韓國IT서비스學會誌
第5卷 第2號
2006年 8月

3 레벨 관점의 서비스 타입의 분류 및 정의 방법*

최미숙** · 이서정***

3 Leveled Viewed Classification and Definition of Service Type for SOA*

Misook Choi** · Seojeong Lee***

■ Abstract ■

SOA(Service Oriented Architecture) can be a technique to make compassable software from mapping business service to software component. To support effective SOA, it is important that services have to be defined or classified more independently for dynamic and reusable composition.

Several methods have been issued but no ways to defined service granularities, service type or service unit. In this paper we introduce 3 level views, service level, service granularity to reuse effectively. And, we suggest service definition guidelines using them.

Keyword : SOA, Service Definition, Service Type, Reuse Service and Component Development Methods

1. 서 론

1990년대 후반에 들어서면서 기업들이 기업의 생산성 향상 및 고객 서비스를 위해 IT 투자를 확대해 감에 따라 기업 내에서도 다수의 이질적인 시스템 도입을 추진하게 되었다. 그 결과 많은 기업 내에는 호환성이 없는 시스템이 산재해 업무 추진에 어려움을 주었다. 이러한 복잡하고 까다로운 분산화 된 시스템을 통합하고자 업계에서는 많은 노력을 기울이게 되었다. 초창기에 통합은 서로 다른 개발 환경과 사용 환경의 차이에 따라 많은 문제점을 갖고 있어, 통합의 결과가 웹 서비스의 중요한

* 이 논문은 2005학년도 한국해양대학교 신진교수연구비 지원의 결과임.
** 우석대학교 컴퓨터공학과
*** 한국해양대학교 IT공학부

미션인 실시간 정보 서비스가 되지 않는 아이러니를 낳기도 했다. 하지만, 거대한 ERP 시스템을 포기하는 것은 막대한 비용 손실을 초래하게 되므로, 새로운 통합을 시도하게 되었고 그 결과로 등장한 기술이 SOA(Service Oriented Architecture)[10]이며 이를 구현한 기술이 웹 서비스(Web Services)[4]이다.

통합의 해결책으로 대두된 SOA(Service Oriented Architecture)는 비즈니스 수준의 서비스를 효율적으로 제공하기 위해 IT수준의 부품을 개발하고 관리하는 것으로, 기존 애플리케이션의 서비스를 조합함으로써 새로운 애플리케이션 서비스를 구현하는 아키텍처이다[12]. 즉, SOA(Service Oriented Architecture)의 서비스는 보다 비즈니스 프로세스(Business Procee) 관점에서 접근하고 하나의 서비스는 독립성을 갖지만 서비스는 상호 이용할 수 있으며 하나의 비즈니스 서비스는 또한 서비스들의 조합으로 완성된다. E-비즈니스 환경에서 모든 가능한 업무 기능이 서비스로서 정의되어질 수 있다. 비즈니스 프로세스 관점의 서비스는 컴포넌트들의 상호 교류에 의해서 실현되어 진다.

서비스 지향 아키텍처에서 기존에 만들어진 서비스의 조합에 의해서 시스템을 구축할 경우나 동적으로 서비스를 선택하여 시스템을 실행할 경우 가장 중요한 것은 이 기종 분산 환경에서 사용자들의 다양한 서비스의 요구에 대응하여 적합한 서비스를 선택할 수 있도록 서비스의 입자의 크기에 따라 다양한 레벨 관점에서 서비스의 타입이 분류되어야 하고 보다 독립적인 서비스가 식별되어야 한다. 또한, 시스템을 구축하기 위하여 분석 및 설계 산출물을 포함하고 있는 기존의 소프트웨어 아키텍처가 서비스를 고려한 서비스 지향 아키텍처로 확장되어질 수 있는 방법이 존재하지 않는다. 또한, 기존의 서비스 지향 아키텍처 설계를 위한 방법론들 중 서비스를 [6]는 컴포넌트의 인터페이스라고 정의하고 있고 [1, 3, 7, 9, 11]은 유스케이스라고 정의하고 있다. 따라서 현재, SOA에서의 서비스의 단위가 명확하지 않다. 기존의 방법론들은 보다 느슨한 결합도 가진 서비스를 효율적으로 식별할 수 있는 구체적인 방법 및 식별 기준 제시가 미약하고 존재한다 하더라도 기능을 실행하기 위해서 필수적으로 필요한 데이터에 해당하는 엔티티 클래스 간의 관계를 고려하지 않고 유스케이스의 시나리오가 되는 프로세스의 흐름 분석을 기반으로 식별 기준을 정의한다.

따라서 본 논문에서는 기존의 소프트웨어 아키텍처가 3 레벨 관점으로 이루어져 있는 모델 및 산출물을 그대로 이용하여 서비스 지향 아키텍처로 확장되어질 수 있는 서비스 식별 방법을 제안한다. 또한 사용자들의 다양한 서비스의 요구에 대응하여 적합한 서비스를 선택할 수 있도록 서비스의 입자의 크기에 따라 다양한 레벨 관점에서 서비스의 타입을 분류하고 정의한다. 이렇게 분류된 서비스 타입에 의한 서비스를 효율적으로 식별할 수 있도록 엔티티 클래스의 정적관계를 도입한 서비스 식별 방법과 기준을 정의한다.

본 논문의 1장에서는 연구 필요성 및 연구 목적을 제시하고, 2장에서는 기존의 SOA(Service Oriented Architecture)지원 방법론의 문제점을 제시한다. 3장에서는 3 레벨 관점에서 서비스 타입의 분류 및 서비스 식별 방법을 제시한다. 4장에서는 온라인 북 주문 시스템의 사례를 통해서 본 논문에서 제시하는 방법의 실용성 및 효율성을 비교 평가한다. 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

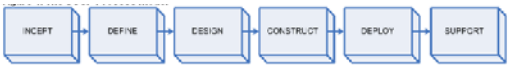
SOA(Service Oriented Architecture)는 기존의 OOAD(Object-Oriented Analysis and Design)나 CBD(Component-Based Development)와 특성이 다르기 때문에 SOA(Service Oriented Architecture)에 적합한 서비스 식별 방법이 필요하다.

따라서 본 절은 SOA(Service Oriented Architecture)를 지원하는 대표적인 방법론으로서 RUP와 XP의 특징만을 모아서 만든 SOUP(Service Oriented Unified Process)[7], IBM에서 정의한 SOAD

(Service Oriented Analysis and Design)[9]와 SOMA (Service-Oriented Modeling and Architecture)[3] 그리고 M4SOD(Service Oriented Development Methodology for SOA)[1]를 제시하고 서비스 식별 방법을 기준으로 분석한다.

2.1 SOUP

SOUP은 Knual Mittal이 제안한 것으로서 RUP와 XP의 특징만을 모아서 SOA에 적용시킨 방법론이다. SOUP은 소프트웨어 개발 단계를 6개의 단계로 나누고 각 단계에서 수행해야 되는 활동들을 정의한다.



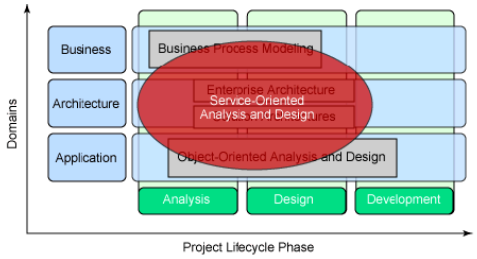
[그림 1] SOUP 개발 프로세스

Define 단계에서 기능적 요구사항과 비 기능적 요구사항을 파악한 후 하향식(Top-Down) 방법으로만 유스케이스 단위의 서비스를 식별한다. 따라서 다양한 레벨 관점의 서비스 타입 분류나 서비스를 식별하기 위한 세부 방법에 대하여 구체적으로 명시하지 않고 있다.

2.2 SOAD

SOAD는 OOAD와 EA frameworks, BPM과 같은 기존의 모델링 개념들을 SOA에 적용시킨 방법론이다[9]. [그림 2]에서는 OOAD, EA, BPM의 주된 적용분야를 보여주고 있으며, 가로축은 프로젝트의 생명주기를 세로축은 각 방법이 적용되는 도메인의 추상화 단계를 나타낸다.

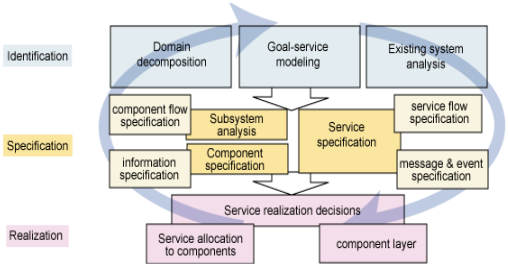
SOA에서 요구하는 사항들을 지원할 수 있도록 상향식(Buttom-Up) 및 하향식(Top-Down)을 모두 포함한 방법론을 제시하고 있다. 그러나 서비스 식별을 위해서 서비스 단위나 서비스 타입 그리고 서비스를 정의하기 위한 세부절차에 대하여 구체적으로 명시하지 않고 있고 또한 SOA를 지원하기 위한 각 단계에 대한 세부적인 방법 및 활동을 명시하지 않고 있다.



[그림 2] SOAD 개발 프로세스

2.3 SOMA

SOMA는 SOA를 설계하고 모델화하고 구현하기 위해 필요한 방법론이다. SOMA는 SOA를 지원하는 모델링, 분석, 설계 기술과 활동을 포함하며 SOA를 7 단계의 레이어로 나누고 레이어 내의 각 요소를 정의하고 아키텍처를 결정하면서 동시에 [그림 3]에서 제시하듯이 서비스의 식별, 명세 그리고 실현의 세단계로 나누어 SOA의 모델을 정의하고 있다.



[그림 3] SOMA 개발 프로세스

SOMA는 기존 자산의 비즈니스 도메인을 중심으로 하향식(Top-Down) 접근법과 레거시 시스템을 중심으로 상향식(Bottom-Up) 접근법을 동시에 통합하여 사용하고 상향식이나 하향식으로 식별되지 않은 다른 서비스를 찾아내는 목표 서비스 모델링(Goal-Service Modeling)으로 구성된다. 상위 수준의 비즈니스 프로세스 기능은 크기가 큰(large-

grained) 서비스로 식별되고 좀 더 작은 단위(smaller-grained) 서비스는 기존에 존재하는 기능을 중심으로 식별한다.

그러나 SOMA는 서비스를 식별하기 위한 세부 방법에 대하여 구체적으로 명시하지 않고 있다.

2.4 M4SOD

M4SOD 개발 방법론에서의 서비스 식별 방법은 유스케이스 중심의 서비스 식별 방법으로 유스케이스의 워크플로우의 흐름을 분석하여 서비스를 식별한다. 이 방법은 프로세스 중심의 제어흐름의 관계만 고려하여 서비스 단위를 식별하고 기능을 수행하기 위해서 포함되는 데이터 간의 관계에 의한 종속성은 포함하지 않고 있다. 유스케이스는 클래스와 클래스 간의 메소드 호출에 의한 상호교류에 의해서 실행된다. 따라서 시스템의 분석 단계에서 유스케이스가 보다 독립적인지 아닌지를 결정하기 위해서는 유스케이스의 워크 플로우 분석에 의한 프로세스 간의 종속적 관계 뿐만 아니라 포함되는 데이터 간의 관계에 의한 종속성을 고려하여야 한다.

유스케이스 기반의 Top-Down 방식으로 비즈니스 레벨의 서비스를 서비스라고 정의하고 있다. 따라서 다양한 레벨 관점에서 입자의 크기에 따른 서비스의 타입을 분류하고 있지 않다.

기존의 서비스 식별 방법은 대다수 유스케이스와 유스케이스의 워크플로우 흐름분석을 통한 프로세스 중심의 제어흐름 간의 종속 관계만 고려하므로 서비스를 식별하기가 용이하지 않고 서비스 식별의 정확도가 떨어질 수 있다.

또한 SOA가 서비스의 조립에 의해서 새로운 시스템을 구축하거나 재사용하는 관점에서 보면 다양한 레벨 관점에서의 서비스의 식별 및 명세가 이루어져야 하는데 대 다수 유스케이스 기반의 비즈니스 레벨의 서비스만을 서비스로 정의하고 있고 그들의 서비스 식별 방법만을 제공하고 있다. 따라서 동적으로 변화하는 사용자의 요구 사항에 대처, 재사용 그리고 소프트웨어의 유지 관리가 효율적이지 못하다.

3. 3 레벨 관점의 서비스 타입의 분류 및 정의

3.1 서비스의 정의

SOA 기술의 특징은 재사용가능(Reusable), 조립가능(Composable), 상호운영적(Interoperability) 그리고 느슨한 결합(Loosely coupled)에 있다. 이를 기반으로 각 서브시스템에서 요구하는 서비스에 맞게 부품을 조립하는 방식으로 구현한다. 기존의 아키텍처에서 코딩에 많이 의존하던 것에 비하면 경제적 개발 방식이 될 수 있다. 이러한 서비스 지향 아키텍처를 구현하기 위해서 가장 중요한 것은 서비스이고 서비스는 애플리케이션의 기능이라 정의할 수 있다. 예를 들면, 시스템에서 제공하는 기능은 비즈니스 관점에서는 유스케이스가 서비스가 되지만 컴포넌트(Component) 관점[8]에서는 컴포넌트가 제공하는 인터페이스가 된다.

SOA를 지원하기 위해서 느슨한 결합도를 가진 서비스의 식별 및 정의 방법이 가장 중요하고 SOA의 서비스는 보다 비즈니스 관점인 유스케이스[5]를 기반으로 정의되어진다. 그러나 SOA의 서비스 단위인 유스케이스는 컴포넌트 인터페이스 및 컴포넌트 인터페이스 간의 상호교류를 통해서 실현이 된다. 따라서, 서비스의 단위를 컴포넌트의 인터페이스라 고정한다면 사용자의 다양한 요구사항을 만족시키는 시스템을 제공하는 것에 한계가 있다.

3.2 서비스의 특성

성공적인 서비스 지향 아키텍처 설계를 위해서 서비스들의 조립에 의해서 새로운 시스템을 효율적으로 구현할 수 있도록 하여야 하므로 서비스는 다음과 같은 특성을 가지고 있어야 한다.

첫째, 서비스는 기능 단위가 되어야 한다.
둘째, 서비스는 액터(Actor)에게 가치 있는 결과를 제공해야 한다.
셋째, 조립에 의한 재사용이 효율적으로 이루어지도록 서비스는 느슨한 결합도를 가져야 한다.
넷째, 액터(Actor) 및 시스템의 다양한 요구 사항을 만족할 수 있도록 다양한 레벨 단위의 서비스가 제공되어야 한다.

3.3 서비스 타입의 분류 및 정의

서비스 지향 아키텍처는 애플리케이션 서비스를 분리하여 필요에 따라 조합함으로써 새로운 애플리케이션 서비스를 제공할 수 있도록 소프트웨어 부품을 관리하는 것이 특징이다. 이를 위해서는 서비스를 분리하는 기준, 서비스를 조합하는 기준 그리고 서비스들 사이에 프로세스가 요구된다. 이에 대한 명세가 완벽할 때에 SOA 개념을 실현한 애플리케이션 서비스를 생산할 수 있다.

기존의 서비스 식별 방법은 대다수 유스케이스와 유스케이스의 워크플로우 흐름분석을 통한 프로세스 중심의 제어흐름 간의 종속 관계만 고려한다. 그러나 유스케이스는 유스케이스의 워크플로우의 흐름 따라 클래스와 클래스 간의 상호교류에 의해서 실행된다. 따라서 시스템의 분석 단계에서 유스케이스가 보다 독립적인지 아닌지를 결정하기 위해서는 유스케이스의 워크 플로우 분석에 의한 프로세스 간의 종속적 관계 뿐만 아니라 포함되는 클래스 간의 관계에 의한 데이터의 종속성 또한 고려하여야 한다.

따라서 본 절에서는 이러한 서비스의 특성을 고려하여 3 레벨 관점에서 서비스의 의미와 입자의 크기를 고려한 서비스 타입을 분류하고 보다 독립적인 서비스를 효율적으로 식별할 수 있는 방법을 제시한다.

3.3.1 타입 1. 애플리케이션 레벨의 서비스

기존의 SOA 설계 방법론들은 유스케이스 또는 컴포넌트 인터페이스만을 서비스로 식별하고 있다. 그러나 본 논문에서는 서브시스템의 단위의 기능을 애플리케이션 레벨의 서비스로 정의한다. 서비스 지향 아키텍처의 특징은 애플리케이션의 서비스를 분리하여 필요에 따라 조합함으로써 새로운 애플리케이션 서비스를 제공할 수 있도록 소프트웨어 부품을 관리하는 것이 특징이다. 따라서 기능 단위가 되는 서비스 관점에서 보면 가장 큰 입자이면서 가장 느슨한 결합도를 가지는 재사용 단위는 서브시스템의 단위라 정의할 수 있다. 따라서 본 논문에서는 서브시스템의 단위를 애플리케이션 레벨의 서비스라 정의한다.

애플리케이션 레벨의 서비스는 유스케이스(Use Case)[5] 단위의 서비스들의 조합, 즉 비즈니스 레벨의 서비스들의 조합에 의해서 이루어지는 서비스로서 서브시스템 단위가 된다. 애플리케이션 레벨의 서비스는 비즈니스 레벨의 서비스나 비즈니스 레벨의 서비스들 간의 상호교류에 의해서 이루어지는 서비스이다.

서비스 정의 방법은 기능 분할을 통해서 전체 업무를 적절한 단위 업무로 분할한 후에 단위 업무를 애플리케이션 레벨의 서비스로 정의한다. 애플리케이션 레벨의 서비스는 산출된 유스케이스 다이어그램에 의해서 다음과 같은 기준을 고려하여 정의한다.

기준 1 : 같은 액터(Actor)[5]와 상호 작용한다.
기준 2 : 유스케이스가 서로 포함관계나 확장 관계를 갖는다.
기준 3 : 공통의 데이터를 사용한다.
기준 4 : 유사한 업무를 실행한다.

3.3.2 타입 2. 비즈니스 레벨의 서비스

본 논문에서 정의하고 있는 비즈니스 레벨의 서비스는 유스케이스를 기반으로 한 서비스를 일컫는다. 비즈니스 레벨의 서비스는 애플리케이션 레벨의 서비스를 실현하기 위한 부품이 되는 서비스 단위이다. 따라서 비즈니스 레벨의 서비스 입자의

크기는 애플리케이션 레벨의 서비스 보다는 작다. 애플리케이션 레벨에 포함된 서비스는 기능이 유사한 유스케이스들을 포함하고 있고 각각의 유스케이스는 비즈니스 레벨의 서비스로 매핑 된다. 따라서 애플리케이션 레벨의 서비스와 비즈니스 레벨의 서비스 간의 추적성이 유지된다. 애플리케이션 레벨의 서비스를 재사용할 경우 애플리케이션 레벨의 서비스가 포함하고 있는 비즈니스 레벨의 서비스 그리고 비즈니스 레벨의 서비스와 매핑되고 있는 컴포넌트 레벨의 서비스가 추적성에 의하여 자연스럽게 포함되어지므로 서비스의 재사용 및 유지 보수 그리고 서비스의 관리가 용이하다.

또한 비즈니스 레벨의 서비스는 컴포넌트의 인터페이스나 컴포넌트 인터페이스들 간의 상호교류 그리고 비즈니스 레벨의 서비스 간의 상호교류에 의하여 실현된다. 기존의 SOA 설계 방법들 중 대다수는 유스케이스를 기반으로 서비스를 정의하고 있지만 유스케이스 기반의 서비스를 어떻게 식별해야 보다 독립적인 서비스를 효율적으로 식별하는지에 대한 명확한 기준과 방법이 명시되어 있지 않다. 따라서 본 절에서는 유스케이스 기반의 비즈니스 레벨의 서비스를 효율적으로 식별하기 위한 기준과 방법을 제시한다.

비즈니스 레벨의 서비스는 유스케이스 단위의 서비스이다. 서비스는 액터(Actor)에게 가치 있는 결과를 제공하는가를 기준으로 분석해야 한다. 따라서, 액터(Actor)에 의해서 일어나는 이벤트는 비즈니스 레벨의 서비스이다. 유스케이스 기반의 비즈니스 레벨의 서비스는 유스케이스를 기반으로 결합도가 낮은 보다 독립적인 서비스를 식별하여야 한다. 또한 본 논문에서는 비즈니스 레벨의 서비스를 원자 서비스, 원자 서비스의 조합에 의한 복합 서비스 그리고 공유 서비스로 분류한다.

비즈니스 레벨의 서비스는 유스케이스 모델을 참조하여 분리되어질 수 없고 독립적으로 수행이 가능한 원자 서비스를 먼저 정의한 다음, 원자 서비스들의 조합에 의해서 이루어지는 복합 서비스를 정의한다. 또한, 여러 개의 유스케이스, 즉, 원자 서비스나 복합 서비스가 서로 공유하는 서비스를 공유 서비스로 분류한다.

① 비즈니스 레벨 서비스 1. 원자 서비스

유스케이스가 하나의 원자 서비스로 식별되는 경우이다. 원자 서비스를 식별할 경우 유스케이스의 프로세스 흐름에 의한 종속성뿐만 아니라 클래스 간의 구조적 관계에 의한 정적 종속성도 고려하여 원자 서비스를 식별 한다. 원자 서비스를 식별하는 기준은 다음과 같다.

기준 1 : 유스케이스가 단일 엔티티 클래스의 데이터를 이용하는 경우 원자 서비스로 식별한다.

기준 2 : 유스케이스의 실행을 위해서 단일 엔티티 클래스가 아니라 여러 개의 엔티티 클래스를 참조할 경우 엔티티 클래스간의 정적관계가 상속관계(Inheritance Relationship)나 부품관계(Aggregation Relationship)이고 유스케이스가 다른 유스케이스에 의존되지 않고 독립적으로 실행될 경우 원자 서비스로 식별한다.

기준 3 : 유스케이스로 식별되지 않았다 하여도 단일 엔티티 클래스를 사용하는 액터 관점의 기능이라면 원자 서비스로 식별 한다.

② 비즈니스 레벨 서비스 2. 복합 서비스

유스케이스가 하나의 복합 서비스로 식별되는 경우이다. 원자 서비스를 식별하는 기준은 다음과 같다.

기준 1 : 하나의 유스케이스가 여러 개의 원자 서비스들의 조합에 의해서 실행될 때 복합 서비스로 정의한다.

③ 비즈니스 레벨 서비스 3. 공유 서비스

공유 서비스를 식별하는 기준은 다음과 같다.

기준 1 : 여러 개의 유스케이스가 같은 프로세스의 흐름 단위를 사용할 경우 그 프로세스 흐름 단위에 속하는 유스케이스를 공유 서비스로 식별한다.

기준 2 : 여러 개의 원자 서비스나 복합 서비스에서 임의의 서비스를 공유할 경우 그 서비스를 공유 서비스로 식별한다.

3.3.3 타입 3. 컴포넌트 레벨의 서비스

컴포넌트 레벨의 서비스를 구현 관점의 서비스 레벨로 볼 수 있다. 컴포넌트 레벨의 서비스를 식별하는 기준은 다음과 같다.

기준 1. 컴포넌트의 인터페이스 메소드를 컴포넌트 레벨의 서비스라 정의한다.

4. 사례 연구 및 분석 평가

4.1 사례 연구

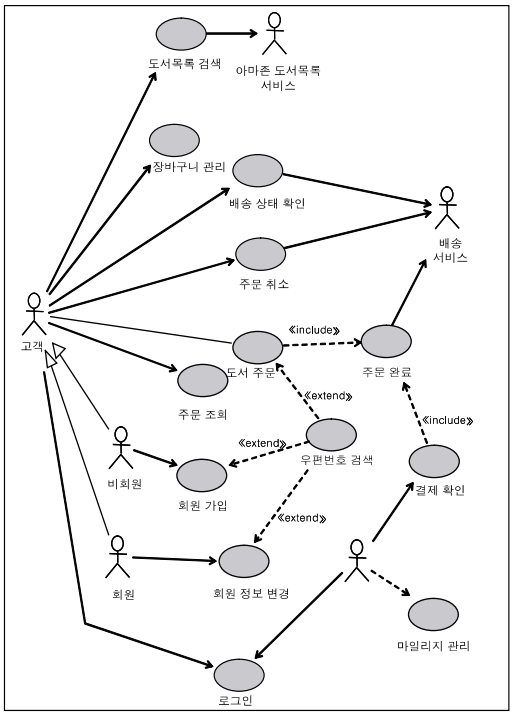
본 절에서는 온라인 북 주문 시스템[2]을 통해서 각 레벨 당 서비스 타입에 의한 서비스를 식별하여 정의한 사례를 제시한다.

4.1.1 온라인 북 주문 시스템의 업무 개요

고객은 인터넷을 사용하여 온라인 서점에서 도서 목록을 검색하고 원하는 도서를 장바구니에 담아 주문하는 시스템이다. 도서 주문을 원할 때 회원인 경우는 시스템에서 고객의 기본 정보를 제공하며 배송 주소의 변경을 원하는 경우 배송지를 수정하여 입력할 수 있고, 비회원인 경우는 고객 기본 정보와 배송 정보를 입력해야 한다. 기본 정보와 배송 정보의 입력이 완료되면 결제 방법을 선택하여 대금 결제를 한다. 대금 결제의 방법에는 신용카드, 마일리지, 무통장 입금, 인터넷 계좌이체 등의 방법이 있다. 입금이 완료되면 배송업체에 배송을 요청한다. 무통장 입금인 경우에는 관리자가 입금을 확인 후 배송업체에 요청한다. 주문 취소를 원하는 고객은 주문 상태를 확인하고 주문을 취소할 수 있다. 이와 함께 도서를 주문한 고객에게 현재의 배송 상태를 추적하는 배송 추적 서비스를 제공 한다.

4.1.2 온라인 북 주문 시스템의 유스케이스 다이어그램

다음 [그림 4]는 온라인 북 주문 시스템의 유스케이스 다이어그램이다.



[그림 4] 온라인 북 주문 시스템의 유스케이스 다이어그램

4.1.3 온라인 북 주문 시스템의 액터(Actor) 목록

다음 <표 1>은 온라인 북 주문 시스템의 액터의 목록과 정보를 제시한다.

4.1.4 온라인 북 주문 시스템의 유스케이스 목록

다음 <표 2>는 온라인 북 주문 시스템의 유스케이스 목록을 제시한다.

〈표 1〉 온라인 북 주문 시스템의 액터 목록

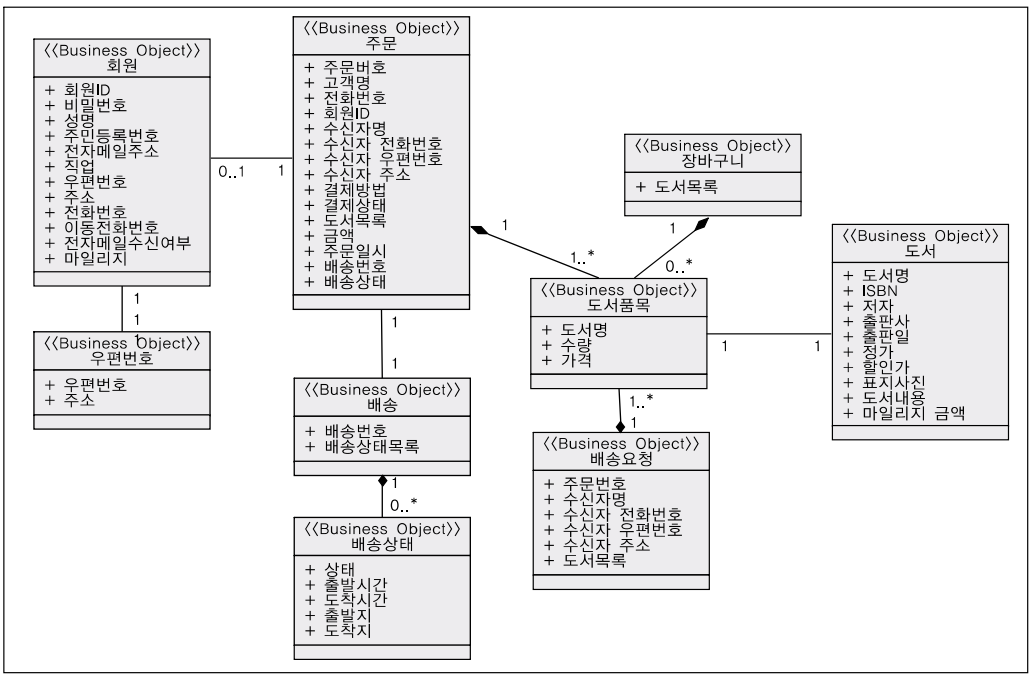
액터(Actor)	설 명
고객	온라인 서점에서 도서를 주문하고 배송 상태 확인 서비스를 제공 받고자 하는 사용자로 회원과 비회원으로 구분할 수 있다.
회원	온라인 서점에 등록된 고객으로 도서를 구입할 때 마일리지가 적립되며 도서 구입에 적립된 마일리지를 사용할 수 있다.
비회원	온라인 서점에 등록되지 않은 고객
관리자	고객정보, 도서정보를 관리하고 배송업체에 배송 요청 및 취소 업무를 수행하는 시스템 사용자
아마존 도서 시스템	시스템에 도서목록 검색을 제공하는 타 시스템
배송 시스템	배송 요청, 취소 그리고 조회 기능을 제공하는 타 시스템

〈표 2〉 온라인 북 주문 시스템의 유스케이스 목록

ID	이름	설 명
U001	도서 목록 검색	고객이 원하는 조건으로 검색하여 도서 목록을 보여준다.
U002	장바구니 관리	온라인 서점에서 도서 목록 검색 중 원하는 도서를 장바구니에 담아둔다. 고객은 장바구니에 담기, 조회, 주문 수량 변경, 삭제 등의 기능을 사용할 수 있다.
U003	도서 주문	장바구니에서 구매를 원하는 도서를 선택하여 주문한다. 회원은 가입 시 입력한 정보를 기준으로 주문을 하게 도지만 주문 시 임시로 배송주소를 변경할 수 있다. 비 회원은 주문, 배송을 위한 정보를 입력해야 한다.
U004	주문 조회	고객이 주문한 현황을 목록으로 보여준다. 조회되는 정보는 주문번호, 주문 일시, 도서명, 수량, 금액, 주문 처리 상태 등이다. 회원인 경우 로그인 시에 입력한 고객 ID와 암호로 조회가 가능하지만 비회원은 이름과 주민등록번호를 입력해야 조회가 가능하다.
U005	주문 취소	고객은 주문 현황을 조회하고 주문 내역을 취소할 수 있다. 주문의 취소는 배송이 시작되기 전까지만 가능하다.
U006	결제 확인	자동으로 입금 확인이 되지 않는 입금 방법에 대해 관리자는 결제를 확인한다.
U007	배송상태 확인	처리된 주문에 대해 현재 배송 상태를 조회한다.
U008	마일리지 관리	관리자는 도서에 따라 마일리지를 부여한다.
U009	회원 가입	비회원인 고객이 개인 정보를 입력하여 회원으로 가입하는 기능을 가진다.
U010	회원 정보 변경	회원인 고객이 개인 정보 조회 및 수정 또는 회원 탈퇴의 기능을 가진다.
U011	로그인	회원은 자신의 회원 ID와 암호를 입력하여 로그인하며, 비회원과 관리자는 각각 방문자와 관리자로 로그인 한다.
U012	주문 완료	도서 주문 또는 결제 확인된 주문 사항에 대하여 주문 완료 처리를 한다. 주문 완료 결과를 고객에게 통지하고 배송업체의 배송 서비스를 통해 배송을 요청한다.
U013	우편번호 검색	도서주문, 회원가입, 회원정보 변경 등 주소 입력 시 우편번호를 검색할 수 있다.

4.1.5 온라인 북 주문 시스템의 비즈니스 객체 모델

다음 [그림 5]는 유스케이스의 실현을 위하여 필요한 비즈니스 객체를 도출하고 비즈니스 객체 사이의 관계를 정의한 비즈니스 객체 모델이다. 유스케이스 실현화 과정을 통해서 산출된 비즈니스 객체는 클래스 다이어그램의 엔티티 클래스가 된다. 따라서 비즈니스 레벨의 결합도가 낮은 보다 독립적인 서비스를 식별하기 위해서 유스케이스와 비즈니스 객체 모델에 포함된 비즈니스 객체와의 관계 분석을 통하여 비즈니스 레벨의 서비스가 식별된다.



[그림 5] 온라인 북 주문 시스템의 비즈니스 객체 모델

〈표 3〉 온라인 북 주문 시스템의 비즈니스 객체의 설명

비즈니스 객체	설 명
회원	회원의 이름과 주민등록번호, 전자메일주소, 직업, 우편번호, 주소, 전화번호, 이동전화번호, 전자메일을 수신할지의 여부, 확보한 마일리지 등의 정보를 표현한다.
도서	고객이 검색 또는 주문한 도서의 정보를 표현한다. 여기에는 도서명, ISBN, 저자, 출판사, 정가, 할인가, 표지사진, 도서내용, 구입 시 고객이 받게 되는 마일리지 금액 등의 정보가 표현된다.
주문	주문 고객명과 전화번호 및 수신자의 이름과 전화번호, 우편번호, 주소가 포함되며, 결제방법, 결제완료 여부를 나타내는 상태, 배송확인을 위한 배송 번호 등의 주문 관련 정보도 포함된다. 또한 고객이 주문한 도서목록과 이들 도서의 총합계 금액, 주문일시 등도 포함된다.
배송요청	고객이 배송 요청을 한 정보를 표현한다. 여기에는 주문 번호와 함께, 이름과 전화번호, 우편번호, 주소 그리고 고객이 주문한 도서목록이 포함된다.
장바구니	고객이 선택하여 장바구니에 담은 도서의 목록이 포함된다.
도서품목	고객이 주문 또는 배송 요청을 한 도서 품목에 대한 정보를 포함한다. 여기에는 도서명, 수량, 가격 등이 포함된다. 주문 비즈니스 객체와 배송 요청 비즈니스 객체는 최소한 하나 이상의 도서 품목 비즈니스 객체를 포함한다. 장바구니 객체는 도서 품목 비즈니스 객체를 포함하지 않을 수도 있다.
배송	고객이 주문한 도서의 주문에서부터 현재까지의 배송 현황을 각 단계별로 표현하는 배송 상태 목록을 관리한다.
우편번호	우편번호와 주소 정보를 표현한다.

위의 <표 3>은 온라인 북 주문 시스템의 비즈니스 객체에 대한 설명이다.

4.1.6 애플리케이션 레벨의 서비스

다음 <표 4>는 본 논문에서 제시한 방법에 의해

서 정의된 온라인 북 주문 시스템의 애플리케이션 레벨의 서비스를 제시한다. 애플리케이션 레벨의 서비스는 본 논문의 3.3절에서 제시한 기준에 의하여 식별된다. 즉, 액터, 유스케이스 그리고 액터와 유스케이스 간의 관계, 유사한 기능의 그룹 및 공통된 데이터의 사용에 의하여 정의된다. 애플리케이션 레벨의 서비스 정의 단계에서는 비즈니스 레벨의 서비스가 상세히 식별되지 않은 단계이므로 유스케이스 다이어그램의 유스케이스를 비즈니스 레벨의 서비스로 간주한다.

〈표 4〉 애플리케이션 레벨의 서비스

애플리케이션 서비스	포함된 비즈니스 서비스	유스 케이스	액터
주문 관리(SC1)	도서 목록 검색	U001	관리자, 고객, 배송 시스템, 아마존도서 시스템
	장바구니 관리	U002	
	도서 주문	U003	
	주문 조회	U004	
	주문 취소	U005	
	주문 완료	U012	
회원 관리(SC2)	마일리지 관리	U008	비회원, 회원, 관리자
	결제확인	U006	
	회원 가입	U009	
	회원 정보 변경	U010	
	우편번호 검색	U013	
	로그인	U011	
배송 확인(SC3)	배송 상태 확인	U007	고객, 배송 시스템

4.1.7 비즈니스 레벨의 서비스

비즈니스 레벨의 서비스는 크게 원자 서비스, 복합 서비스 그리고 공유 서비스로 분류된다. 다음은 본 논문의 3.3절에서 제시한 비즈니스 레벨의 서비스 식별 방법에 의해서 정의된 온라인 북 주문 시스템의 비즈니스 레벨의 서비스를 제시한다.

다음 <표 5>는 비즈니스 객체를 명시한 식별자 테이블이다. 이 비즈니스 객체는 원자적 서비스를 식별할 때 사용되어 진다.

〈표 5〉 비즈니스 객체 식별자

비즈니스 객체	식별자
회원	BO1
주문	BO2
도서품목	BO3
도서정보	BO4
장바구니	BO5
배송요청	BO6
배송	BO7
배송상태	BO8
우편번호	BO9

다음 <표 6>은 식별된 비즈니스 레벨의 서비스 목록과 포함된 비즈니스 객체를 제시한다.

<표 6>에서 제시하고 있듯이 원자 서비스는 단일 엔티티 클래스의 데이터나 2개 이상의 엔티티 클래스의 데이터를 사용하는 경우를 고려하여 식별한다. 만약, 두 개 이상의 클래스 간에 강한 종속 관계를 가지는 사용하는 경우의 유스케이스를 본 논문에서는 원자 서비스로 식별하여 정의하므로 식별하기가 용이하다. 위의 <표 6>에서 제시하고 있는 원자 서비스의 목록 중 두 개 이상의 엔티티 클래스를 포함하고 있는 원자 서비스의 경우를 보면 엔티티 클래스 간의 관계가 부품관계(Aggregation Relationship)를 가지고 있다는 것을 확인할 수 있다. 즉, 서비스를 실행하기 위해서는 데이터를 사용하기 때문에 서비스 식별을 위해서는 엔티티 클래스 간의 연관 관계의 특성을 반드시 고려하여야 한다. 따라서 데이터의 연관관계를 적용하므로 해서 식별된 원자 서비스는 보다 독립적이다. 또한, 본 논문에서 원자 서비스를 식별하기 위해서 단일 엔티티 클래스의 특성을 고려하므로 유스케이스보다는 원자 서비스가 더 상세히 식별된다는 것을 알 수 있다.

복합 서비스인 경우 여러 개의 원자 서비스들의 조합에 의해서 실행되어지는 유스케이스를 복합 서비스로 식별하므로 원자 서비스가 식별되어지면 원자 서비스의 조합에 의해서 실행되어지는 복합 서비스 또한 쉽게 식별되어 진다.

〈표 6〉 비즈니스 레벨의 서비스

비즈니스 서비스 타입	비즈니스 서비스(비즈니스 객체) : 유스케이스	포함된 서비스
원자 서비스(A)	A1 : 도서 목록 검색(BO4) : U001	
	A2 : 도서 품목 조회(BO3) : U004	
	A3 : 주문 취소(BO2, BO3) : U005	
	A4 : 결제 확인(BO2) : U006, U003	
	A5 : 배송상태 확인(BO7, BO8) : U007	
	A6 : 로그인(BO1) : U0011	
	A7 : 장바구니 조회(BO5, BO3) : U002	
	A8 : 장바구니 수정(BO5, BO3) : U002	
	A9 : 장바구니 취소(BO5, BO3) : U002	
	A10 : 장바구니 담기(BO5, BO3) : U002	
	A11 : 회원 가입(BO1, BO9) : U009	
	A12 : 회원 정보 조회(BO1)	
	A13 : 회원 정보 변경(BO1) : U010	
	A14 : 회원 탈퇴(BO1)	
	A15 : 고객 정보 등록(BO2)	
	A16 : 배송 정보 등록(BO2, BO9) : U003	
	A17 : 결제 정보 등록(BO2) : U003	
	A18 : 결제 처리(BO2) : U003	
	A20 : 주문 완료(BO2, BO3, BO6) : U003	
	A21 : 마일리지 부여(BO1) : U003, U008	
	A22 : 마일리지 조회(BO1) : U008	
	A23 : 우편번호 검색(BO9) : U009, U010, U003	
	A24 : 주문 저장(BO2, BO3) : U003	
	A25 : 주문 수정(BO2, BO3)	
복합 서비스(BC)	도서 주문(BC1)	A1 : 도서 목록 검색
		A7, A8, A9, A10 : 장바구니 관리
		A17 : 결제 정보 등록
		A18 : 결제 처리
		A4 : 결제 확인
		A20 : 주문 완료
		A12 : 회원 정보 조회
		A15 : 고객 정보 등록
		A16 : 배송정보 등록
		A21 : 마일리지 부여
		A23 : 우편번호 검색
	주문 조회(BC2)	A6 : 로그인
		A2 : 도서 품목 조회
		A5 : 배송상태 확인
		A16 : 배송 정보 등록
	장바구니 관리(BC3)	A7 : 장바구니 조회
		A8 : 장바구니 수정
		A9 : 장바구니 취소
		A10 : 장바구니 담기
	마일리지 관리(BC4)	A21 : 마일리지 부여
		A22 : 마일리지 조회
공유 서비스(SS)	우편 번호 검색(SS1)	A11 : 회원 가입
		A13 : 회원 정보 변경
		A16 : 배송 정보 등록

위의 <표 6>에서 제시하고 있듯이 공유 서비스인 경우 여러 개의 원자 서비스에 의해서 공유되어지는 것을 볼 수 있다.

따라서 명확한 원자 서비스 식별 기준에 의하여 원자 서비스를 식별한 후 복합서비스를 식별하고 복합 서비스를 식별한 후 공유 서비스를 식별하면 개발자가 비즈니스 레벨의 서비스를 효율적으로 식별할 수 있다.

4.1.8 컴포넌트 레벨의 서비스

컴포넌트 레벨의 서비스는 비즈니스 레벨의 서비스를 구현하는 구현 관점의 서비스 레벨로 볼 수 있다. 컴포넌트 레벨의 서비스는 컴포넌트의 인터페이스 메소드를 컴포넌트 레벨의 서비스라 정의한다. 컴포넌트는 유스케이스의 기능에 해당하는 애플리케이션 컴포넌트와 유스케이스의 기능을 실현하기 위해서 존재하는 비즈니스 컴포넌트가 존재한다. 애플리케이션 컴포넌트의 기능은 비즈니스 컴포넌트나 비즈니스 컴포넌트 간의 상호교류에 의해서 실현되어진다.

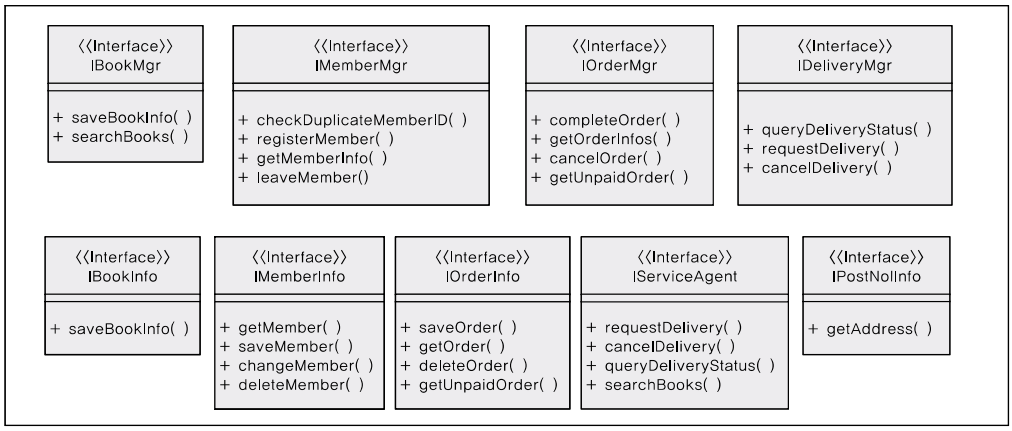
따라서 비즈니스 레벨의 서비스는 애플리케이션의 컴포넌트의 인터페이스 메소드나 비즈니스 컴포넌트의 인터페이스로 매핑될 수 있다. 다음 [그림 6]은 온라인 북 주문 시스템의 컴포넌트 레벨의 서비스를 제시한다.

4.2 분석 평가 및 기대 효과

4.2.1 식별된 서비스의 독립성

유스케이스 실행은 유스케이스가 포함하고 있는 클래스와 클래스들 간의 메소드 호출에 의한 상호교류에 의하여 이루어진다. 따라서 유스케이스 기반의 서비스가 보다 독립적인지 아닌지, 즉, 서비스가 느슨한 결합도를 가지고 있는지를 평가하기 위해서는 클래스 들 간의 구조적 관계 그리고 메소드 호출에 의한 동적 관계를 다 고려하여야 한다. 그러나 SOA에서 구현 관점이 아닌 비즈니스 관점의 서비스를 식별하기 위해서는 분석 초기 단계에서 이루어지므로 클래스 간의 메소드 호출에 의한 동적 관계는 고려되어질 수 없다. 따라서 유스케이스 기반의 보다 독립적인 서비스를 식별하기 위해서는 유스케이스 실행을 위한 유스케이스 워크플로우와 포함된 비즈니스 객체, 즉 엔티티 클래스 간의 구조적 관계를 참조하여 유스케이스 기반의 서비스가 보다 독립적인지를 평가하여야 한다.

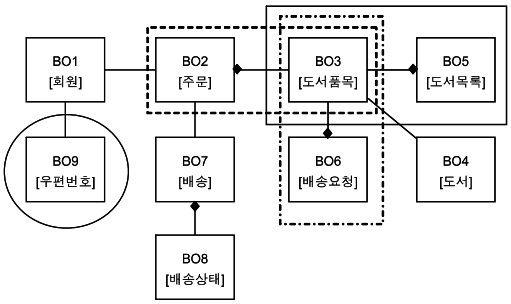
클래스 간의 구조적 관계는 연관관계, 상속관계 그리고 부품(집합)관계가 존재한다. 이때 클래스 간의 관계가 상속관계와 부품관계라면 강한 종속관계를 가지고 있다고 볼 수 있으므로 유스케이스가 두 개 이상의 클래스를 참조한다 하여도 두 개 이상의 클래스 간의 관계가 강한 종속관계를 가진



[그림 6] 온라인 북 주문 시스템의 컴포넌트 레벨의 서비스

다면 이를 원자적 서비스로 식별한다. 따라서 이렇게 식별된 유스케이스 기반의 서비스는 당연히 데이터의 입장에서는 독립성을 만족한다고 볼 수 있기 때문에 식별된 원자 서비스는 느슨한 결합도를 가지고 있다.

또한, 두 개 이상의 클래스를 참조 한다하여도 공유 유스케이스를 포함하고 있는 유스케이스는 원자적 서비스로 식별한다. 유스케이스의 워크플로우 관점에서 공유 유스케이스는 항상 포함해야할 유스케이스이기 때문이다.



[그림 7] 온라인 북 주문 시스템의 비즈니스 객체들 간의 구조적 관계 다이어그램

〈표 7〉 비즈니스 레벨의 원자 서비스와 비즈니스 객체와의 관계

비즈니스 레벨의 원자 서비스	BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9
A1 : 도서 목록 검색				●					
A2 : 도서 품목 조회			●						
A3 : 주문 취소		●	●						
A4 : 결제 확인		●							
A5 : 배송상태 확인							●	●	
A6 : 로그인	●								
A7 : 장바구니 조회			●		●				
A8 : 장바구니 수정			●		●				
A9 : 장바구니 취소			●		●				
A10 : 장바구니 담기			●		●				
A11 : 회원 가입	●								●
A12 : 회원 정보 조회	●								
A13 : 회원 정보 변경	●								
A14 : 회원 탈퇴	●								
A15 : 고객 정보 등록		●							
A16 : 배송 정보 등록		●							●
A17 : 결제 정보 등록		●							
A18 : 결제 처리		●							
A20 : 주문 완료		●	●			●			
A21 : 마일리지 부여	●								
A22 : 마일리지 조회	●								
A23 : 우편번호 검색									●
A24 : 주문 저장		●	●						

즉, 유스케이스가 단일 엔티티 클래스를 사용한다면 가장 독립적이라고 볼 수 있다. 그 이유는 다른 클래스에게 영향을 미치지 않기 때문이다. 그러나 두 개 이상의 엔티티 클래스를 사용하는 유스케이스 존재할 경우 두 개 이상의 엔티티 클래스가 강한 종속 관계를 가지고 있을 경우 유스케이스를 분리하여 서비스로 식별하면 두 개의 엔티티 클래스 관계에 의해서 각각 분리된 서비스는 강한 영향을 미치기 때문에 그들 간의 서비스는 강한 종속 관계를 가지므로 보다 독립적으로 식별되었다고 볼 수 없다.

따라서, 위의 [그림 7]과 <표 7>을 보면 두 개 이상의 엔티티 클래스 즉, 비즈니스 객체를 포함하고 그들의 관계가 부품관계(집합관계)에 있는 유스케이스는 모두 원자 서비스로 식별되어짐을 확인할 수 있다. 또한 단일 엔티티 클래스를 사용하는 유스케이스는 모두 원자 서비스로 식별 되어짐을 확인할 수 있다. 또한, 유스케이스가 공유 유스케이스를 참조하고 있다면 공유 유스케이스는 항상 포함되어져야할 유스케이스이기 때문에 원자 서비스로 식별되어 졌음을 확인할 수 있다. 보다 독립적인 원자 서비스들의 조합에 의해서 복합 서비스가 식별된다면 복합 서비스 역시 독립성을 만족한다고 볼 수 있다.

이러한 이유로 인하여 본 논문에서 식별된 비즈니스 관점의 서비스는 느슨한 결합도 즉, 독립성을 만족하는 서비스로 식별됨을 확인할 수 있다.

4.2.2 서비스 간의 관계에 의한 추적성

온라인 북 주문 시스템의 식별된 서비스 그리고 유스케이스 간의 관계를 통하여 애플리케이션 레벨의 서비스와 비즈니스 레벨의 서비스, 비즈니스 레벨의 서비스와 컴포넌트 레벨의 서비스 그리고 식별된 서비스와 유스케이스 간의 매핑 관계를 통하여 산출물둘 간에 추적(trace)이 가능하다는 다음 <표 8>을 통하여 알 수 있다. 따라서 애플리케이션 레벨로부터 컴포넌트 레벨까지의 서비스의 추적이 용이하므로 서비스의 재사용 및 관리가 효율적이다.

〈표 8〉 식별된 서비스와 유스케이스, 서비스들 간의 관계에 의한 추적성

애플리케이션 레벨의 서비스	비즈니스 레벨의 서비스	컴포넌트 레벨의 서비스	유스케이스
주문관리(SC1)	A1 : 도서 목록 검색	SearchBooks	U001
주문관리(SC1)	A2 : 도서 품목 조회	SerchItems	U004
주문관리(SC1)	A3 : 주문 취소	CancelOrder	U005
주문관리(SC1)	A4 : 결제 확인	requestDelivery	U006, U003
배송확인(SC3)	A5 : 배송상태 확인	queryDeliveryStatus	U007
회원관리(SC2)	A6 : 로그인	getMember	U0011
주문관리(SC1)	A7 : 장바구니 조회	getShoppingCart	U002
주문관리(SC1)	A8 : 장바구니 수정	updateShoppingCart	U002
주문관리(SC1)	A9 : 장바구니 취소	cancelShoppingCart	U002
주문관리(SC1)	A10 : 장바구니 담기	saveShoopingCart	U002
회원관리(SC2)	A11 : 회원 가입	registerMember	U009
회원관리(SC2)	A12 : 회원 정보 조회	getMemberInfo	U010
회원관리(SC2)	A13 : 회원 정보 변경	changeMember	U010
회원관리(SC2)	A14 : 회원 탈퇴	leaveMember	U003
주문관리(SC1)	A15 : 고객 정보 등록	saveCustomerInfo	
주문관리(SC1)	A16 : 배송 정보 등록	saveDeliveryInfo	
주문관리(SC1)	A17 : 결제 정보 등록	saveApproval	U003
주문관리(SC1)	A18 : 결제 처리	requestApproval	U003
주문관리(SC1)	A20 : 주문 완료	completeOrder	U003
회원관리(SC2)	A21 : 마일리지 부여	saveMileage	U003, U008
회원관리(SC2)	A22 : 마일리지 조회	getMileage	U008
공유 서비스	A23 : 우편번호 검색	getAddress	U009, U010, U003
주문관리(SC1)	A24 : 주문 저장	saveOrder	U003
....	.....	....	....

4.2.3 기대 효과

본 논문에서 제시한 방법의 기대효과는 다음과 같다.

첫째, 유스케이스의 워크플로우만이 아니라 유스케이스가 포함하고 있는 엔티티 클래스 간의 구조적 관계를 고려하여 서비스를 식별하므로 보다 독립적인 서비스를 용이하게 식별할 수 있고 서비스 타입, 식별 방법 및 기준을 구체적으로 제시했기 때문에 개발자의 노력과 시간을 줄일 수 있다.

둘째, 다양한 레벨 단위의 관점에서 서비스 입자의 크기를 고려하여 서비스의 타입을 분류하여 정의하기 때문에 기존에 만들어진 서비스의 조합에 의해서 시스템을 구축할 경우나 동적으로 서비스를 선택하여 시스템을 실행할 경우, 사용자들의 다양한 서비스의 요구에 대응하여 적합한 서비스를 용이하게 선택할 수 있다.

셋째, 비즈니스 서비스를 정의할 경우 유스케이스를 분석하여 원자 서비스를 정의한 이후에 복합 서비스를 정의하기 때문에 정의된 서비스는 보다 독립적으로 용이하게 식별되어 진다.

넷째, 애플리케이션 레벨로부터 컴포넌트 레벨까지의 서비스의 추적이 용이하여 서비스의 재사용 및 관리가 효율적이다.

다섯째, 상향식 방법과 하향식 방법을 동시에 사용하여 서비스를 레벨 단위로 정의한 후 매핑하므로 비즈니스 관점과 구현 관점의 서비스들의 의미적인 갭 차이를 줄일 수 있다.

5. 결론 및 향후과제

기존의 SOA(Service Oriented Architecture)를 지원하고 있는 방법론의 경우 서비스 단위나 서비스를 정의하기 위한 세부 방법에 대하여 구체적으로 명시하지 않고 있으며 레벨 단위의 입자의 크기 및 서비스의 의미적 특성을 고려한 서비스의 타입을 상세히 분류하고 있지 않고 있다. 서비스를 식별할 경우도 유스케이스 기반의 워크플로우만 참조하여 식별한다.

본 논문에서는 동적인 사용자의 요구사항을 만족시키고 서비스의 재사용이 효율적으로 이루어질 수 있도록 3 레벨 관점의 레벨 단위로 서비스 타입을 분류하고 서비스 입자의 크기를 고려한 서비스 단위나 유스케이스의 워크플로우 뿐만 아니라 참조하는 엔티티 클래스 의 구조적 관계를 고려한 서비스 정의 방법과 기준을 제시하였고 온 라인 북 주문 시스템의 사례제시를 통하여 그 실용성을 검증하고 기대효과를 제시 하였다. 본 논문에서 제시한 서비스 식별 방법은 보다 독립적인 서비스의 식별 용이성, 재사용성 그리고 추적성에 의한 서비스의 관리의 용이성의 효과를 부여한다.

향후 연구과제로는 서비스 식별 방법의 프로세스, 서비스 정보 저장소를 모델링하기 위한 명세 방법을 진행할 예정이다.

참 고 문 헌

[1] 윤홍란, “M4SOD : SOA를 위한 서비스 지향 개발 방법론”, 박사학위 논문, 숙명여대, 2005.

[2] 전병선, 「객체지향 CBD 개발 방법론 : J2EE Enterprise System」, 영진출판사, 2004.

[3] Ali Arsanjani, “Service-Oriented Modeling and Architecture(SOMA)”, IBM developer Works, 2004.

[4] Gartner group, “SOA/Web Services”, 2005.

[5] Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[6] John, Cheesman and John Daniels, *UML Components : A Simple Process for Specifying Component-Based Software*, Addison-Wesley, 2001.

[7] Knual Mittal, “Service Oriented Unified Process(SOUP)”, 2006, http://www.knual-mittal.com/html/soup.shtml

[8] Misook Choi, Eunsook Cho, “Component

Identification Methods applying Method Call Types between Classes", *Journal of Information and Science Engineering*, 2006.

[9] Olaf Zimmermann, Pal Krogdahl, Clive Gee, "Elements of Service-Oriented Analysis and Design", IBM, 2006.

[10] Roy, W. Schulte, Yefim V. Natis, "Service Oriented Architecture", Gartner Group, SSA Research Note SPA-401-068, 1996.

[11] Samir Nigam, "Service Oriented Development of Applications(SODA) in Sybase Workspace", Sybase whitepaper, 2005.

[12] Thomas Erl, *Service-Oriented Architecture (A field Guide to Integrating XML and Web Service)*, Prentice-Hall, 2004.

◈ 저 자 소 개 ◈

최 미 숙 (khc67_kr@hanmail.net)

숙명여자대학교 컴퓨터과학과 대학원에서 소프트웨어공학 전공으로 석사와 박사 학위를 취득하였다. 1995년부터 1999년까지 나주대학 소프트웨어 개발과 전임교수로 재직하였으며, 현재 우석대학교 컴퓨터공학과의 초빙교수로 재직 중이다. 최근 컴포넌트 개발 방법과 컴포넌트 메트릭에 관한 논문으로 SCI급 저널에 4편이 게재되었으며 현재 전자태그(RFID)를 이용한 시스템 개발에 관한 중기청 프로젝트를 진행하고 있다. 주요 관심분야는 컴포넌트 개발 방법론, 컴포넌트 메트릭, 서비스지향 아키텍처 설계, 임베디드 시스템, RFID등이다.

이 서 정 (sjlee@bada.hhu.ac.kr)

현재 한국 해양대학교 IT공학부 컴퓨터공학전공 조교수로 재직 중이며, 숙명여자대학교 전산학과를 졸업하고 동 대학교 대학원에서 석사 및 박사학를 취득하였다. 건설기술연구원의 건설부 전산화와 정통부의 컴포넌트 분류체계 개발 프로젝트 등에 참여한 바 있으며, 주요 관심 분야는 소프트웨어 개발 방법론, 소프트웨어 아키텍처, 웹 서비스 통합 등이다.