서비스 지향 아키텍처를 위한 컴포넌트기반 시스템의 서비스 식별

(Service Identification of Component-Based System for Service-Oriented Architecture)

이 현 주 † 최 병 주 †† 이 정 원 †††
(Hyeonjoo Lee) (Byoungju Choi) (Jungwon Lee)

요 약 서비스 지향 아키텍처(Service Oriented Architecture)는 기업 인프라의 복잡성 및 유지비용을 최소화하고, 기업의 생산성과 유연성을 극대화할 수 있는 것으로, 경영환경이 빠르게 급변하는 최근에 떠오른 이슈이다. 엔터프라이즈 수준에서 서비스 지향 아키텍처를 도입하는 전략은 조직의 비즈니스 프로세스를 정의하고 이에 필요한 서비스를 모델링하여, 필요한 서비스를 찾아내거나 개발하는 하향식 전략이 대부분이다. 그러나 대부분의 조직은 SOA를 도입하면서도 기존에 사용하던 컴포넌트 시스템을 최대한 재사용할 수 있기를 바라고 있다. 본 논문에서는 이미 개발된 컴포넌트 기반 시스템에서 입출력 데이타가 아닌 GUI 이벤트 정보를 이용하여 상향식 방법으로, 서비스 재사용성과 유지보수성을 고려하면서 비즈니스 서비스 모델에 적합한 크기의 서비스를 식별할 수 있는 방법을 제안한다. 본 논문에서 제안한 방법은 실제 129개의 GUI와 13개의 컴포넌트를 가진 경영정보시스템에 적용한 결과 기존의 컴포넌트를 기준으로 서비스를 식별하는 것보다 결합도가 5배정도로 약해지면서 3개의 서비스가 명확히 구분되었고, 식별 후 연관관계에 따른 문제점도 약 49%정도 줄어드는 것을 보였다.

키워드 : 서비스 지향 아키텍처, 서비스 식별, 서비스 모델링, 순차패턴

Abstract Today, businesses have to respond with flexibility and speed to ever-changing customer demand and market opportunities. Service-oriented architecture (SOA) is the best methodology for minimizing the complexity and the cost of enterprise-level infrastructure and for maximizing the productivity and the flexibility of an enterprise. Most of the enterprise-level SOA delivery strategies deal with the top-down approach, which organization has to define the business processes, to model business services, and to find the required services or to develop new services. However, a lot of peoples want to maximally reuse legacy component-based systems as well as to deliver SOA into their organizations. In this paper, we propose a bottom-up approach for identifying business services with proper granularity. It can improve the reusability and maintenance of services by considering not data I/O of components of legacy applications but GUI event patterns. Our proposed method is applied to MIS with 129 GUIs and 13 components. As a result, the variance of the coupling value of components is increased five times and three business services are distinctly exposed. It also provides a 49% improvement in reducing the relationship problems between services over a service identification method using only partitioning information of components.

Key words : Service-Oriented Architecture(SOA), Service Identification, Service Modeling, Sequential Patterns

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2007-(C1090-0701-0032))

† 정 회 원 : 이화여자대학교 컴퓨터정보통신학과
lee.hyeonjoo@gmail.com
†† 종신회원 : 이화여자대학교 컴퓨터정보통신학과 교수
bjchoi@ewha.ac.kr
††† 종신회원 : 아주대학교 전자공학부 교수
jungwony@ajou.ac.kr

논문접수 : 2007년 4월 16일
심사완료 : 2007년 12월 28일

정보과학회논문지: 소프트웨어 및 응용 제35권 제2호(2008.2)

1. 서 론

서비스 지향 아키텍처(Service-Oriented Architecture)는 기업의 인프라의 복잡성 및 유지비용을 최소화하고, 기업의 생산성과 유연성을 극대화할 것으로 기대되어 차세대 소프트웨어 아키텍처로서 주목받고 있다[1].

서비스 지향 아키텍처의 서비스 식별 방법에 대한 연구는 서비스 지향 아키텍처를 지원하기 위한 개발 방법론인 SODA(Service Oriented Development of Applications)[2], SOUP(Service Oriented Unified Process)[3], SOMA(Service-Oriented Modeling and Architecture)[4], SOAD(Service Oriented Analysis and Design)[5] 등에 언급되어 있으나, 개발 절차상의 서비스 식별 방법을 하향식(Top-down), 상향식(Bottom-up), 절충형(Hybrid) 방식으로 구분할 뿐 세부적인 절차나 활동을 명시하지 않고, 조금 구체적이라고 하더라도 대부분 하향식 방법론에 대해 제시하고 있다[6].

현재의 대부분 서비스 지향 아키텍처 도입 전략 및 관련 연구는 하향식에 그 초점을 두고 있다. 서비스 지향 아키텍처가 자연스럽게 비즈니스 모델에서 출발하여 서비스 모델로 매핑 될 수 있도록 하는 데 의의를 두고 만들어진 패러다임이기 때문이다. 그러나 대기업의 경우, 혹은 중기업이라 하더라도 전사적인 시스템의 서비스 지향 아키텍처 도입을 계획할 때, 하향식 도입을 결정 하기란 매우 어려운 실정이다. 비용과 시간이 많이 소모되는 하향식 방법을 채택하는 모험보다는 이미 잘 정의된 컴포넌트 기반의 시스템의 구조를 크게 변화시키지 않으면서, 즉 최대한 재사용하면서 서비스 모델을 도출해 내어 서비스 지향 아키텍처의 장점을 도입할 수 없을까에 좀 더 관심을 가질 수밖에 없다.

한편, 많은 개발자들이 컴포넌트 모델과 서비스 모델 간의 차이를 모호하다고 생각한다. 따라서 컴포넌트 기반의 시스템을 서비스 지향 아키텍처로 전향하는 경우, 기존의 컴포넌트를 그대로 ESB(Enterprise Service Bus)나 웹 서비스에 얹어 서비스화 하는 것이 기업에서 시스템을 서비스 지향 아키텍처화하는 가장 쉬운 길이라고 오해 하고 있다. 이러한 상향식 방법은 비즈니스 모델로부터 출발한 것이 아니라 레거시 시스템(엔터프라이즈 어플리케이션)으로부터 서비스를 도출해 나가는 방식으로서 특별히 서비스-지향 분석이나 설계를 반영할 수 있는 절차 및 방법이 고려되고 있지 않다. 이는 비즈니스 모델로부터 출발하지 않았기 때문에 단순 엔터프라이즈 어플리케이션 시스템의 조립(composition)을 통해 좀 더 큰 크기(granularity)의 컴포넌트를 생성하는 데에 그치기 쉽다. 그러나, 분산과 통합의 논리적 측면에서 엔터프라이즈 어플리케이션은 통합의 관점으로 업무 통합만 확보된다면 기술과 업무의 결합도는 높아도 무방하다고 보는 반면, 서비스 지향 아키텍처는 분리의 관점으로 업무와 기술의 분리함으로써 재활용을 가능하게 한다. 그러므로 이렇게 컴포넌트 시스템의 구현환경만 서비스 지향 아키텍처로 바꾸는 작업은 실제 서비스 지향 아키텍처 도입의 초보적인 단계로 언급되고 있으며[1], 궁극적인 서비스 지향 아키텍처의 설계 목표인 민첩성(agility), 융통성(flexibility), 약결합(loosely coupled) 등을 달성하기 어렵게 만든다.

단순 컴포넌트의 합병(merge)만으로 서비스를 식별하는 것이 아니라, 비즈니스 서비스 모델에 적합한 비즈니스 기능을 가진 서비스로 식별하기 위해서는 비즈니스 서비스 모델에 대한 정보가 필요하다. 시스템의 GUI(Graphical User Interface)는 컴포넌트나 애플리케이션과 같은 시스템의 기능보다 한 차원 위에서 사용자의 비즈니스 행위에 대한 정보를 제공한다. 즉, GUI는 한 개의 컴포넌트마다 발생하는 것이 아니라, 하나의 GUI 화면에 여러 개의 컴포넌트가 상호 작용한다고 보고, 하나의 GUI로부터 다른 GUI로 가는 링크를 연결하면, 시스템의 사용 패턴을 추적할 수 있기 때문이다. 그러므로 상향식 서비스 식별 시 시스템의 GUI 이벤트 경로 정보를 이용하면 사용자의 비즈니스 프로세스 패턴을 추출할 수 있고, 컴포넌트를 그대로 서비스화 하는 것보다는 비즈니스 서비스에 가까운 서비스로 식별할 수 있다.

한편, 서비스 지향 아키텍처는 서비스 간에 약결합을 지향하고 서비스간의 의존도를 지양한다. 약결합 형태의 서비스를 지향하기 위해서는 서비스간 결합도가 낮아야 한다. 물론 이러한 특성은 비단 서비스 지향 아키텍처에만 해당하는 것이 아니라 컴포넌트 개발 방법론에서도 계속해서 강조해 왔던 특성이다. 그러나 컴포넌트가 기능적인 단위로 최소 결합도를 가지도록 식별된 반면, 서비스는 독립적으로 새로운 가치를 창출할 수 있는 단위로서 기능보다는 한 차원 위에서 비즈니스 규칙과 그 프로세스를 포함한다는 면에서 컴포넌트와 구별된다[1]. 따라서 기업이 전사적인 시스템을 서비스 지향 아키텍처로 전이하고자 할 때, 컴포넌트 기반으로 잘 정의된 기존의 컴포넌트를 그대로 서비스로 정의 할 것이 아니라, 비즈니스 서비스 모델과 적합한 정도로 큰 크기의 서비스로 컴포넌트를 합성 혹은 분할(partitioning) 하는 방법이 필요하다.

여기서, 비즈니스 서비스 모델은 고정적으로 존재하는 것이 아니다. 그러므로 서비스 식별시 각 기업의 현실에 맞도록, 기업의 경영자와 시스템 엔지니어가 협력하여 추상적인 시나리오를 구체화 하면서 식별하고 있으며 이는 하향식 설계 방법에서 가능하다. 그러나 기업의 현실은 기존의 시스템에서 이미 정의해 놓은 컴포넌트들

을 최대한 재사용하면서도 서비스 지향 아키텍처의 여러 가지 이점을 취할 수 있는 방법을 모색하고 있으며, 이러한 요구로 상향식 방법의 개발이 요구되고 있다.

따라서 본 논문에서는 시스템의 GUI 정보를 이용하여 기존의 컴포넌트를 서비스로 식별하는 상향식 방법을 제안하고자 한다. 동시에 서비스 재사용 및 유지보수성을 위해 단위 서비스를 적절한 크기로 조정 가능하고, GUI들의 결합도 문제를 최소화할 수 있는 상향식 서비스 식별 방법론을 제안하고 실제 시스템에 적용하여 본다. 본 논문에서 제시하는 방법론은 GUI의 수가 컴포넌트의 수에 비해 상대적으로 많은 컴포넌트 시스템에 더욱 유용하게 적용되어질 수 있다. 본 논문에서는 이미 개발된 컴포넌트 기반 시스템에서 입출력 데이타가 아닌 GUI 이벤트 정보를 이용하여 상향식 방법으로, 서비스 재사용성과 유지보수성을 고려하면서 비즈니스 서비스 모델에 적합한 크기의 서비스를 식별할 수 있는 방법을 제안한다. 본 논문에서 제안한 방법은 실제 129개의 GUI와 13개의 컴포넌트를 가진 경영정보시스템에 적용한 결과 기존의 컴포넌트를 기준으로 서비스를 식별하는 것보다 결합도가 5배정도로 약해지면서 3개의 서비스가 명확히 구분되었고, 식별 후 연관관계에 따른 문제점도 약 49%정도 줄어드는 것을 보였다.

본 논문의 구성은, 2장에서는 관련 연구를, 3장에서는 서비스 지향 아키텍처기반 서비스 식별을 위한 상향식 접근 방법론을 기술한다. 4장에서는 3장에서 제안한 방법론을 실제 시스템에 적용하여 적당한 서비스의 크기 제공 여부와 재사용성을 실험을 통해 검증하고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 서비스 지향 아키텍처

서비스 지향 아키텍처란 소프트웨어의 구성요소, 이 구성요소의 가시적인 속성, 이들의 관계로 구성된 구조 혹은 시스템의 구조를 의미하는 것[7]으로써, 어플리케이션 프론트 엔드, 서비스, 서비스 레파지토리, 서비스 버스의 주요 개념에 바탕을 둔 소프트웨어 아키텍처라고 할 수 있다. 서비스 지향 아키텍처의 서비스는 재사용할 수 있는 로직 단위로 분리되어야 하고, 다른 서비스와 조립 가능해야 하며 서비스 로직이 명확한 경계 안에 존재하여 자율성을 가져야 한다. 그리고 서비스끼리 상호작용하기 위해 필요한 최소한의 정보만을 공유하여야 하며, 느슨한 결합을 기초로 상호 작용하도록 설계되어야 한다. 즉, 서비스 지향 아키텍처는 소프트웨어 시스템을 느슨하게 연결된 서비스끼리 상호 협업(Collaboration)하는 집합으로 소프트웨어 시스템 개발의 향상을 가져다주는 아키텍처 패러다임이다[8]. 그림 1

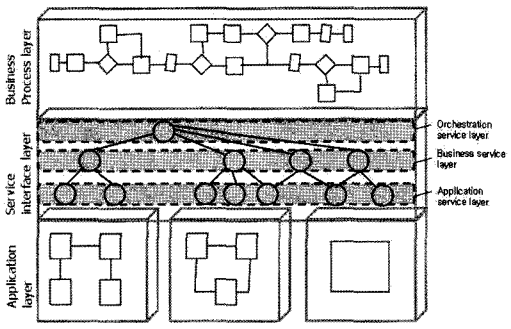


그림 1 서비스 지향 아키텍처의 계층[9]

은 서비스 지향 아키텍처의 핵심적인 요소를 이루는 서비스의 계층을 보여준다[9].

서비스 지향 아키텍처의 서비스 계층은 그림 1에서 보는 바와 같이 가장 하위의 소프트웨어 프로그램으로 구축된 어플리케이션 계층, 가장 상위의 비즈니스 전문가에 의해 잘 구축된 비즈니스 프로세스 계층 그리고 비즈니스 프로세스 계층과 어플리케이션 계층을 연결하는 서비스 인터페이스 계층으로 구성된다. 서비스 인터페이스 계층은 어플리케이션을 서비스로 잘 식별한 어플리케이션 서비스 계층, 비즈니스 계층을 서비스로 식별한 비즈니스 서비스 계층, 그리고 이것을 하나로 연동시켜주는 오케스트레이션 계층으로 나뉜다.

기존의 컴포넌트를 그대로 서비스로 식별하는 것은 서비스 모델링을 가장 하위의 어플리케이션 계층만으로 규정짓는 것으로 향후 이 서비스들을 어떻게 비즈니스 프로세스 서비스 계층과 결합하며 협업할 것인지에 대한 기준이 모호해 진다. 그러므로 본 논문에서는 어플리케이션 계층과 비즈니스 프로세스 서비스 계층 사이에 존재하는 GUI 정보를 이용하여 서비스를 식별함으로써 서비스 인터페이스 계층의 어플리케이션 서비스 계층을 비즈니스 프로세스 서비스 계층에 가까운 서비스로 식별하고자 한다.

2.2 서비스와 컴포넌트

서비스 지향 아키텍처를 구축하기 위한 핵심 개념인 서비스는 기존 컴포넌트 개발 방법론(CBD)의 컴포넌트와 구분되어야 한다. 서비스는 명확한 기능적인 의미를 지닌 소프트웨어 컴포넌트로 고차원의 비즈니스 개념을 캡슐화하고 있다[1]. 컴포넌트 개발 방법론의 컴포넌트는 클래스/객체들이 상호 의존적인 경향이 있고 종종 만들어진 상황(Context)에도 영향을 받으나 서비스 지향 아키텍처의 서비스는 각각의 서비스가 독립적으로 개발될 수 있고, 다른 서비스 혹은 시스템의 전체 상태에 영향을 주지 않고 정해진 기능을 수행할 수 있어야 한다.

표 1 컴포넌트와 서비스의 차이점

CBD의 Component	SOA의 Service
미립질	조립질
시스템의 관점에서 컴포넌트 도출	비즈니스의 관점에서 서비스 도출
기능 중심적	비즈니스 프로세스 중심적
Context-dependent (동종기술간 호환)	Context-Independent (이기종기술간 호환가능)
객체 지향적	메시지 지향적
어플리케이션 블록	서비스 조립
재사용성	재사용성, 확장성, 유연성

서비스와 컴포넌트는 재사용성(Reusable)이라는 이점에서는 동일하지만, 차이점을 가진다. 우선, 표 1에서 보는 바와 같이 입자의 크기(Granularity)가 컴포넌트는 미립질(Fine granularity)인데 반해 서비스는 조립질(Coarse granularity)로 서비스가 더 크고, 컴포넌트가 시스템 관점에서 도출되는 데 반해 서비스는 비즈니스 관점에서 도출되므로 비즈니스 가치를 상승시킨다. 그러므로 기존의 컴포넌트를 서비스로 식별할 때 비즈니스 관점에서 서비스 크기에 맞는 적절하게 컴포넌트를 합치거나 분리해야 한다. 또한 컴포넌트는 시스템 관점에서 만들어지는데 반해 서비스는 EA(Enterprise Architecture)와 결부한 비즈니스 관점에서 도출되고, 컴포넌트가 객체 지향적인 개발로 입출력의 데이타 중심이었다면 서비스는 메시지 지향적으로 프로세스(User)가 발생하는 이벤트 중심이다. 즉, 서비스는 컴포넌트와 같은 재사용성에 비즈니스 측면에서의 확장성과 유연성이 덧붙여진 개념이라고 할 수 있다.

2.3 서비스의 식별(Service Identification)

잘 모델링된 서비스는 SOA의 재사용성과 유지보수성을 결정하므로 SOA 개발 프로세스단계에서 서비스를 식별하는 것은 매우 중요하다. 서비스 식별이란 기존의 시스템 혹은 비즈니스 도메인으로부터 재사용자원인 서비스를 식별하는 것으로 비즈니스 도메인을 고려하여야 하고, 서비스 재사용성을 위해 비즈니스 서비스 모델과 적합한 정도의 기존 객체나 컴포넌트보다 큰 입자의 크기가 식별되어야 한다.

기존의 서비스 식별에 관한 연구[2-5]는 서비스 개발 단계(Service Based Development) 방법론인 SODA, SOUP, SOMA, SOAD에 하나의 단계로서 기술되고 있다. 서비스를 식별하는 방법에는 하향식과 상향식 그리고 이 둘을 혼합한 절충형 방식이 있는데 하향식 방법은 비즈니스 서비스 계층부터 구성해 나가는 것으로, 비즈니스 도메인으로부터 후보 서비스를 도출해가는 방식이고, 상향식은 그림 1의 어플리케이션 서비스 계층부터 구성하는 방식으로 이미 구성된 레거시 시스템의 컴포넌트로부터 서비스 후보를 도출해 나가는 방식이다. 또, 절충형은 상향식과 하향식의 두 방식을 혼합한 방법이다. 그러나, 기존의 서비스 식별에 관한 연구는 표 2에서 보는 바와 같이 대체로 하향식 방법론에 대해서 언급하고 있고, 언급하더라도 구체적인 단계 및 단계별 수행 내용을 명시하고 있지 않다.

표 2 서비스 개발 단계 방법론

방법론	특징
SODA	• CBD, 분산시스템 개발, SOA의 공통 요소를 뽑아 적용 • SOA를 구현하기 위한 서비스 설계, 개발, 조합에 관한 원칙 정의 • (문제점) 서비스 단위나 세부 절차에 대하여 구체적으로 명시하지 않음
SOAD	• OOAD, EA, BPM 같은 기존의 모델링 개념들을 SOA에 적용시킴 • 하향식과 상향식을 연결하여 기존시스템의 통합을 위한 아이디어 제시 • (문제점) 세부적인 절차나 활동을 명시하지 않음
SOUP	• RUP와 XP의 특징만을 모아서 SOA에 적용시킨 방법론 • 소프트웨어 개발 6단계를 정의하고, 단계별 활동을 정의 • (문제점) 비즈니스 프로세스, 서비스 규약(choreography)이나 서비스 저장소 같은 SOA에 대한 요구사항을 지원하지 않음
SOMA	• 기존 IBM의 SOAD를 참조하여 확장된 방법론 • 서비스식별, 명세, 실현 세단계로 구성 • 하향식과 상향식 방식을 통합하여 정의 • (문제점) 서비스 식별에 대한 구체적인 활동이 제시되지 않음.

그러나, 본 논문에서는 기존의 컴포넌트 시스템의 GUI를 이용한 상향식 서비스 식별 방법론을 제시하고자 한다. GUI는 비즈니스 레벨과 어플리케이션 레벨의 중간레벨에 위치하여 GUI를 이용한 서비스 식별은 컴포넌트만을 이용한 상향식방법론의 서비스 식별보다 사용자의 이벤트 발생 패턴 반영을 통해 비즈니스 프로세스 측면을 고려할 수 있다는 장점을 가진다. 다음 표 3은 GUI가 컴포넌트와 서비스 가운데 위치하여 컴포넌트보다는 좀 더 서비스에 가까운 처리단위와 입출력, 사용자를 반영할 수 있음을 비교하였다.

표 3 컴포넌트, GUI, 서비스

	Component	GUI	Service
처리 단위	Function	User Interaction	Business Process
입출력	Data(I/O)	Event	Message
사용자	Developer	User/ Developer	User/ Developer

2.3 결합도

서비스 지향 아키텍처는 서비스 간에 약결합을 지향한다. 서비스간 결합도가 높으면 한 모듈의 변화에 따른 다른 모듈로의 파장이 커지게 되어 시스템의 유지보수가 어려워지기 때문이다.

기존 객체지향에서 사용되는 결합도는 공유 인스턴스 변수를 가지는 메소드 쌍의 수 또는 메소드에 의해 참조되는 인스턴스 변수로 정의하고[10,11], 컴포넌트 개발 방법론(CBD)에서 컴포넌트 식별에 사용되는 결합도는 클래스 간의 메시지 호출 수[12] 및 메소드 호출 유형과 방향에 따른 의존성 특성을 고려한다. 컴포넌트 개발 방법론에서의 응집도 연구는 대체적으로 데이타, 메시지 등 컴포넌트나 객체의 데이타 명세(Data specification)에 의해 결정되지만, 서비스 지향 아키텍처의 서비스는 2.2 절의 서비스 개념에서 보듯이, 비즈니스 요구 사항, 즉 사용자의 행위를 모델링하여 이를 기반으로 식별되어야 하므로 데이타 명세화는 다른 방법으로 결합도를 고려해야 한다.

따라서 본 논문에서는 서비스가 비즈니스 관점에서 생성된다는 점에 주목하여, 서비스간 결합도를 사용자가 발생하는 이벤트 정보를 담고 있는 GUI정보를 이용하여 재정의해 보고, 결합도로 서비스의 크기를 조정할 수 있도록 하였다.

3. GUI 기반의 상향식 서비스 식별 단계

레거시 컴포넌트 시스템을 서비스 지향 아키텍처로 구축하기 위한 서비스 식별 과정은 그림 2에서 보는 바와 같이 크게 후보 서비스 식별을 위한 준비 단계인 전처리 과정과 순차이벤트의 중복성 발견을 위한 알고리즘을 적용하여 후보 서비스를 식별하고 정제하는 과정으로 나뉘며, 이를 통해 최종 서비스를 식별한다. 여기서 최종 서비스란 상향식 서비스 식별방법론을 통해 최종으로 얻어진 서비스를 말한다.

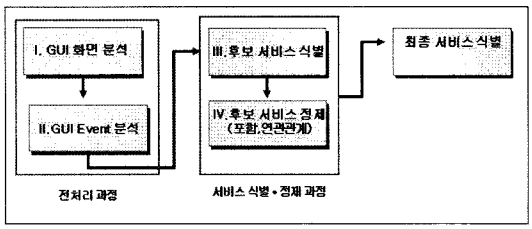


그림 2 상향식 서비스 식별 과정

3.1 전처리 과정

I. GUI 화면 분석

서비스 지향 아키텍처로 구축할 시스템의 화면을 체계적으로 분석하여 GUI 이벤트 분석을 한다. 모든 시스템의 GUI 화면은 그림 3과 같은 형식으로 나타낼 수 있고, 이런 분류과정을 통해 모든 이벤트에 체계적인 시리얼 번호를 붙임으로써 사용자의 이벤트를 고려한 서비스 식별 과정을 용이하게 한다. GUI 이벤트를 발생시키는 부분은 그림 3의 명시적인 부분(Explicit Part)으로 테이블, 그래프, 검색 그리고 그 외 부분으로 나누어지고 그 각각에 그림, 텍스트, 메시지, 아이콘, 라디오 버튼 등을 통해 이벤트가 발생된다. 암시적인 부분(Implicit Part)는 GUI의 제목이나 기능, 윈도우창/팝업창 여부, 이전 화면과 다음 화면에 대한 전반적인 정보를 표시한다. 이벤트 시리얼 번호는 '이벤트를 발생하는 GUI화면(R1) - 명시적 부분 또는 암시적 부분 - 그에 따른 이벤트 발생 유형'을 나타내며, 예를 들어 R1-Exp-Tb-Int-Ic 는 이벤트를 발생시키는 R1화면의 명시적 부분 중 테이블의 인터렉션을 일으키는 아이콘에 의해 발생되는 이벤트임을 보여준다.

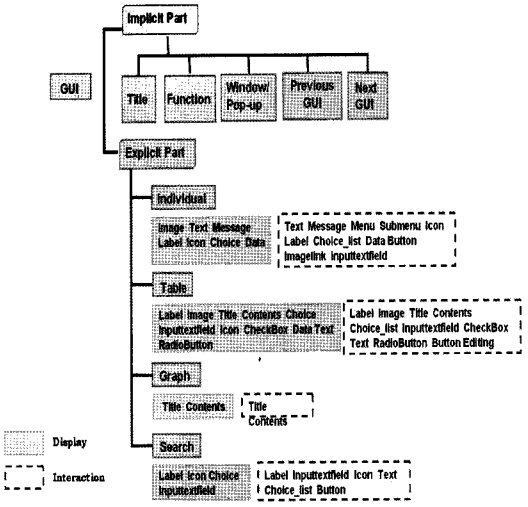


그림 3 GUI 형식을 나타낸 텍사노미

II. GUI 이벤트 분석

사용자의 GUI 이벤트 패턴을 분석하기 위해 GUI 이벤트 흐름도를 작성한다. GUI 이벤트 흐름도는 GUI 각 화면은 클래스로, 이벤트는 화살표로 표시하여 GUI 화면 별로 발생되는 출력이벤트와 입력이벤트를 나타낸 것이고, GUI 형식을 나타낸 텍사노미를 기반으로 작성된 이벤트의 시리얼 번호를 명시하여, 이벤트에 대한 정보를 표시한다. GUI 이벤트 흐름도를 통해서 가장 많은 이벤트를 발생시키는 GUI화면에 대한 정보와 GUI화면 간 발생하는 이벤트관계 정보를 시각적으로 볼 수 있다. 본 논문의 실험을 위해 적용한 사례에서 작성된 GUI 이벤트 흐름도의 일부 모습은 그림 4와 같다.

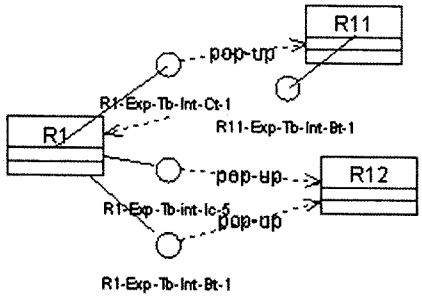


그림 4 GUI 이벤트 흐름도

그림 4는 R1, R11, R12라는 세 개의 GUI화면 간의 이벤트 흐름을 보여주는데 R1은 R11, R12로의 R1-Exp-Tb-Int-Ct-1과 R1-Exp-Tb-Int-Ic-5라는 이벤트 시리얼 번호를 가지는 출력이벤트를 통해 R11과 R12 GUI화면을 팝업형태로 활성화함을 보여준다.

3.2 서비스 식별 및 정제 과정

전처리 과정을 통해 분석된 GUI 화면에 대한 정보와 GUI 이벤트의 정보를 바탕으로 후보 서비스를 식별하고 정제한다. 그림 2의 서비스 식별·정제 부분을 구체적으로 나타내면 그림 5와 같다.

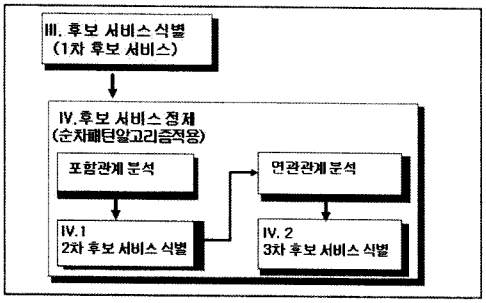


그림 5 후보 서비스의 식별 및 정제 과정

III. 후보 서비스 식별(1차 후보 서비스)

1차 후보 서비스는 서비스 간 출력 이벤트 발생 최다 빈도수를 기준으로 핵심루트(정의1)를 설정하고 핵심루트에서 발생되는 이벤트를 포함하는 GUI화면들을 모두 묶어서 하나의 1차 후보 서비스로 식별한다. 핵심루트의 정의는 경험적인 수치를 공식화한 것이다.

정의 1. 핵심루트

OE(x) 가 x의 총 출력 이벤트의 수 일때,

$MaxOE = MAX(OE(GUI_1), OE(GUI_2),..., OE(GUI_n))$

$MinOE = MIN(OE(GUI_1), OE(GUI_2),..., OE(GUI_n))$

If $OE(GUI_k) > (MaxOE+MinOE) \times \alpha$ then

$CoreRoot_i = GUI_k$

(이때, α는 임의의 비율, k는 임의의 GUI 번호)

IV. 후보 서비스 정제

핵심루트를 이용하여 식별한 1차 후보서비스는 핵심루트끼리 혹은 핵심루트와 서브GUI들 (Level1, Level 2 등)간에 포함 또는 연관관계가 발생한다. 1차 후보 서비스 사이에 발생한 포함 및 연관관계는 순차 패턴 알고리즘으로 구한 결합도와 서비스의 개념 및 크기를 고려한 기준결합도로 파악한다.

여기에 쓰인 순차 패턴 알고리즘은 엘리먼트 간의 유사성이 인식된 최소화된 구조로부터 다중 문서간의 최대 유사 경로를 추출하기 위한 변형 순차 패턴(Adapted sequential patterns) 마이닝 알고리즘[13]으로 후보 서비스들 간에 가장 유사한 경로를 찾아냄으로써 연관된 GUI들을 파악할 수 있다. 순차 패턴 알고리즘은 이벤트들의 경로 표현을 앞서 작성했던 GUI 이벤트 흐름 다이어그램으로부터 추출하여(Sort), 식별(Identification)한 후에, 모든 추출된 경로들을 정수로 교체하는 단계(Transformation)를 거친다. 그 다음, 길이 1, 2, ..., n의 빈발 경로를 공유여부에 따라 찾고(Sequence), 마지막으로 공유경로로 판단된 경로 중 최대 유사 경로를 추출(Maximal)하는 과정을 거친다. 이 과정을 통해 어떤 GUI화면들이 후보서비스(CS) 사이에 연관 관계 및 포함관계를 가지는지 파악할 수 있다. 결합도(Coh)에 관한 정의는 아래와 같다.

정의 2. **GUIp** 와 **GUIq** 사이의 결합도

CE(P) : CSp와 결합된 후보서비스(CS)로 이벤트를 발생하는 GUI 출력 이벤트수

NG(P) : CSp와 결합되어 있는 다른 후보서비스(CS)의 GUI수

$$Coh(GUIp,GUIq) = \frac{\sum_{i=p}^{q}(CE(P)_i + 1}{\sum_{i=1}^{n}(CE(P)_i + NG(P)_i + 1}$$

기준결합도(β)는 최종 서비스의 개념과 크기를 고려하여 임의로 정하는 것으로, 정의1의 α와 더불어 최종 서비스의 크기를 조절 할 수 있다. 기준 결합도에 따라 서비스의 식별이 달라지므로, 식별하고자 하는 서비스에 맞게 적용 가능하다. 결합도와 기준 결합도를 이용하여 포함 및 연관 관계(정의3)를 파악한다.

정의 3. 포함관계(**Inclusion Relation**) / 연관관계 (**Connection Relation**)

```
if (Coh(CSp,CSq) ≥ β)
{
if ((Coh(CSp,CSq) > Θ) and (Coh(CSq,CSp) > β))
        {
            if ((Coh(CSp,CSq) ≥ Coh(CSq,CSp))
                Inclusion Relation(CSp ⊃CSq)
            Else
                Inclusion Relation(CSq ⊃CSp) }
        Inclusion Relation(CSp ⊃CSq)}
Else
        Connection Relation  }
```

즉, 결합도가 기준 결합도보다 큰 경우는 포함관계로, 결합도가 기준결합도보다 작은 경우는 연관관계로 파악

한다.

IV-1. 2차 후보 서비스 식별

포함관계를 가지는 1차후보서비스는 하나의 서비스로 묶어 2차 후보 서비스로 식별한다.

IV-2. 3차 후보 서비스 식별

3차 후보 서비스는 각 후보 서비스간 연관 관계 고려가 필요하다. 순차패턴 알고리즘을 이용하여 후보 서비스 간 연관관계에 있는 GUI를 파악한다. 연관관계는 아래와 같이 분류할 수 있다.

1. 한 후보서비스의 Root에서 다른 후보서비스의 Root를 호출
2. 한 후보서비스의 Root에서 다른 후보서비스의 Subset 호출
3. 한 후보서비스의 Subset에서 다른 후보서비스의 Root 호출
4. 한 후보서비스의 Subset에서 다른 후보서비스의 Subset을 호출

각각의 연관관계 유형은 또다시 이벤트 발생 유형에 따라 참조관계인 경우(Retrieval)와 수정, 생성, 삭제 관계(Update)로 나눌 수 있다.

연관관계를 가지는 부분의 서비스 식별 방법은 두 가지가 있는데 그림 6과 같이 연관된 부분을 후보 서비스 각각에 중복하여 편입시키는 방법과 연관된 부분을 쪼개거나 변형하여 후보 서비스에 편입시켜 서비스를 식별하는 방법이 있다.

4가지의 연관관계 분류 유형과 그 각각마다 발생 가능한 이벤트 발생 유형을 어떻게 처리하느냐에 따라 크고 작은 문제점이 발생한다. 그러므로 최종 서비스 식별을 위해서는 이러한 문제점을 어떻게 해결하느냐가 관건이다. 연관관계를 가진 GUI의 정보를 바탕으로 연관

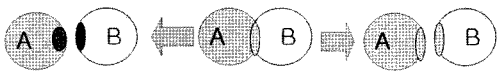


그림 6 연관 관계의 서비스 식별 처리

관계에 따른 문제점 발생 유형과 그에 따른 문제점을 최소화 하여 최종 서비스를 식별할 수 있도록 하기 위해 제시하는 방안은 표 4와 같다. 특히, 이벤트 발생 유형이 Update인 경우는 데이타베이스의 수정 및 변형을 일으키는 부분이라 분리하기가 쉽지 않기 때문에 상황에 따라 디자이너가 판단해야 한다.

4. 적용사례 및 분석

본 논문에서 제시한 서비스 지향 아키텍처의 서비스 식별 방법론을 실제 국내 기업의 경영정보시스템에 적용한 사례 및 분석에 대해 소개한다.

4.1 적용 대상 및 실험 목적, 방법

국내 기업에서 실제 사용되고 있는 경영 정보 시스템에 본 논문에서 제안한 서비스 식별 방법론을 적용했다. 이 엔터프라이즈 수준의 시스템은 컴포넌트 개발 방법론에 의해 잘 정의된 13개의 상위의 컴포넌트들로 구성되어 있으며, 129개의 GUI화면과 상호작용 한다.

13개의 컴포넌트는 기능적으로 독립이 가능한 단위로 설계 되어 있으나, 정확히 129개의 컴포넌트가 배타적으로 GUI에 할당되는 것이 아니라 매우 복잡하게 GUI와 연결되어 있었다. 즉, 컴포넌트의 독립성은 보장하도록 개발되었으나, 실제 경영정보 시스템이 여러 정보에 의해 비즈니스 프로세스가 구동되기 때문에, 하나의 화면이 여러 개의 컴포넌트와 상호 작용하고 있는 엔터프라이즈 애플리케이션이라 할 수 있다.

이 시스템은 표 5와 같은 특징을 갖는다. 13개의 컴포넌트는 편의상 대문자 알파벳으로 명명하였고, '컴포넌트(해당 GUI수)'란 컴포넌트와 해당 컴포넌트를 구성하는 GUI수를 나타낸 것이다. 본 논문에서는 표 5에서 제시한 경영 정보 시스템을 대상으로 다음과 같은 목적과 방법으로 실험하였다.

목적 1. 크기 조정이 가능한 서비스 식별

본 논문의 서비스 식별 방법에서 제안한 크기 조절을 가능하게 해주는 두 개의 한계값, 정의 1의 α와 기준결합도(3.2.2.1참조) β에 따라 서비스 크기를 조정해 보고

표 4 연관 관계에 따른 문제점 최소화 방안

연관관계에 따른 문제점 발생 유형		문제점 최소화 방안
한 후보서비스의 Root에서 다른 후보서비스의 Root를 호출한 경우	Retrieval	연관된 부분을 쪼개거나 변형하여 후보 서비스에 편입
	Update	상황에 따라 디자이너가 판단
한 후보서비스의 Root에서 다른 후보서비스의 Subset을 호출한 경우	Retrieval	연관된 부분을 후보 서비스 각각에 중복하여 편입
	Update	연관된 부분을 후보 서비스 각각에 중복하여 편입
한 후보서비스의 Subset에서 다른 후보서비스의 Root를 호출한 경우	Retrieval	연관된 부분을 쪼개거나 변형하여 후보 서비스에 편입
	Update	상황에 따라 디자이너가 판단
한 후보서비스의 Subset에서 다른 후보서비스의 Subset을 호출한 경우	Retrieval	연관된 부분을 후보 서비스 각각에 중복하여 편입
	Update	연관된 부분을 후보 서비스 각각에 중복하여 편입

표 5 적용 사례: 컴포넌트 기반 경영 정보 시스템

컴포넌트 수	총 GUI 화면 수	컴포넌트 (해당 GUI 수)
13	129	A(22), C(25), D(31), E(5), F(5), G(2), H(4), J(5), K(3), P(5), Q(4), R(3), S(15)

기존의 잘 정의된 컴포넌트를 그대로 서비스로 식별 했을 경우와 비교해 본다. 이는 단순 크기 비교를 의미하는 것이 아니라 GUI 정보를 이용하여 서비스간의 결합도를 분산시켜 약결합(Loosely coupled)된 서비스를 추출해 낼 수 있다는 것을 의미한다.

목적 2. GUI 재사용 문제점을 최소화 한 최종 서비스 식별

서비스의 식별 후, 분할된 GUI를 재구성 할 때, 연관관계에 따른 심각한 문제가 발생한다면 서비스의 재사용성에 저해될 수 있다(IV-2 참조). 식별된 서비스간의 연관관계로 발생하는 문제점(표 8 참조)의 개수를 컴포넌트 자체를 서비스로 식별했을 때와 비교해 본다.

실험 방법은 3장에서 제시한 바와 동일하게 수행하였다.

I. GUI 화면 분석 경영 정보 시스템의 GUI화면정보를 이용하여 GUI 형식을 텍사노미 형태로 구축하였고, 이를 통해 GUI 이벤트를 발생하는 모든 정보를 시리얼 번호로 정리하였다.

II. GUI 이벤트 분석 경영 정보 시스템의 GUI화면 전체에 대한 GUI 이벤트 흐름도를 작성하였다.

III. 서비스 식별 (1차 후보 서비스) GUI 이벤트 흐름도로부터 최다 GUI 출력이벤트 수를 발생시키는 GUI를 정의 1에 의해 핵심 루트로 정하였다. 본 실험에 사용된 경영 정보 시스템의 경우, 총 129개 GUI 화면의 최다 출력 이벤트(MaxOE)는 19이고 최소(MinOE)는 0이므로 적절한 서비스 크기를 위해 α를 50%로 하면 (19+0)/0.5=9.5가 된다. 따라서 아래 그림과 같이 10개 이상의 출력 이벤트를 가지는 6개, 즉 A1(CS1_1), C2(CS1_2), C3(CS1_3), D5(CS1_4), S11(CS1_5), D3(CS1_6)이 핵심루트가 된다. 그리고 각 핵심루트로부터 발생되는 이벤트의 시퀀스로 도달 가능한 GUI들을 그림 7과 같이 정리하여 1차 후보 서비스로 정리한다. 편의상 129개의 화면 번호를 컴포넌트 이름과는 상관없이 1~129번으로 재정의하여 표기하였고, CS1(First Candidate Service)은 1차 후보서비스들의 집합을 나타낸다. 예를 들어 CS1_1은 자기 자신을 포함, {3, 102, 103, …, 127, 1}을 GUI 집합으로 갖는다.

IV. 서비스 정제 1차 후보서비스를 기준으로 section 3.2에서 제시한 순차패턴 알고리즘을 적용하여 결합도를 분석한다. 분석한 결과는 4.2절의 표 6과 같다. 본 실험

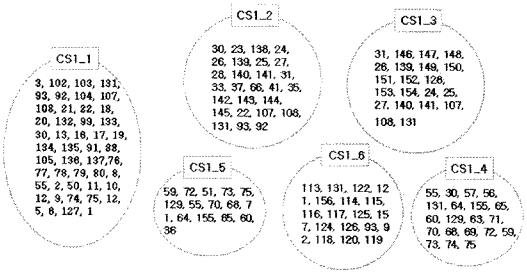


그림 7 적용한 경영 정보 시스템의 1차 후보 서비스

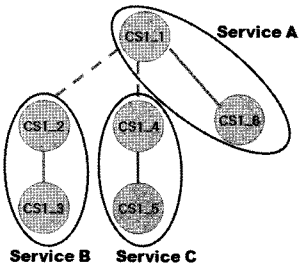


그림 8 2차 후보 서비스 식별

에서는 기준 결합도 β는 80%로 하였다. 정의3에 기준 결합도 80을 적용하여 포함관계를 분석하고 2차 후보서비스로 식별한 결과는 그림 8과 같다.

본 실험에 사용된 경영 정보 시스템은 그림 8에서 보는 바와 같이, CS1_1과 CS1_6, CS1_2과 CS1_3, CS1_4과 CS1_5이 각각 묶여 Service A, Service B, Service C 3개의 2차 후보 서비스(그림 8의 실선원)로 식별되었다. 2차 후보 서비스 간의 연관관계(그림 8의 점선)에 있는 GUI 들은 순차 패턴 알고리즘[13]의 결과물로 파악하였고, 이벤트들의 문제점(4.2절의 표 8)을 정리하였다. 3절의 표4를 이용 문제점을 최소화할 수 있는 방안을 적용, 독립적인 3개의 최종 서비스로 식별하였다.

4.2 실험 및 실험 분석

실험목적1: 시스템의 크기를 고려한 서비스 식별

1) 본 논문에서 제안한 방식 적용

본 실험에 사용된 13개의 컴포넌트를 본 논문에서 제안한 방법으로 1차 후보 서비스를 식별하였을 때의 결합도는 다음 표 6과 같다.

표 6 본 논문에서 제안한 방식 적용 시 결합도

MIS	CS1_1	CS1_2	CS1_3	CS1_4	CS1_5	CS1_6
CS1_1		50	25	37.5	6.25	2.5
CS1_2	36		84	0	4	84
CS1_3	19	100		0	0	0
CS1_4	25	8.3	0		83.3	0
CS1_5	5	0	0	100		0
CS1_6	100	0	0	25	0	

표 7 기존 컴포넌트의 1차 후보 서비스 식별 시 결합도 (기준: 행, 단위: %)

	A	C	D	E	F	G	H	J	K	P	Q	R	S
A		40	42.1	100	0	0	40	0	50	42.9	26.7	0	0
C	12.5		10.5	0	100	0	0	22.2	0	0	40	0	0
D	22.5	10		0	0	0	0	0	25	0	13.3	0	0
E	0	0	0		0	0	0	0	0	0	0	0	0
F	0	10	0	0		0	0	0	0	0	0	0	0
G	0	0	0	0	0		0	0	0	0	0	0	0
H	5	0	0	0	0	0		22.2	0	0	0	0	0
J	0	10	0	0	0	0	40		0	28.6	0	0	100
K	7.5	0	10.5	0	0	0	0	0		0	0	0	0
P	10	0	0	0	0	0	0	22.2	0		0	0	0
Q	15	20	10.5	0	0	0	0	0	0	0		0	0
R	0	0	0	0	0	0	0	0	0	0	0		0
S	0	0	0	0	0	0	0	22.2	0	0	0	0	

표 6에서 결합도는 최대값과 최소값를 제외했을 때, 4%에서 84%까지 넓은 분포도를 보이며, 분산값은 880이다. 이것은 4%에서 84%까지 다양한 기준 결합도를 이용하여 서비스의 크기를 다양하게 조절할 수 있음을 의미한다.

2) 기존 컴포넌트를 그대로 서비스로 사용하였을 경우의 결합도 분석

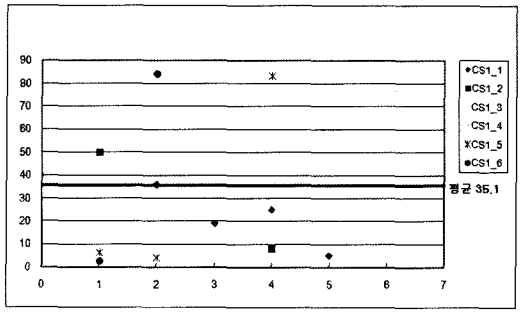
본 실험에 사용된 13개의 컴포넌트로 구성된 경영 정보 시스템의 컴포넌트 각각을 1차 후보 서비스로 식별하였을 때의 결합도를 분석하였다. 그 결과는 표 7과 같다.

표 7에서 1차 후보 서비스 식별 시 결합도의 값들은 최대, 최소값을 제외하면, 최소 5%에서 50%까지의 값에 집중되어 있음을 볼 수 있다. 또한, 분산은 173이다. 표 7을 이용하여 정의 2, 3과, 포함관계를 적용하여 서비스를 식별하면 그림 9와 같은 9개의 2차 후보 서비스가 식별된다. 서비스 식별 결과 서비스의 크기는 기존 컴포넌트의 크기와 변화가 거의 없다.

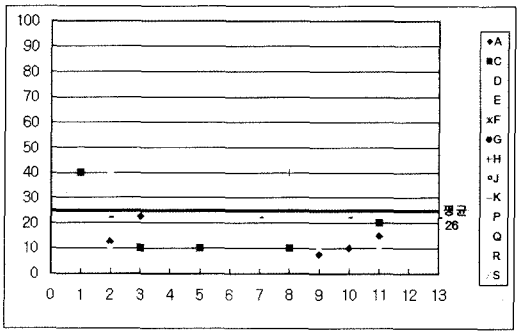
그림 9 기존 컴포넌트의 서비스 식별

3) 실험 목적 1의 결과 분석

경영 정보 시스템을 본 논문에서 제안한 방식과 기존 컴포넌트를 그대로 서비스로 식별하였을 때의 서비스 크기를 고려한 서비스 식별 결과를 분석한 결과는 그림 10과 같았다. 그림 10(a)는 본 논문에서 제안한 방식을 적용하여 13개의 GUI컴포넌트를 6개의 서비스로 식별하였을 때의 결합도 분포이고, 그림 10(b)는 기존의 컴포넌트를 그대로 서비스화 하였을 때의 결합도 분포이다. 본 논문에서 제안한 방법으로 식별된 서비스간 결합도는 평균을 기준으로 결합도의 분포가 고루 퍼져 있고,



(a)



(b)

그림 10 결합도 분석 (분산값)

분산값은 880이다. 반면 컴포넌트를 바로 서비스로 식별한 경우는 평균에 가깝게 결합도가 분포되어 있고, 분산값은 173이다.

즉, 그림 10(b) 그래프의 컴포넌트를 그대로 서비스화하는 경우와 같이 평균에 밀접하게 결합도가 분포되어 있는 경우, 기준 결합도를 정하는 데 있어 다양한 값을 정할 수 없고 이것은 곧 어떠한 컴포넌트를 서로 컴포지션하여 새로운 서비스로 식별할 것인가에 대한 판단

표 8 최종 서비스 식별시 발생하는 문제점 분석 결과

연관관계에 따른 문제점 발생 유형		본 논문 제시 방법으로 서비스 식별		컴포넌트의 서비스식별	
		최초 문제점 수	식별후 문제점 수	최초 문제점 수	식별후 문제점 수
한 후보서비스의 Root에서 다른 후보서비스의 Root를 호출한 경우	Retrieval	20	7	4	1
	Update	7	1	0	0
한 후보서비스의 Root에서 다른 후보서비스의 Subset을 호출한 경우	Retrieval	6	6	9	9
	Update	1	0	1	1
한 후보서비스의 Subset에서 다른 후보서비스의 Root를 호출한 경우	Retrieval	0	0	6	6
	Update	0	0	1	1
한 후보서비스의 Subset에서 다른 후보서비스의 Subset을 호출한 경우	Retrieval	0	0	6	6
	Update	0	0	5	2

기준이 모호함을 의미한다. 그러나 그림 10(a) 그래프에서 보는 바와 같이 본 논문에서 제시한 방법으로 서비스를 식별할 경우, 분산 880으로, 컴포넌트를 바로 서비스로 식별할 때보다 더 다양하게 기준 결합도를 정할 수 있고, 모호함이 없이 다양한 크기로 서비스 식별이 가능함을 알 수 있다.

이것은 최종서비스 식별 시 관련된 데이타베이스와 같은 자원을 어떻게 나눌지에 대한 문제를 일으키기 때문이다. 표 8의 결과, 기존 컴포넌트를 바로 서비스로 식별하는 경우는 Retrieval 19%, Update 43%, 총 18.75%의 문제점만을 해결한데 반해, 본 논문에서 제시한 방법으로 서비스를 식별하는 경우 처음과 최종 서비스 식별 후에 Retrieval 50% Update 87.5%로 총 60%의 문제점을 해결함을 볼 수 있다.

특히, Update에서의 본 논문에서 제시한 방법으로 식별했을 때 무려 40% 넘게 효과적으로 문제점을 해결함을 볼 수 있다.

실험목적2: 최종 서비스 식별시 발생하는 문제점 최소화

본 논문에서 제시한 방법으로 최종 서비스 식별할 때와 기존 컴포넌트를 그대로 서비스로 식별할 때 발생하는 문제점 수를 나타낸 결과는 표 8과 같다. 각 연관관계에 따른 문제점 발생 유형은 검색이나 단순 내용 보기 등의 Retrieval유형과 수정, 변경, 삭제 등 데이타베이스의 변화를 가져오는 Update로 나뉘는데 문제가 되는 것은 Update이다.

4.3 실험결과 분석

본 논문에서 제시한 상향식 서비스 식별법을 이용하면 CoreRoot를 설정할 때의 α값과 포함관계를 구할 때의 β값을 이용하여 기존 컴포넌트를 바로 서비스화 할 때에 비해 식별에 대한 판단기준을 명확하면서 서비스의 크기를 융통성 있게 조절, 적당한 크기를 갖는 약결합된 서비스를 식별해 낼 수 있고, 최종 서비스 식별시 발생되는 연관관계에 따른 여러 가지 유형의 문제점들을 효과적으로 해결함을 보여준다. 따라서 본 논문에서 제시한 방법은 GUI의 이벤트를 고려하여 사용자의 행위를 모델링하고 그 이벤트가 두 서비스간의 결합도에 가중치로 작용하여 서비스의 크기를 조정할 수 있는 기초를 마련한 것이다.

5. 결론 및 향후 연구

본 논문에서는 GUI와 연계된 컴포넌트 시스템을 대상으로 서비스 지향 아키텍처를 위한 크기를 고려한 상향식 서비스 식별 방법을 제안하였다. 제안된 서비스 식별 과정은 GUI 화면을 분석하여 이벤트 정보를 추출하고 GUI 이벤트 흐름도를 작성하는 과정인 전처리 과정과 순차이벤트의 중복성 발견을 위한 알고리즘을 적용하여 후보 서비스를 식별하고, 정제하는 서비스 식별 및 정제 과정으로 이루어진다.

본 논문에서 제안한 서비스 식별 방법은 후보 서비스를 식별, 정제하는 과정에서 후보 서비스간의 포함관계 및 연관관계를 최소화함으로써 서비스의 유지보수성 및 재사용성을 높이고, α, β와 같은 서비스 식별에 사용된 변수의 값을 조절함으로써 적당한 크기를 갖는 약결합된 서비스로 식별 가능하게 해 준다.

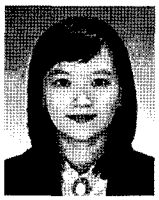
향후, 본 논문의 실험에서 보여주지 못했던 서비스의 크기를 조절하는 두 가지 변수, CoreRoot를 설정할 때의 α값과 포함관계를 구할 때의 β값의 다양한 크기 조절을 보완하여 최적의 Threshold값을 산출할 수 있는 실험을 추가 실시하여 결과를 산출할 것이다.

참 고 문 헌

[1] Dirk Krafzig, Karl Banke, Dirk Slama, "Enterprise SOA," 2005, Pearson Education, Inc.

[2] Gregg Kreizman, "How to Build a Business Case for Service-Oriented Development of Applications in Government," Gartner. Industry Research, 2005.9.

[3] Kunal Mittal, "Service Oriented Unified Process (SOUP)," IBM Journal, 2005.6.
[4] Ali Arsanjani, "Service-Oriented Modeling and Architecture : How to identify, specify, and realize services for your SOA," IBM developerWorks, 2004.11.
[5] Ash Parikh, Rajesh Pradhan and Nirav Shah, "Modeling of Web Services : A Standards-Based Approach," Software Magazine, 2004.5.
[6] keith Levi, Ali Arsnaiani, "A Goal-driven Approach to Enterprise Component Identification and Specification to Enterprise Component Identification and Specification," Communications of The ACM, Vol.45, No.10. 2002.
[7] Bras, L., P. Clements and R. Kazman, "Software Architecture in Practice," Addison-Wesley, 1998.
[8] V. Kapoor, "Services and Automatic Computing: A Practical Approach for Designing Manageability," In Proceedings of the 2005 IEEE International Conference on Service Computing(SCC'05), Vol.2, pp. 41-48, 2005.
[9] Thomas Earl, "Service-Oriented Architecture :Concept, Technology, and Design," Prentice hall, 2005.
[10] V. B. Misic, S. Moser, "Measuring Class Coupling and Cohesion : A Formal Metamodel Approach," APSEC'07, pp. 31-40, Dec.1997.
[11] Chidamber S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Software Engineering, Vol.20. pp. 476-498, 1994.
[12] Hyung Ho Kim and Doo Whan Bae, "Component Identification via Concept Analysis," Journal of Object Oriented Programming, 2001.
[13] J.W.Lee, K.Lee, and W.Kim, "Preparations for Semantics-based XML Mining," In Proc. of IEEE *International Conference on Data Mining (ICDM* '01), pages 345~352, Nov./Dec.2001.



이 현 주

2005년 8월 이화여자대학교컴퓨터학과(학사). 2007년 8월 이화여자대학교컴퓨터정보통신공학과(공학석사). 관심분야는 서비스지향아키텍처(SOA), 소프트웨어공학

최 병 주

1986년~1987년 Purdue Univ. Computer Science 석사. 1987년~1990년 Purdue Univ. Computer Science 박사. 1991년~1992년 삼성종합기술원. 1992년~1995년 용인대 전산통계학과 조교수. 1995년~현재 이화여대 컴퓨터학과 교수. 관심분야는 소프트웨어공학, 소프트웨어 테스트, 소프트웨어 및 데이타 품질 측정

이 정 원

1993년 이화여자대학교 전자 계산학과 학사. 1995년 이화여자대학교 대학원 전자계산학석사. 1995년~1997년 LG종합기술원 연구원. 1997년~2003년 이화여자대학교 컴퓨터학박사. 2003년~2006년 이화여자대학교 컴퓨터학과 BK교수, 전임강사(대우). 2006년~현재 아주대학교 정보통신대학 전자공학부 조교수. 관심분야는 SOA, 유비쿼터스컴퓨팅, 임베디드 소프트웨어