유즈케이스를 서비스로 전이할 경우 투자 대비 수익율을 나타내며, -1 과 1사이의 값을 지정하도록 한다. 여기서 음수의 값은 투자 대비 수익율이 낮음을 나타낸다. '시장성'은 서비스화가 된 해당 유즈케이스 기능의 잠재적인 발전가능성을 나타내며, 0과 1사이의 값을 시장성 분석을 통해 지정한다.



(그림 8) 유즈케이스 정제 형태

<표 4> 정제된 유즈케이스 평가표

정제된 유즈케이스 목록	재사용성 정도(가중치:v)	기능성 중요도(가중치:w)	ROI (가중치: x)	시장성 (가중치: y)	기타 기준(가중치:z)	평가값
UC_1	1	0.58	1	0.3	…	0.67
UC_2	0.9	0.7	0.8	0.5	…	0.58
…	…	…	…	…	…	…
UC_m	0.5	0.7	0.3	0.8	…	0.4

'평가값'은 여러 기준을 다음의 식을 이용하여 계산한다 (n은 적용된 추가 기준 개수).

$$\text{평가값} = \frac{(\text{재사용성정도} \times \text{가중치}_{\text{재사용성}} + \sum_{1}^{n}(\text{추가기준} \times \text{가중치}_{\text{추가기준}}))}{n+1}$$

둘째, 위의 표에서 평가된 평가값을 이용하여 서비스화가 될 수 있는 정제된 유즈케이스들을 도출한다. 높은 평가값을 가질수록 서비스가 될 확률이 높다는 것을 의미한다.

셋째, <표 5>와 같이 서비스 목록에 포함된 유즈케이스들을 나열하고, 각 서비스 별로 서비스 명세서를 작성한다. <표 5>에 기술되는 유즈케이스 목록은 기존의 유즈케이스를 포함하여 이후 서비스 인터페이스 또는 서비스 컴포넌트 도출 기법에 활용 한다.

서비스 명세서는 서비스 이름, 기능에 대한 개요, 참여 유즈케이스, BPMN 표기법을 이용한 비즈니스 프로세스 등을 포함하여 작성한다.

〈표 5〉 기능성에 따른 서비스 조합

서비스 목록	관련 있는 유즈케이스 목록
$Service_1$	UC_1, UC_2
...	...
$Service_k$	UC_9, UC_{10}

5.2 서비스 인터페이스 도출 기법

개요: 서비스 인터페이스 도출 기법에서는 유즈케이스 명세서와 시퀀스 다이어그램을 이용하여 사용자와의 상호작용의 수단을 제공하는 서비스 인터페이스를 도출한다. (그림 9)는 서비스 인터페이스를 도출하기 위한 세부 태스크와 산출물 간의 관계를 보여준다.

입/출력물: 서비스, 기능모델인 유즈케이스 모델을 입력물로 사용하며, 본 기법의 상세 활동을 통해 서비스 인터페이스가 생성된다.

태스크 1. 시퀀스 다이어그램 식별: 유즈케이스와 대응되는 시퀀스 다이어그램들을 추출한다. 유즈케이스의 정제 모델에 따라 시퀀스 다이어그램을 정제가 요구되기 때문에, 정제 전의 유즈케이스와 관련 있는 시퀀스 다이어그램을 식별한다.

태스크 2. 시퀀스 다이어그램 정제: 서비스 도출 과정에서 일부 유즈케이스는 정제되기 때문에 이에 완전하게 대응되는 시퀀스 다이어그램을 찾기 어렵다. 따라서, 5.1절의 다섯 가지 유즈케이스 정제 모델에 따라 시퀀스 다이어그램 역시 정제 되어야 한다.

병합: 병합된 유즈케이스를 위해 관련 있는 시퀀스 다이어그램을 병합한다.

- 병합된 유즈케이스들이 각각 사용자와의 상호작용을 필요로 한다면, 중복된 상호작용을 제거함으로써 사용자와의 상호작용을 부분적으로 통합한다.
- 사용자와 상호작용이 없는 «include» 또는 «extends»관계의 유즈케이스가 병합된다면, 사용자와의 추가적인 상호작용을 고려하지 않고, 엔티티 클래스만을 추가적으로 포함 한다.

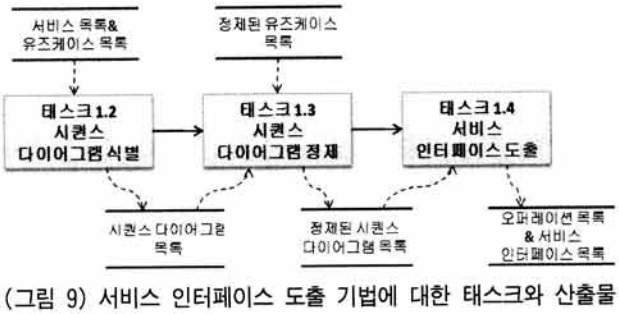
분리: 분리된 유즈케이스를 위해, 해당 시퀀스 다이어그램을 두 개 이상의 시퀀스 다이어그램으로 분리한다.

- 시퀀스 다이어그램이 분리되면, 사용자와의 상호작용이 새롭게 요구될 수 있다. 따라서, 분리된 유즈케이스에서 제공하는 기능성을 만족할 수 있도록 분리된 시퀀스 다이어그램에 사용자와의 상호작용을 추가 한다.

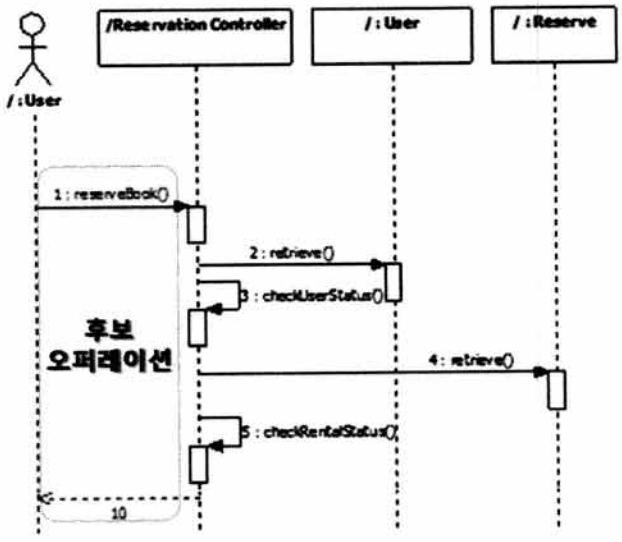
생성: 새롭게 생성된 유즈케이스를 위해, 새로운 시퀀스 다이어그램을 생성한다. 이는 일반적인 객체 지향 방법론을 이용하여 시퀀스 다이어그램을 작성한다.

태스크 3. 서비스 인터페이스 도출: 정제 과정을 거친 시퀀스 다이어그램에서 사용자와의 상호작용 부분을 추출하고 각각을 서비스 인터페이스의 오퍼레이션으로 도출한다.

서비스와 관련된 모든 시퀀스 다이어그램에서 사용자와의 모든 상호작용은 (그림 10)과 같이 서비스 인터페이스를 위한 오퍼레이션 후보로 선정된다. 상호작용에서 입/출력은 쌍을 이뤄 하나의 오퍼레이션으로 도출하고 도출된 오퍼레이션들의 집합은 서비스 인터페이스를 구성하게 된다.



(그림 9) 서비스 인터페이스 도출 기법에 대한 태스크와 산출물 관계



(그림 10) 유즈케이스의 후보 오퍼레이션

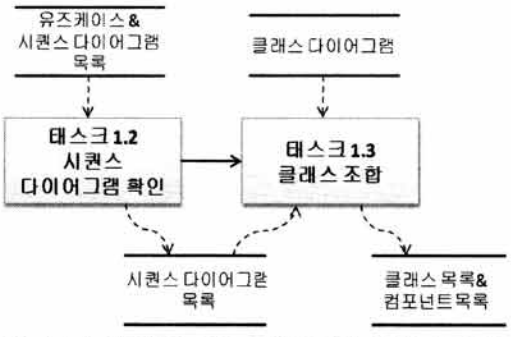
5.3 서비스 컴포넌트 도출 기법

개요: 서비스 컴포넌트 도출 기법은 정적 모델과 동적 모델을 이용하여 서비스를 구현하는 서비스 컴포넌트를 도출

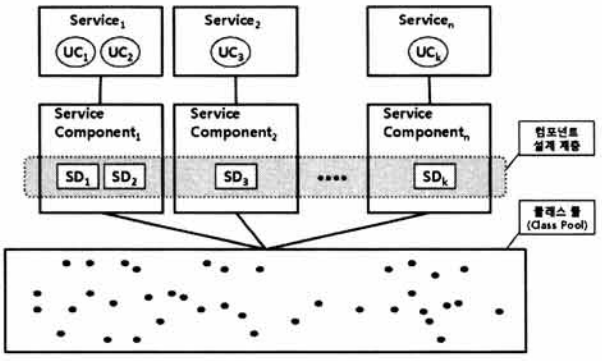
한다. (그림 11)은 서비스 컴포넌트를 도출하기 위한 세부 태스크와 산출물 간의 관계를 보여준다.

서비스는 하나 이상의 유즈케이스에서 도출되기 때문에, 서비스 컴포넌트는 객체지향 시스템의 시퀀스 다이어그램의 조합으로 도출된다. 컴포넌트의 기능성을 구현하는 구현체를 확인하기 위해 클래스 모델의 클래스 다이어그램도 함께 활용한다. 대표적인 아키텍처 구조인 MVC 아키텍처에서는 일반적으로 시퀀스 다이어그램에 컨트롤러 클래스를 구현하고 있다. 따라서, 컨트롤러 클래스를 활용하여 서비스 컴포넌트를 도출한다. (그림 12)는 유즈케이스(UC)와 시퀀스 다이어그램(SD)의 관계를 활용하여 서비스에 대응되는 서비스 컴포넌트를 도출하는 과정을 나타내고 있다. 그림에서 서비스 컴포넌트 설계 계층은 서비스 컴포넌트 도출을 위한 시퀀스 다이어그램의 조합 된다.

입/출력물: 기능모델인 유즈케이스 모델, 동적 모델인 시퀀스 다이어그램을 입력물로 사용하며, 본 기법의 상세 활동을 통해 서비스 컴포넌트가 도출된다.



(그림 11) 컴포넌트 도출 기법에 대한 태스크와 산출물 관계

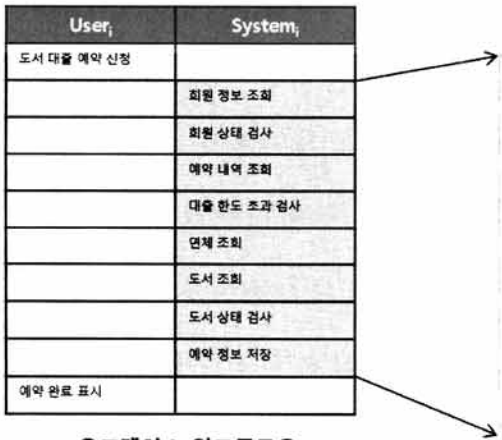


(그림 12) 서비스 컴포넌트

태스크 1. 시퀀스 다이어그램 확인: 서비스의 기능성을 만족하기 위해, 유즈케이스에 대응되는 시퀀스 다이어그램을 확인하여 서비스 컴포넌트를 구성한다. 시퀀스 다이어그램의 확인은 5.2절에서 확인된 유즈케이스와의 관계 정보를 활용한다. 관계 정립은 유즈케이스의 워크플로우와 시퀀스 다이어그램의 시간 흐름에 따른 메시지들을 비교하며 수행한다. 시퀀스 다이어그램은 메시지의 의미를 상세히 기술하지 않기 때문에, 직접 유즈케이스와의 비교가 쉽지 않다. 따라서 클래스 다이어그램과 명세를 통해 클래스와 클래스의 기능이 가지는 의미를 파악 한다.

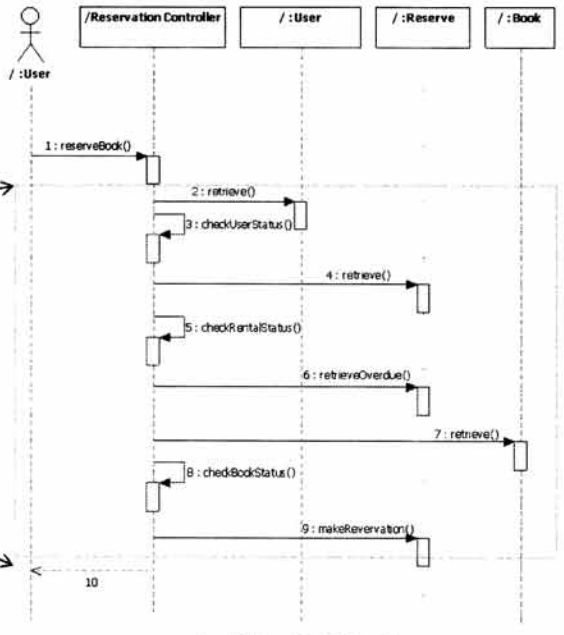
유즈케이스가 가지고 있는 워크플로우의 각 단계와 시퀀스 다이어그램의 시간의 순서에 따른 메시지를 비교하여 서로가 대응된다면, 서비스 컴포넌트를 구성하기 위한 시퀀스 다이어그램으로 확인 한다. (그림 13)는 유즈케이스와 이와 대응되는 시퀀스 다이어그램을 식별하기 위해 유즈케이스의 워크플로우와 시퀀스 다이어그램을 비교하는 것을 보여준다.

태스크 2. 클래스 조합: 도출된 서비스에 대응하는 서비스 컴포넌트는 하나 이상의 시퀀스 다이어그램 또는 하나



$User_i$	$System_i$
도서 대출 예약 신청	
	회원 정보 조회
	회원 상태 검사
	예약 내역 조회
	대출 한도 초과 검사
	연체 조회
	도서 조회
	도서 상태 검사
	예약 정보 저장
예약 완료 표시	

a. 유즈케이스 워크플로우



b. 시퀀스 다이어그램

(그림 13) 시퀀스 다이어그램 확인

이상의 컨트롤러 클래스의 조합으로 구성된다. 따라서, 시퀀스 다이어그램의 조합은 서비스의 기능성, 서비스 인터페이스 그리고 컨트롤러 클래스의 형태에 따라 적당한 방법을 사용해야 한다. 시퀀스 다이어그램을 활용한 서비스 인터페이스도출 과정에서 시퀀스 다이어그램은 정제과정을 거치게 된다. 따라서, 사용자와 엔티티 클래스 사이에 메시지의 중재자 역할을 하는 컨트롤러 클래스에 대한 조합이 필요하다. 본 절에서는 서비스 컴포넌트를 도출하기 컨트롤러 클래스의 조합을 방법을 중재(Mediation)과 생성 두 가지 형태로 제시 한다.

태스크 2-1. 중재: 정제된 시퀀스 다이어그램은 하나 이상의 시퀀스 다이어그램의 조합으로 구성되며, 정제된 시퀀스 다이어그램에 맞추어 컨트롤러 클래스의 조합이 필요하다. 조합 방법은 조합 형태에 따라 세 가지 방법으로 제시한다. 객체지향 시스템이 보유하고 있는 어떠한 소스의 변경 없이 서비스 컴포넌트를 도출하기 위해 컨트롤러 클래스와 엔티티 클래스를 수정하지 않고 중재자 역할의 클래스를 추가하여 서비스 컴포넌트를 도출 한다.

단순 중재 조합(Simple Mediation Composition)

서비스 컴포넌트가 하나 이상의 컨트롤러 클래스로 조합되고, 서비스의 기능성을 구현하기 위해 다른 컨트롤러 클래스를 참조할 필요가 없는 경우 단순 중재 형태로 서비스 컴포넌트를 도출 한다. 이 경우 객체지향 시스템의 컨트롤러 클래스는 별도의 추가나 변경 없이 조합되어 서비스 컴포넌트로 도출 된다. <표 6>와 같이 서비스$_i$ 의 시퀀스 다이어그램$_A$ 는 컨트롤러 클래스$_C$ 와 대응되고, 시퀀스 다이어그램$_B$ 는 컨트롤러 클래스$_D$와 대응된다면 서비스 컴포넌트$_j$는 두 개의 컨트롤러 클래스를 단순 중재 조합함으로써 서비스 컴포넌트를 도출 한다.

기능 병합(Function Composition)

서비스에서 요구하는 기능성을 하나의 컨트롤러 클래스$_B$로 만족시키지 못하고 컨트롤러 클래스를 조합하기엔 컨트롤러 클래스가 다소 클 경우, 컨트롤러 클래스를 완전하게 병합하지 않고 추가 기능을 보유하고 있는 클래스만을 병합하여 서비스 컴포넌트를 도출한다. <표 7> 에서 정제된 시퀀스 다이어그램$_{A'}$는 컨트롤러 클래스$_C$와 컨트롤러 클래스$_D$를 요구하고 있지만, 엔티티 클래스$_E$의 기능만으로도 만족하기 때문에, 컨트롤러 클래스$_D$를 조합하지 않는다. 단순 중재 조합을 사용하는 경우, 필요하지 않은 기능을 제공하는 엔티티 클래스도 포함될 수 있다. 서비스에서 요구하는 기능성이 비교적 적고, 엔티티 클래스와의 조합으로도 만족시킬 수 있다면, 기능 병합을 통해 서비스 컴포넌트를 도출 한다.

〈표 6〉 단순 조합

서비스$_i$		연결관계	서비스컴포넌트$_j$
정제된 시퀀스 다이어그램$_{A'}$	시퀀스다이어그램$_A$	→	컨트롤러 클래스$_C$
	시퀀스다이어그램$_B$	→	컨트롤러 클래스$_D$

〈표 7〉 서비스 컴포넌트의 기능성 병합

서비스$_i$		연결관계	서비스컴포넌트$_j$
정제된 시퀀스 다이어그램$_{A'}$	시퀀스다이어그램$_A$	→	컨트롤러 클래스$_C$
	시퀀스다이어그램$_B$	→	컨트롤러 클래스$_D$ → 엔티티 클래스$_E$

태스크 2-2. 생성: 새롭게 생성된 유즈케이스에 대한 객체지향 시스템의 클래스를 찾을 수 없기 때문에, 추가된 기능성에 맞는 클래스를 새로이 작성해야 한다. 추가된 클래스는 중재 기법을 적용하여 서비스 컴포넌트의 구성요소가 된다.

태스크 2-3. 분리: 유즈케이스가 분리된 경우, 시퀀스 다이어그램도 분리 기법을 적용하여 정제한다. 유즈케이스이스의 일부분을 새로운 분리하며 새로운 유즈케이스를 생성하였듯이, 시퀀스 다이어그램에서 컨트롤러 클래스를 확인하여 해당 기능을 가진 클래스를 시퀀스 다이어그램에서 분리한다. 정제된 유즈케이스가 서비스로 도출되었기 때문에, 정제된 시퀀스 다이어그램을 기반으로 하는 컨트롤러 클래스를 서비스 컴포넌트로 도출한다.

6. 사례 연구

본 연구에서는 제안한 전이기법의 유효성과 실용성을 보여주기 위하여 여행정보시스템을 사례연구로 선정하였다. 5장에서 제시한 기법에 여행정보시스템의 산출물을 적용하여 사례 연구를 수행한다.

6.1 서비스 도출

여행정보시스템의 기능 모델 산출물을 정제하여 재사용성이 고려된 유즈케이스를 선별하고, 이를 서비스 추출을 기반으로 활용한다.

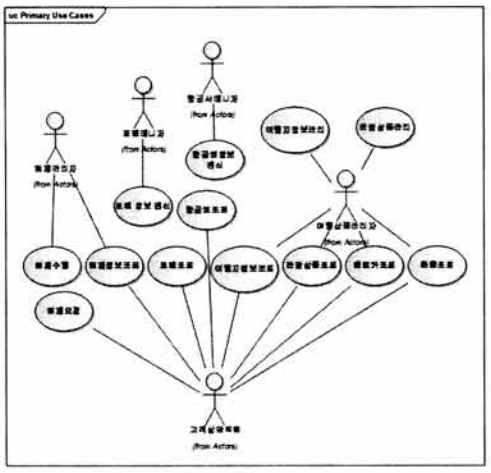
태스크 1. 도메인 정의

여행정보시스템(Travel Itinerary System - TIS)은 숙박, 교통, 여행지에 대한 정보 제공기능과 숙박과 교통, 관광을 위한 이용권 발권 및 예약기능을 포함하고 있다. 또한 고객관리와 유치를 위해 회원제로 운영되며, 따라서 회원관리를 위한 기능을 갖추고 있다. (그림 14)와 같이 유즈케이스 모델로 표현된다. TIS가 호텔과 항공권, 관광상품에 대한 조회와 예약, 구매/결제, 그리고 해외 여행객을 위한 환율 조회 기능을 가지고 있음을 확인 하고 여행상품정보를 제공하고 있는 A사, I사, H사, M사, N사의 어플리케이션을 후보로 선정한다.

태스크 2. 유즈케이스 정제

선별된 유즈케이스에서 재사용성과 가치가 높은 유즈케이스를 선별하기 위해, 유즈케이스를 정제한다. 후보 어플리케이션의 관리 기능은 각 어플리케이션에 종속적이고, 관찰하기 어렵기 때문에 공통성 분석표에서 제외한다. 따라서, TIS의 관리 기능도 공통성 분석표에서 제외 한다. <표 8>은 TIS의 공통성 분석표를 보여준다. 가독성을 높이기 위해, 각 유즈케이스에 U01부터 U18까지의 번호를 부여 한다

<표 8>의 재사용성 평가에서 U01은 모든 어플리케이션에서 사용되는 기능성으로 1으로 평가된다. U05부터 U13까



a. TIS 의 유즈케이스 모델

A사 기능목록	
항공	조회,예약,구매,
호텔	조회
철도	조회,구매
여행자보험	조회,가입

I사 기능목록	
항공	조회,예약,구매
호텔	조회,예약대행
패키지	조회,예약,결제
콘도/펜션	조회,예약

H사 기능목록	
항공	조회,예약,구매
호텔	조회,예약
패키지상품	조회,예약,결제
크루즈여행	조회,결제
박람회	조회

M사 기능목록	
항공	조회,예약,구매
호텔	조회
환율	조회
날씨	조회

N사 기능목록	
항공	조회,예약,구매
호텔	조회
패키지상품	조회
렌트카	조회,예약,결제

b. 후보 어플리케이션의 유즈케이스 목록

(그림 14) 여행정보시스템의 유즈케이스 모델

〈표 8〉 공통성 분석표

유즈케이스	어플리케이션 목록						재사용성 정도
	TIS	A사	I사	H사	M사	N사	
U01-항공권조회	√	√	√	√	√	√	1
U02-항공권예약	√	√	√	√	√	√	1
U03-항공권구매	√	√	√	√	√	√	1
U04-호텔조회	√	√	√	√	√	√	1
U05-호텔예약	√	√	√	√			0.66
U06-호텔결제	√	√					0.33
U07-여행지정보조회	√						0.16
U08-관광상품조회	√		√				0.33
U09-렌터카조회	√					√	0.33
U10-렌터카예약	√					√	0.33
U11-렌터카결제	√					√	0.33
U12-환율조회	√				√		0.33
U13-결제요청	√						0.16
U14-결제수행	√	-	-	-	-	-	-
U15-여행지정보관리	√	-	-	-	-	-	-
U16-관광상품관리	√	-	-	-	-	-	-
U17-항공권정보갱신	√	-	-	-	-	-	-
U18-호텔정보갱신	√	-	-	-	-	-	-

지의 기능성들은 일부 어플리케이션에만 포함되기 때문에 재사용성 점수가 낮다.

$$\text{U01의 재사용성정도} = \frac{6(\text{유즈케이스를 제공하는 어플리케이션 수})}{6(\text{총 어플리케이션수})}$$

<표 9>는 각 정제된 유즈케이스에 대하여 전이 후보 유즈케이스를 선정한 결과를 보여준다. U01, U02, U03, U04, U05는 0.5 보다 큰 값으로 평가 되었음으로 전이 후보로 선정 한다. U07부터 U13는 재사용성이 낮으므로, 표 2의 제외기준 1번 의해 서비스 전이 대상에서 제외된다. U06은 재사용성이 낮지만, 추가 기준 1번과 2번에 해당되는 것으로 판단하여, 서비스 전이 대상에 포함한다. AU01과 AU02는 A사의 기능성에 포함된 철도 예약과 구매 기능성이다. TIS에 없는 기능성이기 때문에 AU(*Additional Usecase*)로 시작하는 번호를 부여 한다. 두 유즈케이스의 재사용성은 0에 가깝다. 하지만, 항공편과 연계되는 철도서비스는 상승효과를 기대할 수 있고 잠재적 재사용성이 있다고 판단된다. 따라서, 추가 기준 1과 2를 적용하여 전이 대상에 포함한다.

전이 대상으로 선별된 U03, U06, AU02는 각 항공권, 호텔, 철도이용을 위한 구매 행위이다. 구매행위에는 결제수단 선택과 결제행위가 포함된다. 결제 행위는 TIS가 확장되어 추가되는 다른 기능성에서 사용될 수 있다. 만약, 차량렌트에 대한 수용가 급증하여 차량대여 기능을 추가한다면, 결제 기능은 변경없이 재사용 가능하다. 따라서, 5.1절의 태스크 2에 제시한 '분리' 기법을 적용하여 (그림 15)와 같이 결제 기능을 분리하여 하나의 유즈케이스로 추출/생성하고 생성된 유즈케이스는 U03, U06, AU02와 ≪include≫ 관계를 형성 한다. 분리된 유즈케이스는 추가로 생성되었기 때문에 AU03이 된다.

태스크 3. 서비스 도출

정제된 후보 유즈케이스는 <표 10>에 나열한다. <표 8>에서 평가된 값을 이용하여 재사용성에 대한 값을 작성한다. U03'과 UC06'은 기능성에 변화가 없기 때문에 재사용성 값의 변동은 없다.

〈표 9〉 SOA 서비스 전이 후보 결정 테이블

유즈케이스 목록	재사용성 정도	SOA서비스 전이 후보여부	적용된 제외/추가기준
U01	1	○	추가기준 1)
U02	1	○	추가기준 1)
U03	1	○	추가기준 1)
U04	1	○	추가기준 1)
U05	0.66	○	추가기준 1)
U06	0.33	○	추가기준 1), 2)
U07	0.16	X	제외 기준 1)
U08	0.33	X	제외 기준 1)
U09	0.33	X	제외 기준 1)
U10	0.33	X	제외 기준 1)
U11	0.33	X	제외 기준 1)
U12	0.33	X	제외 기준 1)
U13	0.16	X	제외 기준 3)
AU01	-	○	추가기준 1), 2)
AU02	-	○	추가기준 1), 2)

a. 전이 후보 결정 테이블

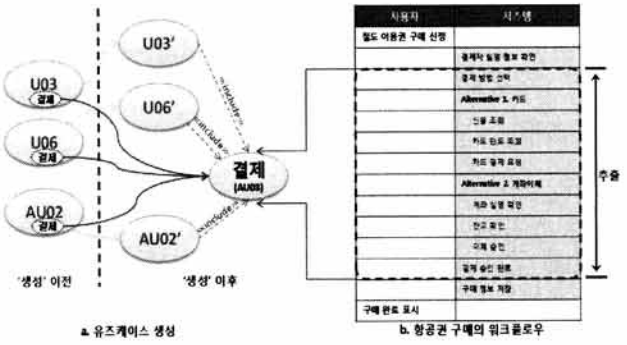
A사 기능목록	
항공	조회, 예약, 구매,
호텔	조회
철도	**조회, 구매**
여행자 보험	조회, 가입

b. A사 기능목록

철도 조회	
유즈케이스	AU01
명세	검색 지역의 이용 가능한 철도 항목을 조회
사전조건	-
사후조건	이용 가능한 철도 항목 표시

철도이용권 구매	
유즈케이스	AU02
명세	이용 하고자 하는 철도 사용권을 구매
사전조건	철도의 Code를 조회 후 구매
사후조건	-

b. 철도의 조회와 구매의 유즈케이스 명세



a. 유즈케이스 생성 b. 항공권 구매의 워크 플로우

(그림 15) 결제 유즈케이스 생성

$$(AU01, AU02') 0.33 = \frac{2(\text{유즈케이스를 제공하는 어플리케이션 수})}{6(\text{총 어플리케이션 수})}, \quad (AU03) 1 = \frac{6(\text{유즈케이스를 제공하는 어플리케이션 수})}{6(\text{총 어플리케이션 수})}$$

AU01, AU02'의 재사용성 값은 <표 8>에 대입하여 0.33을 획득 하고, AU03은 도메인의 모든 어플리케이션에서 결제 기능을 필요로 하기 때문에 1으로 평가 한다. TIS는 교통상품에 대한 거래가 주 업무 이다. 특히, 해외 관광객을 위한 항공권 예약과 판매가 주 목적에 포함된다. 따라서 항공권 거래를 위한 조회, 예약, 판매에 해당 하는 U01, U02, U03'은 기능성 중요가 높게 평가되어 1으로 하고 역시 증가하는 여행객의 추이를 감안하여 ROI와 시장성을 1으로 평가 한다. 호텔의 조회와 예약, 구매 기능은 기능성 중요도는 1으로 높게 평가 되지만 제반 정보가 빈약하여 ROI와 시장성은 0.5으로 낮게 평가 한다. 철도이용권은 해외 지역에 따라 이용빈도수가 일정치 않다라고 판단되어 잠재적 가능성은 인정되나, 현재의 시장성과 ROI는 낮은 편으로 평가 된다. 따라서, 앞으로의 잠재적 시장 수요를 생각 하여 0.5를 부여 한다. 결제기능은 모든 어플리케이션에서 요구 하는 기능이고, 앞으로 추가될 서비스에서 재사용 가능한 기능성이기 때문에 기능성 중요도와 ROI가 1으로 높게 평가 된다. 또한 결제 기능은 종속성이 낮은 기능성으로 다른 도메인에서도 사용 가능하다. 따라서 시장성도 1으로 평가 한다.

〈표 10〉 정제된 유즈케이스 평가표

정제된 유즈케이스 목록	재사용성 정도	기능성 중요도	ROI	시장성	평가값
	가중치:1	가중치:0.8	가중치:0.7	가중치:0.5	
U01	1	1	1	1	0.75
U02	1	1	1	1	0.75
U03'	1	1	1	1	0.75
U04	1	1	0.5	0.5	0.6
U05	0.66	1	0.5	0.5	0.51
U06'	0.33	1	0.5	0.5	0.43
AU01	0.33	0.5	0.5	0.5	0.33
AU02'	0.33	0.5	0.5	0.5	0.33
AU03	1	1	1	1	0.75

정제를 거치며 변형되었거나, 추가된 유즈케이스에 대한 가치가 변경되어 평가가 필요하기 때문에, 정제된 유즈케이스 평가표의 값은 평가 값 계산식을 활용하여 유즈케이스의 가치를 산술 한다. 항공권 구매와 관련한 U01, U02, U03'은 주요 업무에 해당 하여 각 영역별로 높게 평가되고 계산된 평가 값도 0.75으로 상당히 높게 나왔다. 따라서, TIS시스템에서 높게 평가된 항공권 구매 관련 기능과 결제 기능을 서비스로 전이한다.

$$0.75(U01) = \frac{((1\times1)+(1\times0.8)+(1\times0.7)+(1\times0.5))}{3+1}$$

서비스 전이 대상으로 선정된 U01, U02, U03'는 표 11에 작성한다. 유즈케이스 정제 과정을 중 변형이 있었던 U03'은 AU03과 새로운 관계를 형성하고 있음으로 관련 목록에 작성 한다. AU03은 U03, U06, AU02부터 공통적인 부분을 추출하였기 때문에 셋 모두를 기입 한다. 서비스 목록에 작성되는 서비스의 코드는 서비스로 식별 할 수 있게 U에서 S로 변경 한다.

〈표 11〉 기능성에 따른 서비스 조합

서비스 목록	관련 있는 유즈케이스 목록
S01	U01
S02	U02
S03	U03, AU03
AS03	U03, U06, AU02, AU03

유즈케이스 기반으로 선정된 각 서비스는 유즈케이스 명세를 기반으로 서비스 명세를 작성하며, <표 12>는 S01에 대한 서비스 명세서를 보여준다.

〈표 12〉 S01의 서비스 명세 작성

서비스 이름	항공권 조회
서비스 기능성	S01 서비스는 항공편을 조회를 필요로 하는 고객을 위한 기능성이다. 도착지, 출발지, 출발 시간, 도착 시간, 여권번호, 좌석 타입등 사용자가 입력한 조건에 따라 비행목록을 반환한다.
사전 조건	N/S
사후 조건	결과가 없을 경우 비어 있는 값을 반환환다.
관련 유즈케이스	U01

6.2 서비스 인터페이스 도출

도출된 S01서비스에 대응되는 서비스 인터페이스는 관련 유즈케이스 U01의 명세서에서 도출 한다. S01의 인터페이스에 포함되는 오퍼레이션은 U01에 대응되는 시퀀스 다이어그램의 사용자 상호작용을 이용한다.

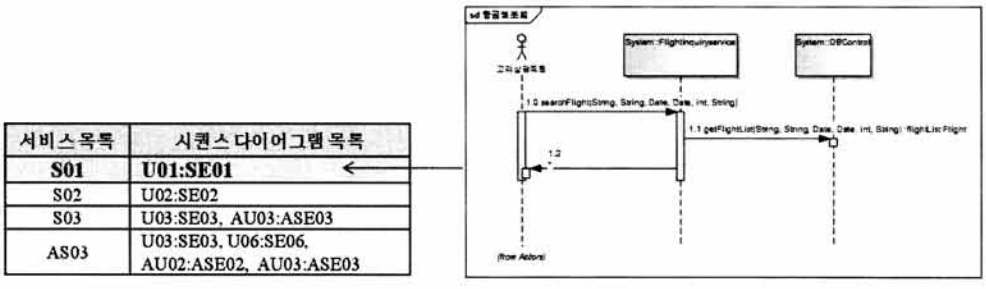
태스크 1. 시퀀스 다이어그램 식별

서비스로 도출에 활용된 U01, U02, U03, U06 에 대응되는 유즈케이스 목록으로 추출한다. TIS시스템의 동적모델 산출물에서 추출 하며 AU02, AU03은 TIS의 산출물에 포함되지 않기 때문에 태스크 2에서 새로 작성 한다. 시퀀스 다이그램의 코드는 SE로 한정 한다. 따라서, S01의 인터페이스를 도출 하기 위해 활용하는 시퀀스 다이어그램은 SE01이 되고 (그림 16)과 같이 표에 목록을 작성한다.

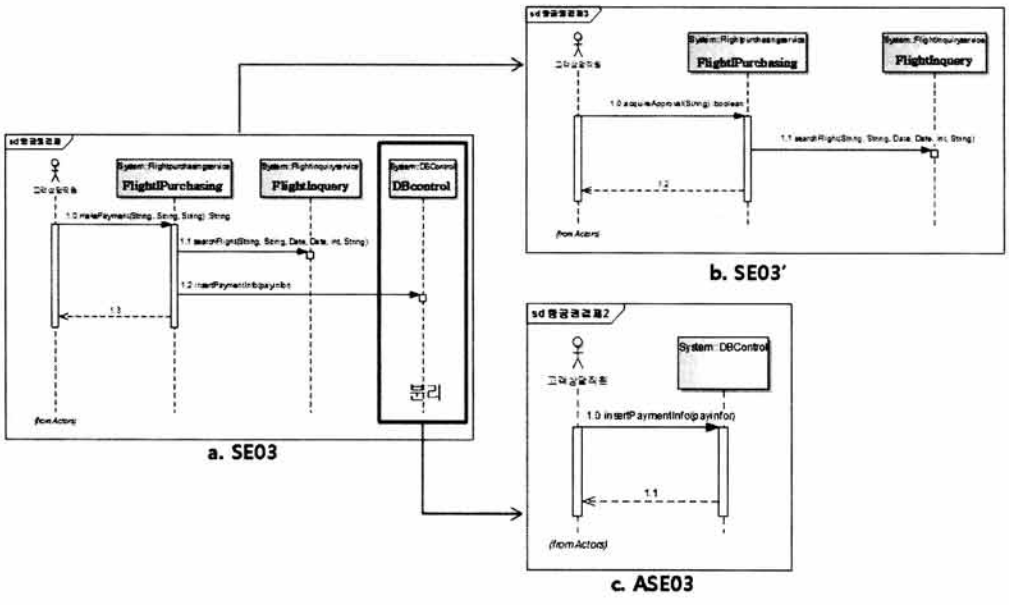
태스크 2. 시퀀스 다이어그램 정제

S03과 AS03은 유즈케이스 정제 과정이 포함되어 있다. 따라서 관련된 시퀀스 다이어그램도 유즈케이스에 맞춰어 정제한다. S03은 AU03을 분리하였고, AU03은 새로이 생성되었기 때문에, 각 시퀀스 다이어그램을 정제하고 생성 한다.

(그림 17)은 AU03에 대응하는 ASE03의 생성과정을 보여준다. 서비스 도출에서 U03의 결제 부분만을 AU03으로 생성하고 나머지를 U03'으로 생성하였기 때문에, ASE03은



(그림 16) 시퀀스 다이어그램 식별



(그림 17) 시퀀스 다이어그램 병합

□ 일반명세

오퍼레이션	고객이 결제 가능한지 자격 조건을 조회한다.
사전조건	N/A
사후조건	N/A

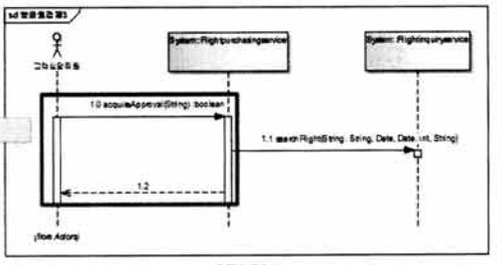
□ 입력인자

Name	Type	Description
customerID	String	고객 고유 ID

□ 반환타입

Name	Type	Description
right	boolean	결제 가능한 사용자 여부 판단

a. $S03'_{OP}$ - ACQUIREAPPROVAL



a. SE03'

(그림 18) 서비스 인터페이스 도출

SE03에서 결제 부분을 추출하여 생성하고 SE03'은 SE03에서 결제 부분을 제외한 시퀀스 다이어그램으로 생성 한다.

태스크 3. 서비스 인터페이스 도출

서비스 인터페이스의 오퍼레이션은 정제된 시퀀스 다이어그램의 상호작용으로부터 추출 한다. 따라서, S01, S02, S03, AS03에 대한 서비스 인터페이스의 오퍼레이션은 각 정제된 SE01, SE02, SE03', ASE03의 상호작용 부분을 추출하여 작성 한다. (그림 18)은 정제된 SE03'에서 상호작용을 서비스 인터페이스의 오퍼레이션으로 추출 하고 있다.

5.3 서비스 컴포넌트 도출

태스크 1. 시퀀스 다이어그램 확인

유즈케이스와 시퀀스 다이어그램을 활용하여 서비스 컴포넌트를 구성하는 클래스를 식별한다. 6.2절에서 서비스 인터페이스를 도출하기 위해 작성한 시퀀스 다이어그램 목록을 활용한다.

태스크 2. 클래스 조합

서비스 인터페이스를 도출할때, SE01, SE02는 변경 없이 재사용되며, SE03은 기능을 분리하고, ASE03은 생성되었다. 따라서, SE03은 5.3절에서 제시한 클래스 조합 기법 중 '병합' 기법에 해당하고, ASE03은 '분리'에 해당한다.

〈표 13〉 시퀀스 다이어그램 목록

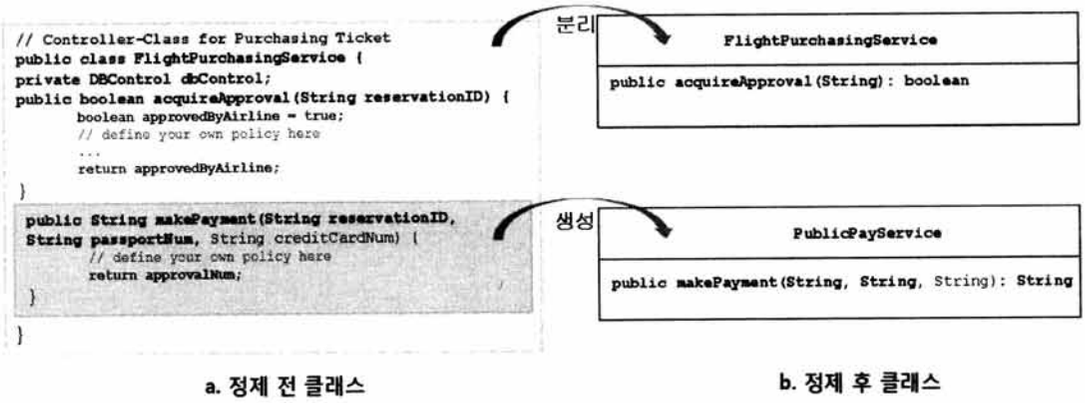
서비스목록	시퀀스다이어그램목록
S01	SE01
S02	SE02
S03	SE03'
AS03	ASE03

(그림 19)에서 항공권 결제(FlightPurchasingService) 컨트롤러 클래스는 정제된 유즈케이스에 대응하여 항공권 구매(FlightPurchasingService)와 결제(PublicPayService) 두 개의 컨트롤러 클래스로 분리된다. FlightPurchasingService 클래스에서 결제 기능을 수행하는 makePayment메소드를 분리하고, 분리한 메소드는 새롭게 생성한 PublicPayService클래스에 추가 된다. 정제가 완료 된 각각의 클래스는 그림 17과 도출한 서비스 인터페이스와 대응 되며, 서비스 컴포넌트의 자격을 갖는다.

7. 평가 및 결론

본 논문에 제안한 방법을 평가하기 위해 SWEBOK[18]에서 제시한 일곱 가지의 소프트웨어 프로세스 평가 기준을 적용한다. 평가 기준에 따라 적합여부를 판단하고 Support (O), Semi-Support(△), Not-Support(X) 의 세 단계로 구분지어 평가하고, <표 14>은 방법론 별 비교 평가결과를 나타낸다.

'프로세스의 체계적 구성'은 제안된 프로세스가 단계, 절차, 세부단계와 같이 일정한 업무단위에 따라 수행될 수 있도록 작성되었는지를 평가한다. '지침의 상세화'는 제안된 지침과 활동이 다음 단계에 일관성 있게 전달 하도록 상세히 기술되었는지를 평가하며, '산출물의 명세'는 핵심 항목과 템플릿, 예제 그리고 관련된 가공물과 활동내역이 산출물에 얼마나 상세히 기록되는지를 평가한다. '추적성'은 산출물간



a. 정제 전 클래스 b. 정제 후 클래스

(그림 19) 컨트롤러 클래스 조합

〈표 14〉 다른 연구와 비교 평가(O: Support, △: Semi-Support, X: Not-Support)

방법론 평가기준	Belushi[6]	Sneed[7]	Wang[8]	Kim[4]	제안된 기법
프로세스 체계성	△	△	△	O	O
지침의 상세화	X	X	X	△	O
산출물 명세	X	X	△	O	O
추적성	△	△	△	△	O
적용성	O	O	O	O	△
간결성	O	O	O	O	O
SOA 적합성	O	O	△	△	△

의 항목들이 추적가능하고 그에 따른 활동이 얼마나 잘 정의되었는지에 따라 평가된다. '적용성'은 잘 정의된 프로세스를 실제 프로젝트에 적용 할 때 사용자가 이해하기 쉽고 타당성 있게 정의 되는지 여부를 평가하며 '간결성'은 프로세스가 불필요한 산출물과 활동들을 만들지 않고 오직 최소한의 노력으로 바람직한 결과를 얻도록 작성되어야 하는지는 나타낸다.

본 논문에서 제안된 기법은 객체지향 시스템의 산출물로부터 웹서비스가 도출 될 때까지 단계별로 나누고 각 단계별 상세 단계를 나누어 제시하였기 때문에 '프로세스의 체계적 구성'을 Support로 평가 하였다. 각 상세 단계별 지침을 상세히 기술 하였기 때문에 '지침의 상세화'를 Support로 평가 하였다. 산출물은 UML을 충분히 활용하였고, 적절한 표와 도식을 제시하였으며 다음 단계에서 산출물로 활용하고 있기 때문에 '산출물의 명세'와 '추적성'을 Support로 평가 하였다. 제안된 기법을 활용하여 실제 프로젝트에 적용해 보시 못 했기 때문에 적용성은 Semi-Support로 평가하고 객체지향의 산출물을 최대한 활용하고 발생되는 산출물은 객체지향 산출물과 동떨어지지 않는 범위를 유지 하였기 때문에 '간결성'도 Support로 평가 하였다. 마지막 'SOA 적합성'은 본 논문의 기법이 온전히 검증과정을 거치지 않았기 때문에 Semi-Support로 평가한다.

SOA는 기업의 생산성과 유연성 향상과 인프라 유지비용의 최소화에 도움이 될 거라 기대되어 산업계에서 수요가 증가하고 있다. 그러나 기존의 시스템을 배제하고 새로운 SOA시스템을 도입하기에는 비용과 시간 두 가지 측면에서 많은 위험을 안고 있다. 따라서 기존의 시스템을 활용하여 점진적인 SOA서비스로의 전이에 대한 시도가 나타나고 있다.

본 논문에서는 이러한 요구를 만족시키기 위해 기존의 산출물과 코드를 최대한 활용함으로써, 재 구축에 대한 비용과 시간적 부담과 그리고 신규 시스템이 가지고 있는 잠재적인 위험을 보다 낮추는 방법론을 제안하고 있다.기존의 시스템에 변경을 최재한 자제하고 산출물에 대한 활용도를 최대한 끌어 올리려 시도 하고 있다. 객체지향 시스템의 산출물과 서비스 산출물간의 관계를 정의하고 SOA서비스를 구성하는 서비스와 서비스 인터페이스 그리고 서비스 컴포넌트 각 단계별로 활용 가능한 객체지향 산출물을 식별 하고, 식별한 객체지향 시스템의 산출물을 입력물로 하고 이에 대응하여 SOA 서비스의 산출물을 출력물로 식별하고 있다. 출력물을 도출하는 태스크를 제시하고 상세한 활동과 지침을 제시하고 있다. 각 과정에서 발생한 산출물과 도출과정에 사용된 입력물은 다음 단계의 산출물을 도출에 활용되며 도출과정에 사용된 입력물들은 추적성을 유지하고 있는 관계이기 때문에 서비스의 산출물간 관련성도 유지하게 된다. 또한 객체지향 시스템의 코드에 대한 변화를 줄일 수 있는 방법을 제시하고 있기 때문에 운영되고 있는 객체지향 시스템을 유지한체로 서비스를 도입할 수 있는 이점을 포함하고 있다.따라서, 기업의 변화는 최대한 줄이고 안정적으로 기존의 시스템을 안정적으로 SOA로 변환하도록 유도함으로써 시간과 비용에 대한 위험성을 줄여주게 되며 SOA에 적합한 유지보수 및 확장성을 높여 줄 것으로 기대된다.

참 고 문 헌

[1] Erl, T., *SOA Principles of Service Design*, Prentice Hall, July 18, 2007.

[2] Arsanjani, A., et al., and Holley, K., "SOMA: A method for developing service-oriented solutions," IBM Systems Journal, Vol.47, No.3, 2008.

[3] Papazoglou, M. and Heuvel, W., "Service-Oriented Design and Development Methodology," *International Journal of Web Engineering and Technology*, InderScience Publisher, Vol.2, No.4, pp.412-442, 2006.

[4] Kim,Y., Doh, k., "The Service Modeling Process Based on Use Case Refactoring," *In Proceedings of the 10th International Conference on Business Information Systems (BIS 2007)*, Lecture Notes in Computer Science 4439, pp. 108-120, 2007.

[5] Kulkarni, N., and Dwivedi, V., "The Role of Service Granularity in A Successful SOA Realization - A Case Study," *In Proceedings of 2008 IEEE Congress on Services (SERVICES'08)*, pp.423-430, July, 2008.

[6] Al Belsushi, W., and Baghdadi, Y., "An Approach to Wrap Legacy Applications into Web Services," *In Proceedings of 2007 International Conference Service Systems and Service Management*, pp.1-6, June, 2007.

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

[8] Xiaofeng Wang, Hu., S.X.K., Haq, E., and, Garton, H., "Integrating Legacy Systems within The Service-oriented Architecture," *IEEE Power Engineering Socienty General Meeting 2007*, pp.1-7, June, 2007.

[9] Lewis, G., Morris, E., Smith., and Simanta, S., "SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment," *CMU/SEI-2008-TN-008*, Software Engineering Institute, May, 2008.

[10] 박옥자, 최시원, 김수동, "레거시 시스템의 웹서비스화를 위한

마이그레이션 기법," *한국정보처리학회 논문지 D*, 제 16-D권, 제4호, pp.583-594, 2009년 8월.

[11] 김유경, "유스케이스 재구성을 통한 서비스 식별," *한국전자거래학회지*, 제12권 제4호, pp.145-163, 2007년 11월.

[12] 윤홍란, 김유경, 박재년, "유스케이스기반 웹서비스 식별 방법," *한국컴퓨터종합학술대회 논문집*, Vol.32, No.1(B), pp.352-354, 2005년 7월.

[13] 김동욱, 국승학, 김현수, "레거시 시스템을 웹 서비스에 통합하기 위한 연구," *한국컴퓨터종합학술대회 논문집*, Vol. 35, No.1 (B), pp.75-80, 2008년 6월.

[14] Blaha, M. and Rumbaugh, J., *Object-Oriented Modeling and Design with UML*, Pearson Prentice Hall, 2005.

[15] Boch, G., Rumbaugh, J., and Jacobson, I., *the United Modeling Language User Guide*, Addison Wesley, 2005.

[16] Glenn E. Krasner and Stephen, T. P., "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80," *Journal of Object-Oriented Programming*, Vol1, Num3, pp.26-49, 1988.

[17] MacKenzie, C., Laskey, K., McCabe, F., Brown, P., and Metz, R. eds., *Reference Model for Service Oriented Architecture 1.0*, OASIS Standard, 12 October, 2006.

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

[19] 양해승, 박병형, 양해술, "레거시시스템의 마이그레이션을 위한 지원도구의설계 및 구현," *한국정보처리학회 논문지 D*, 제 14-D권, 제7호, pp. 763-772, 2007년 12월.

김 지 원

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

2004년 영동대학교 컴퓨터학과(학사)

2008년~현 재 숭실대학교 컴퓨터학과 석사과정

관심분야 : 객체지향 S/W공학, 서비스 지향 아키텍처(SOA), 클라우드 컴퓨팅 (Cloud Computing)

라 현 정

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

2003년 경희대학교 우주과학과(이학사)

2006년 숭실대학교 컴퓨터학과(공학석사)

2006년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 서비스 지향 아키텍처(SOA), 클라우드 컴퓨팅(Cloud Computing), 모바일 서비스(Mobile Service)

김 수 동

e-mail : sdkim777@gmail.com

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 9월~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 서비스 지향 아키텍처(SOA), 클라우드 컴퓨팅(Cloud Computing), 모바일 서비스(Mobile Service), 객체지향 S/W공학, 컴포넌트 기반 개발 (CBD), 소프트웨어 아키텍처