http://www.jsebs.org
ISSN: 2288-3908

The Journal of Society for e-Business Studies
Vol.24, No.3, August 2019, pp.53-71
https://doi.org/10.7838/jsebs.2019.24.3.053

서비스지향 아키텍처와 멀티소프트웨어 프로덕트라인을 결합한 웹 시스템 개발 방법

A Development Method of Web System Combining Service Oriented Architecture with Multi-Software Product Line

정일권(IlKwon Jung)*

초 록

소프트웨어 시스템이 복잡하고 대규모화 되어감에 따라 새로운 기능을 제공하기 위해 소프트웨어 컴포넌트 또는 모듈을 재사용하는 방법을 요구하고 있다. 본 논문에서는 서비스 제공자와 서비스 사용자에게 SOA와 MSPL을 결합하여 가변성 서비스를 제공하고 재사용하여 웹 시스템 개발 방법을 제시하였다. 제시한 방법은 서비스 제공자 관점에서, 재사용 가능한 가변성 서비스를 휘처 식별 가이드라인을 적용하여 신택스 기반, 기능기반과 행위기반 방법으로 휘처로 식별하고 구현하여 재사용 자산으로 관리하였다. 그리고 사용자 관점에서 서비스를 구조적으로 조합하고 재구성하는 방법으로서 워크플로우를 모델링하여 서비스를 구성하여 웹 시스템을 구현하였다. 본 논문에서 구축한 웹 시스템의 재사용을 기능점수로 측정한 결과 재사용의 증가와 유사프로젝트에 적용해 비용 절감 효과를 검증하였다.

ABSTRACT

As software systems become more complex and larger, software systems require a way to reuse software components or modules to provide new functionality. This paper designed a development method of web system combining SOA(Service Oriented Architecture) with MPSL(Multi-Software Product Line). According to provides SOA and MPSL, this paper suggested to service providers and service users to provide and reuse variable services. From the viewpoint of service provider, the suggested method identifies and implements reusable variable services as features by syntax-based, functional-based, and behavior-based methods applying feature identification guidelines and manages them as reuse assets. From the user's point of view, it is possible to develop a web system by constructing a service by workflow model as a method of structure and reconfigure services. As a result of measuring the reuse of the web system constructed in this paper by the function point, the cost reduction effect was verified by applying it to the similar project with the increase of reuse.

키워드 : 멀티 소프트웨어 프로덕트 라인, 서비스 지향 아키텍처, 서비스, 워크플로우, 휘처
Multi-Software Product Line, Service-Oriented Architecture, Service, Workflow, Feature

* Computer Engineering Department, Hanbat National University(ikjung@hanbat.ac.kr)
Received: 2019-05-28, Review completed: 2019-06-21, Accepted: 2019-07-24

1. 서 론

소프트웨어 시스템이 점점 대규모화되고 복잡해짐에 따라 시스템을 구축하는데 소프트웨어의 컴포넌트 또는 모듈의 재사용 규모가 더욱 커지고 있다. 이러한 시스템을 개발하기 위해 서로 다른 이기종 시스템에서 제공하는 일련의 기능을 수정하거나 변경하지 않고 통합하여 새로운 기능의 제품을 체계적이고 효율적으로 지원할 수 있는 방법이 필요하게 되었다.

SOA(Service-Oriented Architecture)는 분산 환경의 빠른 변화에 적용하기 위한 아키텍처 패러다임으로 확장 가능한 유연한 분산 응용 프로그램을 개발하기 위한 수단으로 제시되었다[19]. 하지만 서비스는 가변성을 바탕으로 재사용할 수 있도록 설계되지 않아 개발의 생산성과 유지보수 측면에 비효율적이다. 이에 재사용을 지원하는 기술로 소프트웨어 프로덕트 라인(Software Product Line)이 제시되었다[9, 15].

이러한 소프트웨어 프로덕트 라인의 모델링 방법, 구현 기술을 결합하고 추상화하여 복잡한 대규모 시스템을 구현하기 위해 제안된 것이 멀티 소프트웨어 프로덕트 라인 (MSPL, Multi Software Product Line)이다[16]. 본 논문에서 SOA와 MSPL을 결합하여 서비스 사용자(Service Consumer)와 서비스 제공자(Service Provider)관점에서 SOA의 계층별로 MSPL를 적용하여 웹 시스템 개발 방법을 제시한다.

SOA의 계층에 MSPL의 적용은 다음과 같이 수행한다. 서비스를 도출하기 위해 시스템 기능을 비즈니스 로직을 포함한 태스크로 구분한다. 태스크들을 도메인 자산에서 구체화된 휘처 모델로 매핑하기 위해 태스크의 기능적 제약조건과 행위적 제약조건을 분석한다. 그리고 휘처 식별 가이드라인을 제시하고 제시된 가이드라인을 적용하여 휘처를 신택스 기반, 기능기반과 행위기반으로 식별하여 서비스를 구현하고 재사용 자산으로 관리한다. 또한 서비스를 구조적으로 조합하고 재구성하는 방법으로서 워크플로우를 모델링하여 오케스트레이션 기법으로 웹 시스템을 구축한다.

본 논문에서 제시한 방법을 대학의 입시관리 웹 시스템 개발에 적용하고 기능점수(Function Point)[17]을 측정하여 비용 분석을 통해 재사용성을 검증하고 비용절감 효과를 확인하고자 한다.

2. 관련연구

2.1 서비스 지향 아키텍처

서비스지향 아키텍처는 일종의 소프트웨어 개발 방법인 서비스지향과 기업의 모든 소프트웨어 자산에 대한 전체적 그림인 아키텍처를 결합한 용어로써, 다른 소유권 통제 하에 있을 수 있는 분산된 역량을 구성하고 활용하기 위한 아키텍처 패러다임이다. 이러한 아키텍처 스타일로 표현되는 서비스지향을 소프트웨어 개발의 관점을 변화시키는데 크게 기여했다[11]. SOA의 목표는 어플리케이션 통합, 재사용성, 모듈화 및 상호 운용성과 같은 다양한 문제를 해결하고 확장 가능한 유연한 분산 응용 프로그램을 개발하기 위한 수단으로 제시되었다[19].

SOA의 기본 구성요소는 서비스 저장소(Service Registry), 서비스 공급자(Service Provider), 서비스 요청자(Service Requester)로 구성되며 서로 유기적으로 연계된다. 서비스 공급자는 서비스 명세서(Service Description)를 저장소에

등록하고 요청자에게 입력하는 값을 가공하여, 그에 해당하는 결과를 제공한다. 서비스 요청자는 서비스 제공자에 의해 제공되는 서비스를 사용하기 위해 서비스 요청자는 하나 이상의 서비스 저장소에 등록되어 있는 서비스 명세서를 찾고, 서비스를 호출하거나 연결하기 위하여 서비스 명세서를 사용한다. 서비스 저장소는 서비스에 대한 기술정보를 저장, 검색할 수 있게 한다.

SOA개발 방법론 중에 레이어(Layer)를 고려한 개발 방법론인 SOMA(Service-Oriented Modeling and Architecture)[3]는 SOA를 설계하고 모델화하고 구현하기 위해 모델링, 분석 설계 기술과 활동을 포함하며, SOA의 각 레이어내의 요소를 정의하고 각 레벨에서 일어나는 아키텍처적 결정을 하도록 한다. SOMA는 기존자산의 비즈니스 중심 상향식(Bottom-Up) 접근법과 하향식(Top-Down)접근법을 통합하는 방법으로 서비스의 식별(Identification), 명세(Specification), 실현(Realization)의 세 단계로 구성된다.

2.2 멀티 소프트웨어 프로덕트 라인

MSPL에서 어플리케이션 개발은 상호의존적인 SPL(Software Product Line)의 기능을 통합하여 대형 SPL을 개발할 수 있다. 재사용 정도는 이용 가능한 소프트웨어 프로덕트 라인의 범위에 따라 달라지며, 소프트웨어 프로덕트 라인 구현뿐만 아니라 프로세스, 도구, 요구사항, 테스트, 기술과 산출물을 재사용한다. 그리고 MSPL은 SPL간의 상호운용성(Interoperability)과 핵심자산(Core Asset)을 통합하여 복잡한 기능을 수행하는 어플리케이션 개발이 가능하다. 이러한 시스템을 설명하기 위해 Holl et al.[8]는 멀티 소프트웨어 프로덕트라인에 대한 개념을 “대규모 및 초대형 시스템을 대표하는 제품라인을 여러 독립적이고 상호의존적인 시스템으로 이루어진 시스템으로 표현 될 수 있으며, 공통성과 가변성을 관리하기 위한 상호의존적인 단일 프로덕트 라인군의 집합”으로 정의하였다.

Czarnecki et al.[6]은 휘처 모델을 기반으로 카디널리티 휘처 모델(Cardinality-Based Feature Model)을 단계적으로 통합하는 개발 접근법을 제안하였다. 각 단계마다 제약조건에 따라 특수화된 휘처 모델이 생성된다. 이 접근법에서는 레벨 n의 휘처 모델은 레벨 n에서 수행된 구성에 기초하여 구성하는 방법을 제안하였다.

Mendonca et al.[12]은 제품 구성 프로세스를 주석이 달린 휘처 모델로부터 프로세스 모델을 유도하는 방식을 제안하였다. 두 개의 서로 다른 결정 집합에 속한 두 개의 휘처들이 서로 상충되어 결정 충돌이 발생하면 우선순위가 높은 결정 집합이나 역할이 우선 결정되는 방법을 제안하였다.

Brondum et al.[5]은 대용량 소프트웨어 집약 시스템의 서브시스템간의 관계를 시스템간의 명시적 관계를 정의하고 관계 유형을 분류하는 방법으로 아키텍처 뷰포인트를 제안하였다.

2.3 SOA와 MSPL의 결합

소프트웨어 재사용은 소프트웨어 품질 및 생산성 향상이 매우 중요하다. 이러한 맥락에서 SOA와 MSPL은 공통 목표를 공유하는 두 가지 재사용 전략으로 함께 사용하여 재사용을 늘리고 강력한 조합에서 발생하는 시너지 효과를 나타내는 쪽으로 연구가 진행되었다.

Mohabbati et al.[13]은 SOA와 SPL을 결합한 접근법을 논의하는 체계적인 매핑연구를 진행했다. 대부분의 연구는 SPL 원칙과 접근방식을 SOA에 적용하여 서비스 가변성 모델링, 서비스 식별과 재사용등에 관한 연구가 진행되었으며, 이들 두 분야의 방법과 기법을 통합함으로써 발생하는 시너지 효과를 제시하였다.

Galster et al.[7]은 SOA 어플리케이션을 설계할 때 가변성을 고려하지 않았기 때문에 독립적으로 개발된 서로 다른 이 기종 SOA 시스템의 가변성과 서로 다른 요구사항을 구현하는 문제를 다른 서비스 제공자의 가변성을 관리하는 SPL과 결합을 제안했다.

Kamoun et al.[10]은 MSPL의 패러다임 기반으로 SOA 접근 방식으로 서비스 사용자와 서비스 제공자의 가변성을 상호 연관시키는 방법을 제안했다. 주요 개념은 맞춤화되고 유효하고 일관성 있는 서비스를 생성하기 위해 서비스 사용자 및 서비스 제공자용 두 개의 종속 SPL로 구성된 MSPL을 개발하는 것을 제안했다.

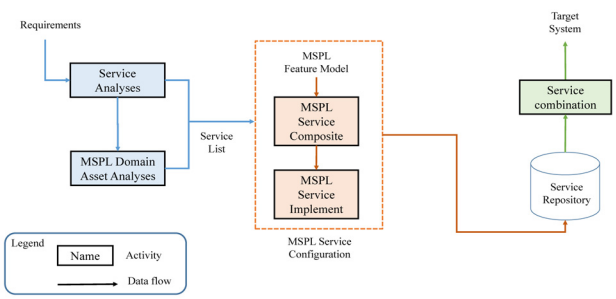
Acher et al.[2]는 프로덕트 라인 아키텍처로서 서비스를 관리하기 위해 서비스의 다양한 포인트를 여러 개별 휘처 모델로 설명하는 접근방식을 제안했다. 이 접근 방식은 서비스 가변성과 관련된 정보를 구조화하기 위해 다중 휘처 모델을 사용한다. 합성기법을 사용하여 연결된 서비스 간의 호환성에 대한 추론을 통해 전체 워크플로우의 일관성을 보장하고 서비스 구성 시 복합 서비스의 추가 구성을 위한 휘처 모델의 구성이 가능한 방법을 제안했다.

Abu-Matar et al.[1]은 휘처 모델링을 기반으로 Service Contract View, Business Process View, Service Interface View, Feature Modeling View로 다중 뷰 모델을 이용하여 가변성과 공통성을 식별하는 방법을 제안했다.

3. SOA와 MSPL을 결합한 웹 시스템 개발

3.1 SOA와 MSPL을 결합한 웹 시스템 개발 방법

본 논문의 개발 방법은 SOA와 MSPL을 결합하여 SOA의 서비스 제공자관점에서 비즈니스 프로세스 계층, 서비스 계층에서 상호 연관된 가변성을 포함한 재사용 서비스를 구현하여 자산으로 관리한다. 서비스 사용자관점에서는 어플리케이션 계층에서 재사용 서비스를 조합하여 시스템을 구축하는 프로세스를 <Figure



〈Figure 1〉 Web System Development Approach

1>과 같은 단계로 웹 시스템을 구축하였다.

3.2 비즈니스 프로세스 계층

3.2.1 서비스 분석

본 논문의 서비스 분석은 비즈니스 업무단위로 분석하여 서비스를 도출하기 위해 사용자 니즈를 수집하고 니즈들로부터 요구사항을 도출하여 비스니즈 로직을 포함한 태스크로 나누어 그룹화 한다. 그리고 태스크의 구성관계를 균형을 맞춰가며 구조적으로 분할하여 계층적으로 나타내는 RBS(Requirement Break-down Structure) 계층 구조도를 작성한다[4].

분류된 태스크들이 언제, 어떠한 순서로 기능을 수행하고 상호작용하는지를 액티비티 다이어그램(Activity diagram)을 작성하여 각 구성요소들의 기능적 흐름을 나타낸다. 이것은 시간적 흐름에 따라 순차적으로 나타나는 행위적 관점에서 발생되는 의존관계인 태스크의 제약조건을 관련, 의존, 제약, 참조, 연결로 관계유형을 파악하여 비즈니스 프로세스를 모델링한다[5].

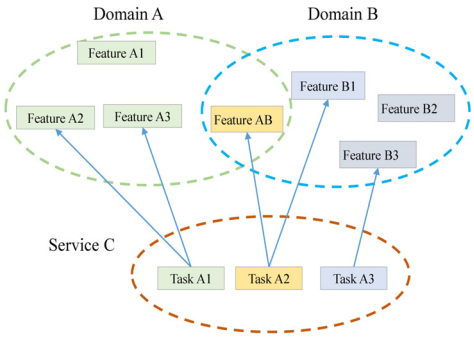
계층 구조도의 태스크와 비즈니스 프로세스에 따라 서비스를 추출하는 단계로 후보서비스 도출과 정제과정으로 서비스를 도출한다. 후보서비스 도출은 서비스간의 의존 관계를 파악하여 서비스 그룹화 및 우선순위화 단계를 진행하고 서비스간의 연관관계를 바탕으로 서비스에 결합하고 서비스 우선순위를 고려하여 최종적인 후보 서비스를 도출한다.

후보 서비스 정제는 도출된 후보 서비스를 서비스 지향 원칙인 결합성, 조합성, 자율성, 입도를 기준으로 정제한다. 또한 M4SOD의 서비스 식별 가이드라인[20]을 기초하여 본 연구에서 분할(decomposition), 결합(Composition), 공동(Share), 동형(Isomorphics) 관계를 제시하여 서비스를 도출하고 서비스 목록을 작성한다.

3.2.2 MSPL 도메인 자산 분석

MSPL 도메인 자산 분석은 도출된 서비스의 태스크가 MSPL의 어떤 도메인에 속하는가를 분석한다. 이는 도메인의 핵심자산인 휘처를 선택하여 서비스를 구성하는 서비스 식별을 위해 분석하며 태스크가 도메인내의 공통, 가변 휘처인지 분석하고 도메인 간의 공통적으로 포함되는 휘처인지 분석을 하며 집합이론을 적용한다[18].

집합이론과 추상화로 요구사항 계층구조도에서 도메인을 <Figure 2>와 같이 기능적, 행위적 의존관계를 분석하여 서비스 C를 구성하는 태스크 A1, A2, A3가 도메인 A, B에 속하는 휘처로 구성됨을 나타낸다. 그리고 각각의 태스크는 도출된 A, B 도메인에 공통으로 포함될 수 있는 태스크를 분석함으로써 공통성과 가변성을 도출한다. 분석된 도메인의 공통 관계는 태스크 A2는 도메인 A와 B의 공통 휘처인 AB와 B1으로 기능이 구성됨을 나타낸다.



〈Figure 2〉 Domain Analysis and Common Relationship

도출된 서비스를 구성하는 태스크의 기능을 MSPL 도메인 자산인 휘처와 매핑하여 휘처를 식별하기 위해 서비스 번호, 서비스 명, 서비스 도메인, 서비스 설명, 연관되는 서비스를 포함하여 서비스 목록을 작성한다.

3.3 서비스 계층

3.3.1 휘처 식별 가이드 라인의 적용

서비스를 구성하는 태스크에 해당하는 기능을 MSPL의 도메인에서 구현된 휘처 모델로부터 식별해 내는 과정으로 휘처 식별 가이드 라인을 1대 1매핑, 동음이의, 이음동의를 제시한다. 제시된 식별 방법인 1대 1매핑은 하나의 기능이 하나의 휘처로 식별되는 경우이다. 동음이의는 서로 다른 기능을 제공하는 휘처가 하나로 이름으로 명명되었다면 둘로 나누는 경우이다. 이음동의는 서로 같은 기능을 제공하는 휘처를 하나로 대치한다. 만약 두개가 서로 다르게 정의된 휘처로 같은 기능을 수행한다면 하나의 휘처로 식별한다.

휘처 식별은 MSPL의 도메인에 존재하는 휘처를 식별하기 위해서는 중복 휘처의 제거 및 휘처 상호작용 분석 등 단순히 이름뿐만 아니라 종속관계에 따라 휘처의 의미와 행위적 부분도 고려한 식별 방법으로 신택스 기반, 기능기반, 행위기반 식별 제시하여 휘처 식별에 적용한다.

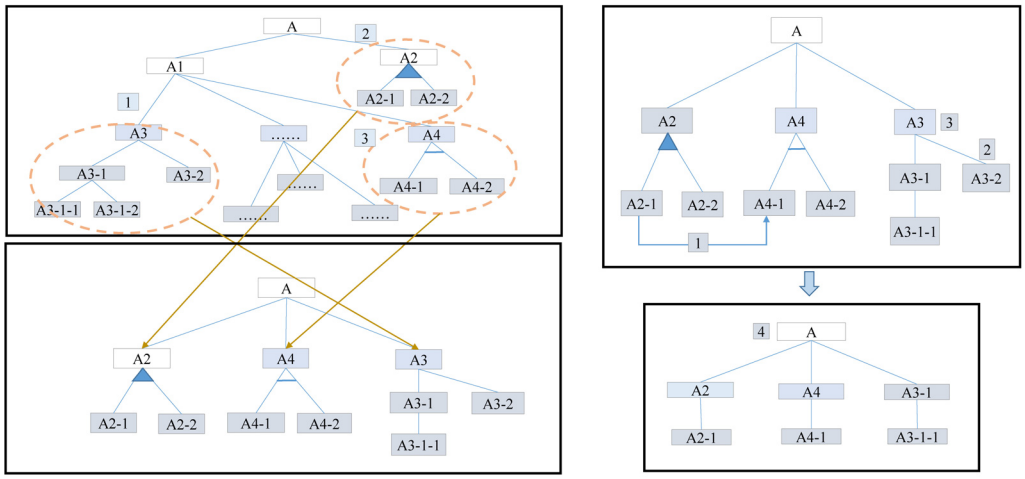
3.3.2 MSPL 서비스 구성

MSPL 서비스 구성은 서비스에 해당하는 기능들을 휘처와 매핑하는 단계이며 휘처는 여러 SPL에 존재할 수 있으므로 앞에서 제시된 휘처 식별 방법으로 구체화된 휘처 모델을 휘처 사이의 관계를 정의한 합성 규칙(Composition Rule)과 추상화(Abstraction)를 적용하여 구성하고 정제한다.

MSPL 서비스 식별은 서비스를 구성하는 기능에 해당하는 휘처를 식별하여 다양하게 나타난다. 신택스 기반 식별은 기능 매칭으로 기능 하나가 하나의 휘처로 식별될 수 있고, 여러 개의 휘처의 모음으로 식별될 수 있는지 판단하여 식별한다. 기능기반 식별은 기능의 범위를 결정하는 방법으로 도메인의 구조적 속성을 반영하는 방법으로 집합의 개념을 적용하여 합집합, 교집합, 여집합의 관계로 공통성과 가변성의 기능을 분류하고 판단하여 식별한다. 행위기반 식별은 도메인이 다르지만 동일한 휘처 이름을 갖는 경우에 휘처가 같은 휘처인지, 혹은 다른 의미를 갖는지 고려하며 그 반대로 다른 이름의 휘처를 갖지만 그 의미가 동일한 경우 하나의 휘처로 식별하고 우선순위를 고려하여 식별하고 휘처 모델로 작성한다.

MSPL 서비스 정제는 휘처의 의미가 다른 휘처의 이름을 바꾸는 경우로 도메인의 휘처의 이름을 바꾸는 경우이며 이때 도메인에 영향을 미치므로 휘처 정제과정에서 수행한다. 또한 서로 다른 도메인에서 동일한 이름으로 식별되어도 도메인의 특성에 따라 다른 휘처들이 식별될 수 있다. 또한 각 영역의 도메인 전문가의 입장에서 분석하므로 휘처 모델의 경계가 불일치하는 경우가 생긴다. 이것은 하위 휘처의 기능이 포함될지 여부를 판단하는 것을 의미한다. 그러므로 기능의 흐름에 따라 달라지는 경우도 포함하여 정제한다.

다음은 MSPL 서비스 구성을 <Figure 3>과 같이 식별 방법에 따라 식별한 휘처를 구성하여 정제하는 과정을 나타내었다.



(A) (B)

〈Figure 3〉 (A) Feature Identification and (B) Feature Model Rationalize

휘처 식별과정으로 <Figure 3>의 (A)에서 A-A1-A3에 대한 서비스 해당하는 휘처 식별을 나타낸다. ①은 A3의 이름 매핑으로 찾은 휘처를 선택한 경우이며 A3-1, A3-2 휘처를 선택하고 A3-1-1은 불필요하므로 제거하여 식별한다. ②번과 ③번은 기능 기반 식별에 해당하며 A2가 선택되면 A4가 기능적 종속관계로 포함되어 식별한다. 이것은 공통기능과 가변기능에 의해 구별되며 A2는 A2-1과 A2-2로 선택기능인데 이것은 정제과정에서 업무 흐름에 따라 결정하여 정제한다.

휘처 정제과정으로 식별된 A3를 정제하는 과정으로 <Figure 3>의 (B)에서 A2에서 A2-1의 휘처가 결정되면 A4에서는 A4-1를 선택한다. ①번에 해당하는 내용이며 이러한 판단은 비스니스 프로세스에 의해 판단할 수 있다. ②번은 A3을 구성하는 A3-1과 A3-2에서 A3-2는 해당사항이 없으므로 제거한다. ③번은 A3는 추상화에 해당하여 제거하고 하위 휘처로 대체하여 A3-1와 A3-1-1로 정제한다. 그리고 ④번은 정제된 휘처 모델 A를 재구성하여 나타내었다.

3.3.3 서비스 구현

서비스 구현은 MSPL의 도메인 공학에서 자산으로 개발된 핵심자산에서 서비스로 식별된 휘처에 해당하는 컴포넌트의 인터페이스를 통하여 연결하여 구현한다. 구현 절차는 자산에서 확인하여 구현될 컴포넌트가 존재하는지 판단하여 컴포넌트를 확인하여 없을 때에는 컴포넌트를 새로 개발하고, 존재하는 컴포넌트의 사용하여 컴포넌트 인터페이스 명세서를 작성하고 조합하여 서비스를 구현한다. 또한 구현된 서비스에 대한 명세작업을 진행한다. 명세서는 서비스 ID, 서비스 명, 휘처 구성에 관한 내용, 처리조건과 특이사항을 기술하여 서비스 명세서를 작성한다.

구현된 서비스는 서비스 리스트와 서비스 정보와 SPL의 핵심자산인 컴포넌트 매핑 정보를 데이터베이스(DataBase)에 저장하고 관리하도록 SQL 스크립트로 데이터베이스에 구현하

고 엔티티 관계를 테이블(Table)로 설계하여 구현한다. 그리고 서비스를 서비스 레지스터가 관리하며 구현되어 제공되는 서비스에 대한 정보로 서비스를 공개(Publish)하기 위해 서비스에 대한 정보를 입력하고, 관리할 수 있는 기능을 제공하며 UDDI(Universal Description, Discovery and Integration)와 상호작용을 한다.

3.4 어플리케이션 계층

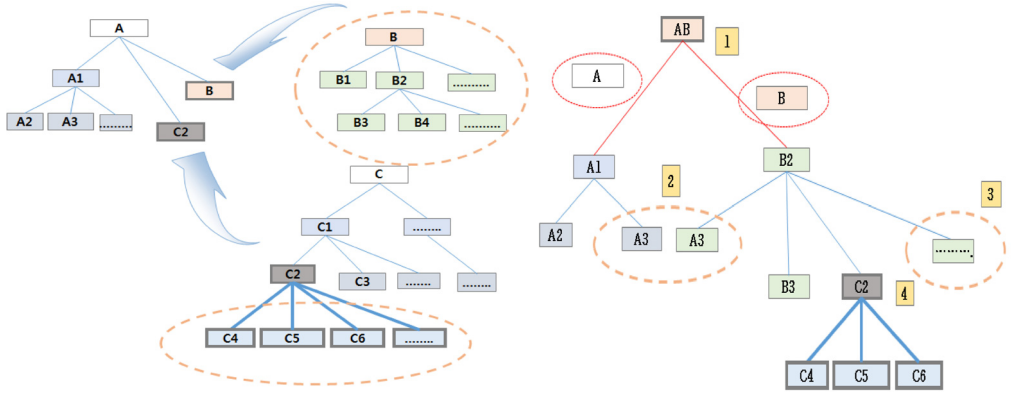
어플리케이션 계층은 서비스 계층에서 휘처 모델로 식별되어 컴포넌트로 구현된 서비스를 조합하는 단계로 진행한다. 본 논문에서 서비스 조합방법을 제시하고 워크플로우를 모델링하여 오케스트레이션 기법으로 워크플로우의 엔진을 이용하여 서비스를 조합하여 웹 시스템을 구현하였다.

3.4.1 서비스 조합

본 논문에서 서비스 조합 방법으로 구체화된 휘처 모델을 조합할 때 적용되는 모델 간 구조적 조합과 재구조화 방법을 제시한다.

휘처 모델 조합은 본 논문에서 제시한 구조적 조합방법으로 휘처 모델을 조합하는 단계이며 여러 휘처 모델 사이의 조합 및 재구조화를 수행하는 절차로 진행한다. 조합은 하나 또는 그 이상의 구체화된 휘처 모델들을 조합하여 구성하며, 조합되는 구체화된 휘처 모델은 하나 이상일 때 여러 가지 구조를 포함할 수 있으며, 기능적 제한이 있는 하나의 구체화된 휘처 모델은 다른 기능의 여러 구체화된 휘처 모델들과 다양한 형태로 조합되어 새로운 구체화된 휘처 모델로 조합한다.

구조적 조합 과정으로 <Figure 4>의 (A)에서 A에 B와 C을 조합하여 통합된 휘처 모델을 조합하는 단계를 나타낸 것이다. B는 휘처 모델 전체를 포함하고 C2는 C의 일부분을 선택하여 조합한 모델을 작성한 것이다. 그리고 조합과정에서 추상화 형태로 구성된 휘처 모델을 재구성하여 최적화한다. 조합된 휘처 모델을 <Figure 4>의 (B)와 같이 휘처 모델 재구조화는 과정을 나타내었다.



(A) (B)

〈Figure 4〉 (A) Feature Model Integration and (B)Feature Model Re-Strutcher

재구조화 과정으로 1번은 추상화에 해당하는 경우로 A, B를 제거하고 추상화 형태로 AB로 명명하여 재구성 한다. 2번은 교집합으로 두 모델에서 중복되는 경우 해당하는 경우이며 하나를 제거한다. 3번은 여집합으로 휘처 모델에서 포함할 필요가 없는 휘처를 제거한다. 4번은 합집합으로 B2의 하위 휘처에 C2가 포함되도록 결합한다.

3.4.2 워크플로우(Workflow) 모델링

워크플로우 패턴으로 병렬 워크플로우는 동시에 두 개 이상의 서비스를 제공하기 위한 것으로, 기본 워크플로우 패턴들 중 분기(split), 병합(Join) 패턴으로 구성되며 보다 큰 규모의 워크플로우를 표현할 수 있다. 또한 다중 워크플로우는 다수의 워크플로우가 동시에 실행되는 형태로 단순 워크플로우 타입의 조합으로 복잡한 워크플로우를 나타낸다. 그리고 각각 다른 여러 사용자를 위해 워크플로우 시스템을 통해 동시에 여러 개의 워크플로우 프로세스가 조합되는 경우에 사용하여 모델링한다.

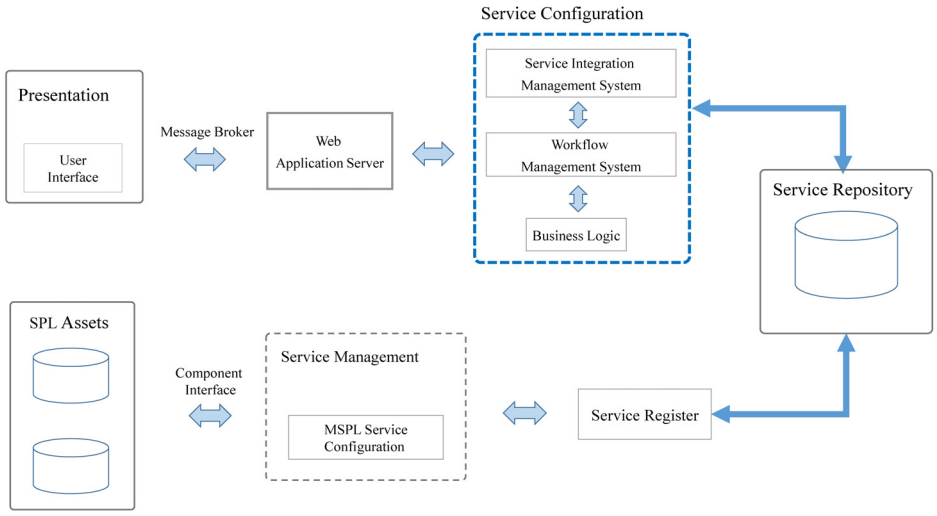
본 논문에서 워크플로우를 모델링하여 오케스트레이션 기법을 적용하여 비즈니스 프로세스에 따라 휘처 모델로 매핑되어 구현된 서비스를 조합하기 위해 비즈니스 로직과 워크플로우 따라 제공되는 도구와 시스템을 이용하여 서비스를 조합하도록 한다.

4. 입시관리 웹 시스템 구현

4.1 시스템 아키텍처 구현

본 논문에서 입시관리 웹 시스템을 SOA와 MSPL을 결합한 웹 서비스 개발방법을 적용하여 서비스 제공자, 사용자관점으로 아키텍처를 <Figure 5>와 같이 구현하였다.

입시관리 웹 시스템의 서비스 제공자관점에서 서비스 매니지먼트는 서비스를 구현하는 과정으로 MSPL 서비스 구성(MSPL Service Con-



〈Figure 5〉 Web System Architecture

figuration)으로 서비스를 도출한다. 그리고 SPL의 핵심자산인 휘처로 구성하고 컴포넌트 인터페이스와 매핑하여 서비스를 구현하도록 구성하였다. 이렇게 구현된 서비스를 서비스 레지스터가 서비스를 저장하기 위해 데이터베이스에서 저장하고 관리하고 UDDI와 상호작용하도록 구성하였다.

서비스 사용자 관점에서 제공될 서비스를 비즈니스 로직과 워크플로우에 따라 워크플로우 매니지먼트 시스템과 서비스 통합 매니지먼트 시스템을 이용하여 서비스를 통합하여 시스템을 구현하도록 구성하였다. 그리고 어플리케이션 서버를 통하여 프리젠테이션의 사용자와 상호작용하며 사용자로부터 요청 받은 내용을 메시지 브로커를 통하여 처리하도록 구성하였다.

4.2 입시관리 웹 시스템 분석

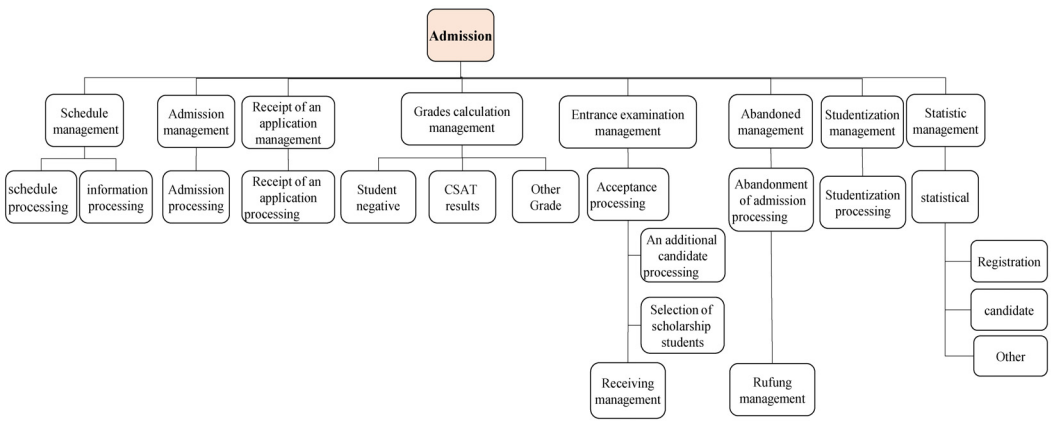
입시관리 시스템의 목표에 따라 요구사항을 분류하고 그룹화 하여 <Figure 6>과 같이 입시관리 요구사항 계층 구조도를 작성하고 각각의 태스크를 도출하였다.

도출된 태스크의 행위적 관점의 종속성을 파악하기 위해 비즈니스 업무 단위로 분류된 태스크의 흐름을 파악하고, 도출된 태스크의 기능적, 행위적 관점의 종속성 관계에 따라 포함될 태스크가 속하는 MSPL을 입시관리시스템, 회계시스템, 학사관리시스템, 통계시스템으로 4개의 SPL로 구분하였다.

태스크로 구분되는 서비스를 후보 서비스간의 의존 관계를 파악하여 후보 서비스 그룹화 및 우선순위화 단계를 수행하고 서비스간의 연관관계를 바탕으로 후보 서비스의 분할, 결합, 공동, 동형관계를 파악하고 서비스 우선순위를 고려하여 최종적인 서비스를 도출하였다. 도출된 서비스를 결합성, 조합성, 자율성, 입도를 분석하여 정제하여 태스크에 해당하는 기능을 휘처와 매핑하여 휘처를 식별하기 위해 서비스 목록을 작성하였다.

4.3 서비스 제공자 구현

본 시스템의 서비스 제공자 구현은 서비스를 구현하는 MSPL 서비스 구성과정으로 분석과



〈Figure 6〉 Admission Management Requirement Break-Down Structure

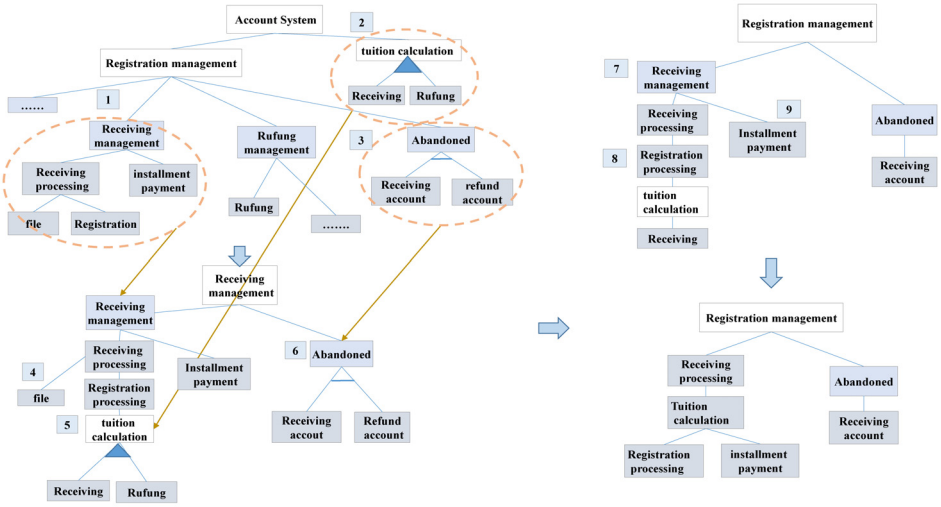
정에서 도출된 서비스를 MSPL의 핵심자산인 휘처로 구성하기 위해 휘처 식별방법인 신택스 기반식별, 기능 기반식별, 행위 기반식별 방법으로 식별하였다. 그리고 식별된 휘처를 구체화된 휘처 모델로 구성하고 정제하였다. 또한 컴포넌트 인터페이스와 매핑하여 서비스를 구현하여 서비스 레지스터가 데이터베이스에 저장하고 관리하도록 테이블을 설계하여 구현하였다.

4.3.1 서비스 구성

서비스 구성으로 서비스 식별을 휘처 모델의 합성 규칙과 추상화를 휘처 식별방법에 적용하여 식별된 수납관리에 대한 서비스를 <Figure 7>과 같이 식별과정으로 휘처를 식별하여 휘처 모델을 구성하고 정제하였다.

서비스 식별과정은 <Figure 7>에서 수납관리에 대한 회계시스템의 도메인에서 신택스 기반으로 이름 매핑은 1번에 해당하며 수납관리의 하위 휘처를 포함하여 수납처리, 분납관리 휘처로 식별하였다. 그리고 2번의 등록금 계산과 3번의 계좌관리는 수납관리의 종속관계에 해당하므로 포함하여 식별하였다. 4번은 수납처리에서 등록처리만 필요하고 파일처리는 제거하여 식별하였다. 그리고 5의 등록금계산은 수납처리에 해당하는 부분이므로 수납금액만 선택하였다. 6에서는 등록금계산의 수납금액에 종속되는 계좌관리는 수납계좌를 선택하여 휘처 모델을 구성하였다.

식별된 수납관리를 정제하는 과정은 <Figure 7>과 같다. 7번은 수납관리와 수납관리는 동일한 이름으로 추상화되어 있어 루트는 남기고 제거한다. 8번에서 등록처리는 등록금계산과 수납금액이 결정된 상태에서 프로세스 흐름에 따라 재구조화한다. 9번은 분납관리는 수납금액의 하위로 수납관리와 같은 레벨로 조정하여 재구조화한다. 정제과정의 결과는 루트인 수납관리와 수납처리-등록금계산-수납금액-등록처리, 분납관리, 계좌관리-수납계좌로 정제하였다.



〈Figure 7〉 Service Configuration

4.3.2 서비스 구현

휘처 모델로 구성된 서비스를 각각의 SPL의 도메인 공학에서 자산으로 개발된 핵심자산에서 컴포넌트를 확인한다. 컴포넌트 인터페이스 명세서의 연관되는 컴포넌트와 사용되는 컴포넌트 및 조건에 따라 <Figure 8>과 같이 등록관리를 합격관리의 합격처리와 등록금에 관련된 수납, 가상계좌에 대한 컴포넌트 인터페이스와 매핑하여 서비스를 구현하였다.

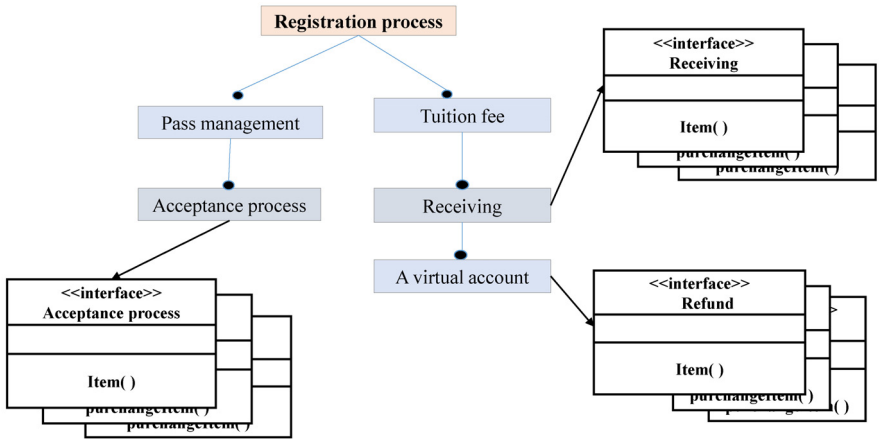
구현된 서비스를 자산으로 재사용하기 위해 서비스 리스트와 서비스 구성정보와 SPL의 핵심자산인 컴포넌트 매핑 정보를 데이터베이스에 저장하고 관리하도록 서비스 레지스터가 관리하며 서비스 매니지먼트에서 제공하는 서비스에 대한 정보로 서비스를 공개하기 위해 서비스에 대한 정보를 입력하고 관리할 수 있도록 구현하였다.

서비스 레지스터는 XML 스키마를 작성하여 데이터베이스에서 조회, 수정, 삭제, 입력의 쿼리로 데이터베이스와 상호작용하는 메시지 브로커 역할을 수행하도록 하였다. 그리고 데이터베이스와 SQL Mapping Code을 통하여 XML 스키마인 SqlMap을 작성하여 데이터베이스를 관리하도록 구현하였고 UDDI와 상호작용하도록 하였다.

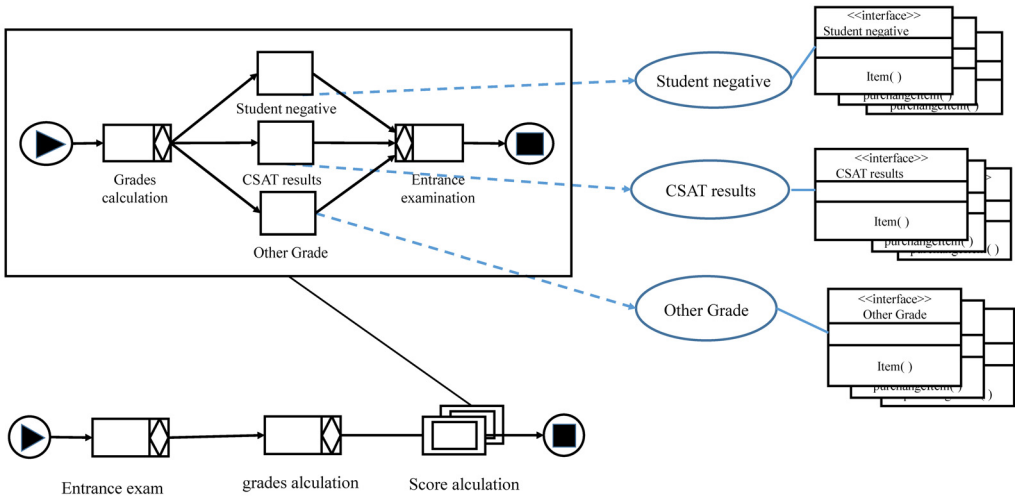
4.4 시스템 사용자 구현

서비스 사용자 구현은 서비스 조합방법을 워크플로우로 모델링하여 오케스트레이션기법으로 서비스를 통합하여 서비스를 구성하고 시스템을 구현하도록 하였다.

입시관리의 워크플로우를 <Figure 9>와 같이 입시로 시작하여 성적산출로 진행하며 다중 워크플로우인 점수계산으로 끝나는 워크플로우를 작성한 것이다. 점수계산은 다중 워크플로우로 성적산출은 병렬 워크플로우로 분기하여 학생부 점수, 수능점수, 기타점수를 산출하여 병합하여 합격자를 선정하는 입시사정으로 진행하도록 모델링하였다. 그리고 워크플로우에 따라 위에서 구현된 서비스 인터페이스와 매핑관계를 나타내었다.



〈Figure 8〉 Service Implemention



〈Figure 9〉 Admission Management Workflow

모델링된 워크플로우는 워크플로우 관리 시스템과 서비스 통합 매니지먼트 시스템을 이용하여 서비스를 조합하여 시스템을 구현하도록 하였다. 또한 사용자에게 시스템을 제공하는 프리젠테이션은 사용자 인터페이스에 따라 웹 어플리케이션 서버와 메시지 브로커를 통하여 상호작용하도록 사용자 인터페이스를 구현하였다.

5. FP 산정과 비용적 측면의 분석

SOA와 MSPL을 결합한 웹 시스템 구축 방법의 평가를 위해 대학의 입시관리 재개발 프로젝트 기능점수(Developemnt Function Point)로 측정하였다[17]. 그리고 비용 이익을 분석하기 위해 비용 및 이익 함수로 구성되는 SIMPLE (Structured Intuitive Model for Product Line Economics)을 활용하여 비용을 분석하였다. 분석 결과로 개발비용 절감 효과를 확인하였다[14].

5.1 재개발 규모 및 원가 산정

본 논문에서 웹 시스템의 재사용 측정 절차로 SW사업 대가 산정 가이드의 소프트웨어 재개발비 FP 산정방법으로 직접경비 및 이윤은 계산하지 않고 순수 개발비만 측정하였다.

5.1.1 기능유형 식별 및 대상 규모 산정

본 논문에서 구축한 입시관리 웹 시스템의 재사용률을 측정하기 위해 우리대학의 입시관리 시스템에 적용하여 FP와 기능 유형을 식별하고 산정하였다. 산정결과로 총 FP는 128.4중에 신규개발 규모는 8.0FP, 수정 없이 재사용 규모는 20.0FP, 수정대상 규모는 100.8 FP로 산정되었다.

5.1.2 변경률 산정

데이터 기능 설계 변경률은 각 데이터 기능별 설계 변경률과 각 데이터 기능의 규모가 차지하는 비중에 의해 결정되며 수정 대상 데이터

규모는 45 FP이며 설계 변경률은 21.7% 산정되었다. 그리고 트랜잭션 기능 설계 변경률은 사용자인터페이스(UI), 업무처리로직(BL), 데이터처리로직(DL) 영역으로 산정되며 대상 트랜잭션 규모는 55.8 FP이며 설계 변경률은 17.6%로 산정되었다. 통합 설계 변경률은 (45/101×21.7%) + (56/101×17.6%)로 산정결과는 19.4%로 산정되었다.

5.1.3 재개발 특성 평가

재개발에 소요되는 노력은 특성요인에 의해 영향을 받으며 재사용 소프트웨어 평가 노력 수준은 기본 모듈의 조사, 결과문서 작성인 2로 산정하였고, 재사용 소프트웨어 난이도 수준은 프로그램의 구조화 정도는 어려움(40), 애플리케이션 관점에서의 명확성은 쉬움(20), 프로그램 소스코드의 서술 정도는 보통(30)으로 산정하였다. 그리고 재사용 소프트웨어 친숙도는 거의 친숙함(0.2)으로 산정하였다.

5.1.4 소프트웨어 규모 및 원가 계산

총 변경률이 50% 이하이기 때문에 재개발 소프트웨어 규모는 계산한 수정 후 재개발 소프트웨어 규모에 별도로 산정된 수정 없이 재개발 소프트웨어 규모와 신규개발 소프트웨어 규모를 합산하여 계산한다.

① 수정 후 재개발 규모 = 100.8×[2+37.8×{1+0.02×(30×0.2)}] / 100 = 44.65 FP
② 수정 없이 재개발 규모 = 20FP×25%(시험 단계 비율) = 4.90 FP
③ 신규개발 규모 = 44.65 FP+4.90 FP+8 FP = 57.55 FP
④ 보정 전 재개발 원가산정 보정 전 재개발 원가 = 57.55×519,203 = 29,880,133원
⑤ 보정 후 재개발 원가산정: 29,880,133×1.28×0.88×0.91×0.94×1.0 = 28,790,182원

본 논문에서 제시한 SOA의 서비스에 MSPL을 적용한 입시관리 웹 시스템의 총 FP는 128.4이며, 신규 개발에 대한 FP는 57.6로 측정되어 70.8 FP가 절감되었다. 그리고 총 FP와 재개발 FP는 1 : 0.448의 비율을 가지며 FP 비율로 재사용률을 구하면 FP 구축 비율 44.8%로 재사용 비율 55.2%로 산출하였다.

5.2 비용적 측면의 분석

본 논문에서 비용 이익을 분석하기 위해 비용 및 이익 함수로 구성되는 SIMPLE을 활용하여 비용을 분석하였다. SIMPLE의 네 개의 기본적인 비용함수에서 $C_{org}(\ )$는 도메인공학에서 수행되는 비용에 관한사항이기에 $C_{cab}(\ )$에 포함하여 산출한 식 (1)을 적용하였다.

$$\sum_{i=1}^{n} C_{Prod}(\mathrm{Product}_i, t) - (C_{cab}(t) + \sum_{i=1}^{n} C_{unique}(\mathrm{Product}_i, t) + C_{reuse}(\mathrm{Product}_i, t)) \quad (1)$$

- $C_{cab}(\ )$: 관련된 인수가 주어졌을 때 특정 범위를 충족시키기 위한 핵심자산 토대를 개발하기 위해 얼마만한 비용이 드는지를 반환하는 함수이다.
- $C_{unique}(\ )$: 관련된 인수가 주어졌을 때 핵심자산에 있는 자산을 토대로 하지 않는 제품의 유일한 부분

을 개발하는데 얼마만한 비용이 드는지를 반환하는 함수이다.

- C_{reuse}( ): 관련된 인수가 주어졌을 때 핵심자산 토대로부터 핵심자산을 재사용해서 제품을 개발하는데 얼마만한 비용이 드는지를 반환하는 함수이다.

SIMPLE의 비용 대한 이익을 계산하기위해 위에서 제시한 FP에 재사용률 입시제도 변경, 기능추가 등 변경요인을 반영하여 우리대학의 3개 프로젝트에 적용하여 개발에 소요되는 노력을 구하면 A는 총 128.4 FP에 대한 신규개발 57.6 FP 감소된 FP는 70.8로 재사용률은 55.2%이며 투입 인력수는 0.56PY이며, B는 총 132.0 FP에 대한 신규개발 50.2 FP 감소된 FP는 81.8로 재사용률은 62.0%이며 투입 인력수는 0.58PY이며, C는 총 125.0 FP에 대한 신규개발 38.1 FP 감소된 FP는 86.9로 재사용률은 69.5%이며 투입 인력수는 0.55PY으로 측정되었다.

본 논문에서는 SIMPLE의 비용 대한 이익을 계산하기 위해 비용이익함수에 대한 투입인력수와 핵심자산의 사용에 대해서도 활용도를 비율과 개발비용을 산출하여 비용절감효과를 확인하기 위해 C_{cab}()의 비용을 총 FP인 128.4의 투입인력의 20%로 핵심자산의 기반마련 인력을 0.11PY로 산정하여 식 (1)에 따라 산출했다.

비용 절감을 투입인력의 비율로 계산하면 A는 27%, B는 34%, C는 39%의 절감을 나타내었고 총 개발 금액과 비용절감 비율에 따라 비용을 S/W 기술자 평균 입금 중급자 기준 순수 인건비로 산정한 결과 각각 9,140,847원, 11,708,508원, 12,860,600원에 대한 비용 절감을 나타내었다. 이러한 비용절감 효과는 <Table 1>과 같이 재사용률에 따라 조직이 취하는 이익은 비례하고 있다는 것을 확인했고, 실질적으로 시스템 구축에 대한 개발원가의 절감을 확인하였다.

비용 분석결과는 A, B, C의 프로젝트에 재사용 비율인 55.2%, 62.0%, 69.5%와 전체 구축비용과 절감된 비용과의 비율인 27%, 34%, 39%로 재사용 비율이 늘어남에 따라 비용 감소 비율도 증가하여 비용이 감소됨을 확인했다. 이러한 결과로 개발비용 절감과 오류감소로 인한 위험감소 효과가 있으며 생산성 향상과 유지보수 측면에서 추가 및 변경이 필요한 해당 서비스만을 수정 혹은 교체, 추가함으로써 신속하고 유연하게 대응할 수 있는 용이성으로 비용절감 효과를 확인하였다.

6. 결 론

소프트웨어 시스템이 점점 대규모화되고 복잡해짐에 따라 시스템을 구축하는데 소프트웨

〈Table 1〉 Cost Savings Effect

project	C_{prod} (PY)	C_{cab} (PY)	C_{unique} (PY)	C_{reuse} (PY)	building cost (PY)	cost saving effect (PY)	effort (M/M)	building cost (won)
A	0.56	0.11	0.25	0.05	0.41	0.15	1.83	9,140,847
B	0.58	0.11	0.22	0.05	0.38	0.20	2.35	11,708,508
C	0.55	0.11	0.17	0.06	0.34	0.21	2.58	12,860,600

어의 컴포넌트 또는 모듈은 재사용 규모가 더욱 커지고 있다. 이러한 시스템을 개발하기 위해 서로 다른 이기종 시스템에서 제공하는 일련의 기능을 수정하거나 변경하지 않고 통합하여 새로운 기능의 제품을 체계적이고 효율적으로 지원할 수 있는 방법이 필요하게 되었다.

본 논문은 비즈니스 환경변화에 대처하기 위해 SOA와 MSPL을 결합하여 체계적인 재사용을 지원할 수 있도록 휘처로 가변성을 포함한 서비스를 구성하고 구현하였다. 또한 서비스 사용자와 서비스 제공자 관점에서 SOA의 계층인 비즈니스 프로세스 계층, 서비스 계층과 어플리케이션 계층에 적용하여 서비스를 재사용하여 복잡한 처리를 요구하는 웹 시스템을 구축할 수 있는 방법을 제시하였다.

본 논문에서 SOA의 계층에 MSPL을 적용은 시스템 기능을 비즈니스 로직을 포함한 태스크로 구분하여 도메인 자산에서 구체화된 휘처 모델로 매핑하기 위해 태스크의 기능적 제약조건과 행위적 제약조건을 분석하여 서비스를 도출하였다. 그리고 도출된 서비스를 MSPL의 핵심자산인 휘처를 선택하는 휘처 식별 가이드라인을 제시하였고, 기존의 방법보다 향상된 휘처 식별을 위해 신택스기반, 기능기반과 행위기반으로 식별하여 서비스를 새로운 휘처 모델로 구성하였다. 그리고 휘처 모델에 해당하는 핵심자산에서 구현된 컴포넌트 인터페이스와 매핑하여 서비스를 구현하여 재사용 자산으로 관리하였다. 또한 구체화된 휘처 모델로 식별된 서비스를 조합하기 위해 서비스 조합방법을 제시하였고 워크플로우를 모델링하여 오케스트레이션 기법으로 서비스를 조합하도록 하여 웹 시스템을 구축하였다.

본 논문에서 구축한 입시관리 웹 시스템을 기능점수를 측정하고 비교분석을 통해 핵심자산의 재사용 비율이 55.2%로 산출되었고, 비용이익을 분석하기 위해 비용 및 이익 함수로 분석한 결과 27%의 비용절감이 되었고 유사프로젝트에서도 같은 결과를 나타내었다. 이는 핵심자산의 재사용을 의미하므로 개발비용 감소 효과뿐만 아니라 오류감소로 위험감소 효과가 있다. 또한 사용자의 요구에 맞는 서비스 구성이 가능하며, 구현의 복잡성을 감소시켜 서비스의 유효성이 검증되어 생산성을 향상되고 재사용 비율에 따라 비용이 감소됨을 확인하였다.

향후 연구는 MSPL의 핵심자산을 관리하고 지원하는 자동화된 도구와 실행 시간에 자동으로 적응하는 동적 환경에서 서비스를 조립하고 생산하는 방법에 관한 연구와 웹 시스템의 다양한 영역으로 확대하는 연구가 추가적으로 요구된다.

References

[1] Abu-Matar, M., Gomaa, H., “Variability modeling for service oriented product line architectures,” Proceedings of the 15th International Software Product Line Conference (SPLC’11), IEEE Computer Society, Washington, DC, USA, pp. 110-119, 2011.

[2] Acher, M., Collet, P., Lahire, P., and France, R., “Managing variability in workflow with feature model composition operators,” Proceedings of the 9th International Conference on Software Composition

(SC'10), LNCS, Springer, p. 16. 2010.

[3] Arsanjani, A., Service-Oriented Modeling and Architecture(SOMA), IBM developer Works, Nov 2004.

[4] Blanchard, B. and Fabrycky, W., Systems Engineering and Analysis (Ed.), Prentice Hall, 2006.

[5] Brondum, J. and Zhu, L., "Towards an architectural viewpoint for systems of software intensive systems," Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge (SHARK '10), ACM, New York, NY, USA, pp. 60-63. 2010.

[6] Czarnecki, K., Helsen, S., and Eisenecker, U. W., "Staged configuration through specialization and multilevel configuration of feature models," Software Process: Improvement and Practice, Vol. 10, No. 2, pp. 143-169, 2005.

[7] Galster, M., Avgeriou, P., and Tofan, D., "Constraints for the design of variability-intensive service-oriented reference architectures-An industrial case study," Journal of Information and Software Technology, Vol. 55, No. 2, pp. 428-441, 2013.

[8] Holl, G., Grünbacher, P., and Rabiser, R., "A Systematic Review and an Expert Survey on Capabilities Supporting Multi Product Lines," Journal Information and Software Technology (IST), Vol. 54, No. 8, pp. 828-852, 2012.

[9] Hwang, B. and Jin, Y., "Application of Software Product Line Engineering for Developing Web Application Families," The Journal of Society for e-Business Studies, Vol. 22, No. 2, pp. 39-60, 2017.

[10] Kamoun, A., Hadj Kacem, M., and Hadj Kacem, A., "Multiple Software Product Lines for Service Oriented Architecture," 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 56-61, 2016.

[11] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F. Metz, R., and Hamilton, B. A., "Reference model for service oriented architecture 1.0," OASIS Standard, 12, 2006.

[12] Mendonca, M., Cowan, D., and Oliveira, T., "A process-centric approach for coordinating product configuration decisions," Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS '07), IEEE CS, Waikoloa, HI, USA, pp. 1-10. 2007.

[13] Mohabbati, B., Asadi, M., Gasevi Lc, D., Hatala, M., and Nuller, H. A. M., "Combining service-orientation and software product line engineering: A systematic mapping study," Journal of Information and Software Technology, Vol. 55, No. 11, pp. 1845-1859, 2013.

[14] Clements, P. C., McGregor, J. D., Cohen, S. G., The Structured Intuitive Model for Product Line Economics (Simple), Technical REPORT CMU/SEI-2005-TR-003, 2005.

[15] Pohl, K., Böckle, G., and van Der Linden, F. J., “Software Product Line Engineering Foundations,” Principles and Techniques. 2005.

[16] Rosenmüller, M. and Siegmund, N., “Automating the Configuration of Multi Software Product Lines,” Proceedings of Fourth International Workshop on Variability Modelling of Software-Intensive Systems, pp. 123-130, 2010.

[17] Sikka, G., Kaur, A., and Uddin, M., “Estimating Function points: Using Machine Leaening and Regression Models,” IEEE ICETC, 2010.

[18] Thompson, J. M. and Heimdahl, M. P. E., “Structuring product family requirements for n-dimensional and hierarchical product lines,” Requirements Engineering, Vol. 8, No. 1, pp. 42-54. 2003.

[19] Thomas, E., “Service-oriented architecture: concepts, technology, and design,” Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[20] Yun, H., “M4SOD: The Service Oriented Development Methodology for SOA,” Soongsil University, 2006.

저 자 소 개

정일권 (E-mail: ikjung@hanbat.ac.kr)

2012년～현재 한밭대학교 컴퓨터공학과 박사과정

1998년～현재 한밭대학교 재직 중

관심분야 소프트웨어 제품라인 공학, 멀티 소프트웨어 제품라인 공학, 서비스 지향 아키텍처