레거시 시스템으로부터 SOA의 서비스 추출 방법

정현호*, 이상범*
*단국대학교 컴퓨터과학과
e-mail:ilhyunli@nate.com

An Extraction Method of SOA Service from Legacy System

Hyun-Ho Jung*, Sang-Bum Lee*
*Dept. of Computer Science, Dankook University

요 약

기업의 비즈니스 시스템은 수시로 변화하는 고객의 요구사항과 새로운 기반기술의 출현 등에 따라 끊임없이 변화하고 있다. 따라서 기업은 비즈니스 환경변화에 유연하고 신속하게 대응할 수 있는 환경을 필요로 하고 있다. SOA는 이러한 요구에 적합하도록 설계된 아키텍처이다. SOA(Service Oriented Architecture)는 비즈니스를 세분화하여 반복 가능하고 가치를 지닌 단위 서비스들로 나누어 조립과 통합을 통해 새로운 가치를 만들어 낼 수 있도록 지원한다. 하지만 비즈니스 시스템의 많은 부분을 차지하던 Legacy 시스템에 이러한 아키텍처를 적용하기에는 많은 어려움을 가지고 있다.

본 논문에서는 Legacy 시스템으로부터 SOA의 서비스를 추출하는 방법을 제안한다. 서비스를 추출하는 과정은 3가지 단계를 거친다. 첫 번째 단계는 COBOL로 제작된 Legacy 시스템의 프로그램에서 DATA DIVISION으로부터 변수를 분류 및 그룹화하고 PROCEDURE DIVISION의 루틴간의 연관관계를 파악하여 객체화시킨다. 두 번째 단계에서는 첫 번째 단계에서 얻어진 변수들 중 핵심 변수를 식별하여 객체화를 통해 식별된 로직에서 서비스가 될수 있는 로직을 식별한다. 마지막으로 세 번째 과정은 식별된 서비스와 데이터를 사용하여 SOA 서비스를 구축한다.

1. 서론

오늘날 기업의 정보 시스템들은 수시로 변화하는 고객의 요구사항을 반영하고 새로운 IT 기반기술을 적용시키기 위해 많은 노력을 하고 있다. 하지만 기업의 비즈니스 시스템에서 중요한 업무처리를 담당하던 기존의 레거시(Legacy) 시스템들은 이러한 비즈니스 환경변화에 유연하고 신속하게 대처하는데 많은 어려움을 가지고 있다.

Legacy 시스템은 기업에서 장기간 운영해온 시스템을 말하는데, 이러한 Legacy 시스템들은 기업 운영에 필요한 핵심적인 비즈니스 로직을 포함하고 있으며 오랜 기간의 축적된 경험과 지식으로 입증된 안정성과 신뢰성을 갖고 있다. 하지만 오래전에 개발된 Legacy 시스템 대부분은 COBOL이나 PL/I와 같은 프로시져 언어로 구현되어 있으며, 완전하지 못한 시스템 문서화와 프로그램 전문가의 부족, 복잡하고 이해하기 어려운 코드 등으로 인해 그것들을 유지보수 하는데 상당한 노력이 요구되고 있다. 또한 이러한 Legacy 시스템들은 비즈니스의 변화에 따른 추가 요구사항 수정 및 보완작업 등의 과정으로 인한 높은 소요비용으로 기업에 있어 큰 문제로 남아있게 되었다.

2000년대 이후 인터넷의 확산과 더불어 이러한 Legacy 시스템이 가지는 가치를 재사용 혹은 현대화 하기위해 많은 논의가 이루어지고 있다. 특히 Legacy 시스템을 래핑(Wrapping)이나 변환(Transforming) 등의 재공학을 통해 컴포넌트 기반 환경이나 웹 환경, 분산 환경으로 이주하는 방법론에 대한 논의가 많은 부분을 차지하고 있다[1].

최근에는 기업 인프라의 복잡성 및 유지비용을 최소화 하고, 기업의 생산성과 유연성을 극대화 할 수 있는 방안으로 서비스 지향 아키텍처(Service Oriented Architecture)가 이슈로 떠오르고 있다.

본 논문에서는 최근 이슈인 SOA에 대해 설명한 후 기존 Legacy 시스템으로부터 SOA의 서비스를 추출하는 방법에 대해 제안한다. SOA의 서비스 구성요소 중 인터페이스 부분은 이를 추출하는 방법이 서비스를 추출하는 방법과 비슷하지만 비즈니스 시스템의 기반환경에 영향을 받기 때문에 본 논문에서

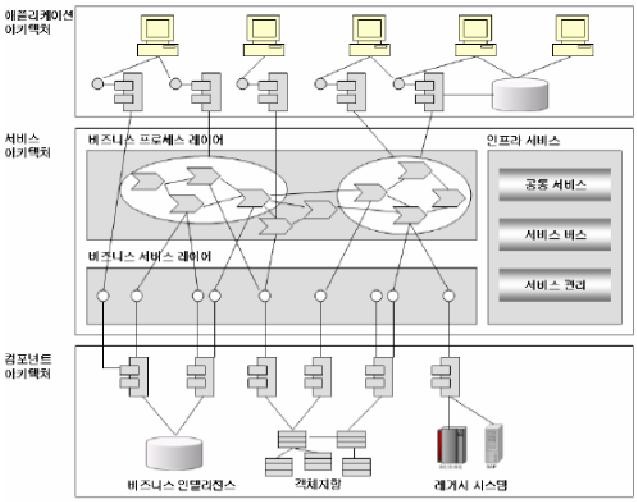
는 다루지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스 지향 아키텍처(SOA)에 대해 간략히 살펴보고, 3장에서는 Legacy 시스템으로부터 서비스를 추출하기 위한 과정을 살펴본다. 그리고 4장에서는 결론과 향후 연구과제에 대해 설명할 것이다.

2. 서비스 지향 아키텍처(SOA)

2.1. SOA 정의

서비스 지향 아키텍처(Service Oriented Architecture)는 [그림 1]과 같이 분산 객체의 형태로 존재하는 비즈니스 단위의 컴포넌트들이 메시지 형태의 정보를 교환하는 느슨하게 연결된(loosely coupling) 형태의 소프트웨어 아키텍처이다. 이러한 SOA는 비즈니스와 밀접한 관계를 가지고 있으며, 비즈니스와 기술을 통합하여 효율적인 시스템을 구축하는데 목적을 가지고 있다[2,3].

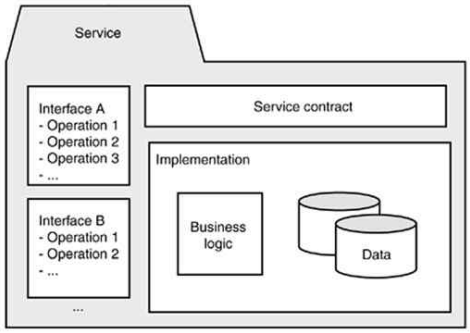


[그림 1] SOA 기반 환경

2.2. SOA 서비스

서비스 지향 아키텍처(SOA)를 구축하는 서비스는 계약, 하나 이상의 인터페이스, 그에 대한 구현으로 이루어지며 명확한 기능적 의미를 지닌 소프트웨어 컴포넌트로, 고차원의 비즈니스 로직을 캡슐화 한다. 또한 서비스는 인터페이스를 통해 자신이 가진 비즈니스 프로세스를 처리할 수 있는 컴포넌트로 정의되며 아래 [그림 2]와 같이 인터페이스와 구현 부분으로 구성된다[4].

이러한 SOA의 서비스는 재사용 할 수 있는 로직 단위로 분리되어야 하고, 다른 서비스와 조립 가능해야 하며 서비스 로직이 정확한 경계 안에 존재하여 자율성을 가져야 한다.

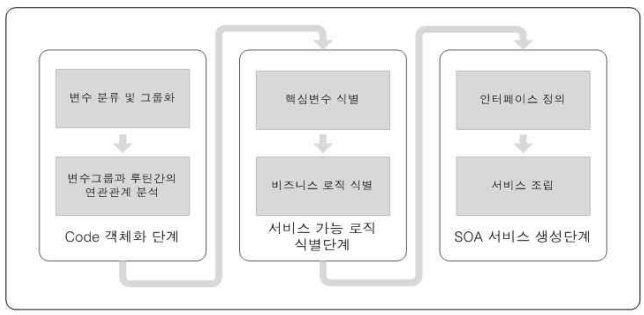


[그림 2] SOA 서비스의 구성

본 논문은 Legacy 시스템으로부터 이러한 서비스를 추출하는데 목적을 가진다.

3. 서비스 추출과정

COBOL 언어로 개발된 전형적인 Legacy 시스템으로부터 SOA의 서비스를 추출하는 과정은 아래의 [그림 3]과 같이 소스코드를 객체화 하는 단계와 서비스를 식별하는 단계, 서비스를 생성하는 단계로 나뉘며 이를 통해 최종 서비스를 추출한다.



[그림 3] 서비스 추출과정

3.1. 코드 객체화 단계

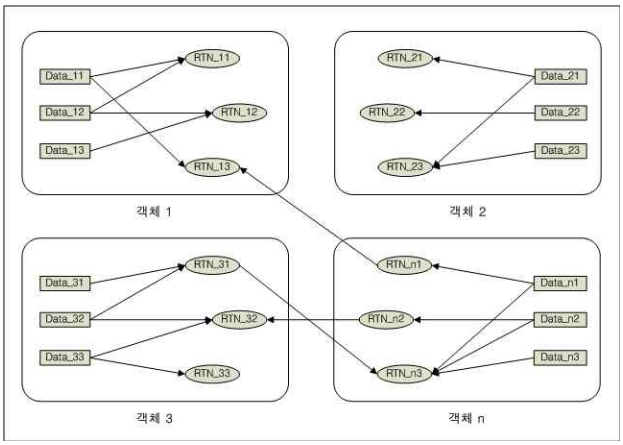
코드 객체화 단계는 COBOL언어로 작성된 소스코드로부터 변수를 분류 및 그룹화 하는 과정과 변수 그룹과 루틴간의 연관관계를 분석하는 과정으로 나뉜다.

COBOL 언어로 작성된 소스코드는 구조적으로 4개의 DIVISION으로 나누어지는데 [그림 4]와 같이 DATA DIVISION에는 변수 정보와 인터페이스 정보가 기술 되어있다. 또한 PROCEDURE DIVISION에는 변수의 처리방법과 순서가 기술되어 있으며, 실제 명령문을 통해 처리 로직이 구현되어 있다.

[그림 4] COBOL 구조

변수 분류 및 그룹화과정은 사전에 정의된 분류 그룹에 따라 DATA DIVISION의 변수들을 그룹화 하는 것이다. Joiner[5]는 영역변수, 입력변수, 프로그램변수, 지역변수, 전역변수, 출력변수, 제약조건 변수, 제어변수와 같이 8개의 변수 형태를 정의하고 그룹화 하였다. 이러한 과정은 PROCEDURE DIVISION의 루틴에서의 호출정보를 통해서 이루어진다.

연관관계 분석과정을 위해 위의 분류된 변수들과 PROCEDURE DIVISION의 루틴이 필요하며, 루틴 실행에 따른 변수의 호출을 통해 변수와 루틴간의 연관관계를 파악할 수 있다. 이를 통해 아래의 [그림 5]와 같이 루틴을 그룹화 할 수 있으며 이렇게 그룹화 된 루틴과 변수는 객체로 식별할 수 있다.



[그림 5] 객체 연관 그래프

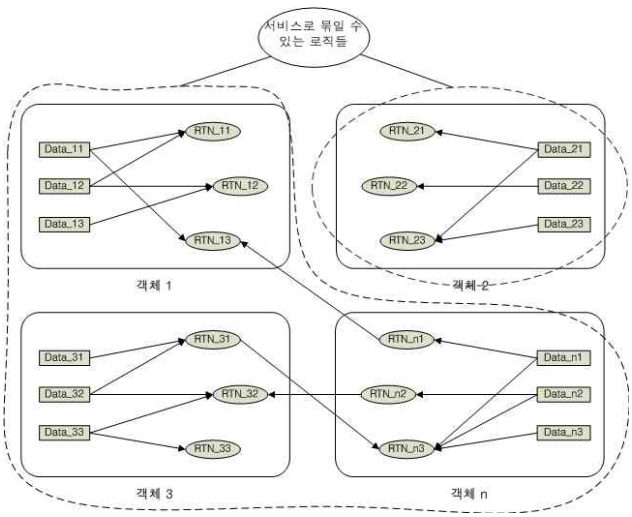
위의 과정을 거치면서 불필요한 변수 및 루틴을 제거할 수 있으며 핵심 변수를 식별 가능하게 된다. 한 루틴에만 호출되는 변수는 해당 루틴의 지역변수가 되며, 여러 루틴에게 호출되는 변수는 전역변수가 된다.

3.2. 서비스 가능 로직 식별단계

서비스 가능 로직 식별단계에서는 코드 객체화 단계를 통해 그룹화 된 변수들과 추출된 객체들의 연관정보를 통해 핵심 변수를 식별하는 과정과 서비스가 될 수 있는 비즈니스 로직을 식별하는 과정으로 나누어진다.

핵심 변수를 식별하는 과정은 이전 단계를 통해 식별된 변수를 후보로 뽑아낼 수 있는데 이중에서 핵심적 가치를 가지는 변수를 찾아야 한다. 핵심적 가치를 가지는 변수는 루틴으로 이루어진 객체에서 가장 밀접한 관계를 가지는 변수가 된다. 이러한 변수들은 주로 입출력 변수가 된다.

서비스가 될 수 있는 비즈니스 로직을 식별하는 과정에서 비즈니스 로직은 업무에 필요한 데이터 처리를 수행하는 응용프로그램의 일부를 말하는데 이것은 주로 데이터 입력, 수정, 조회 및 보고서 처리 등을 수행한다. 서비스는 하나 이상의 비즈니스 로직으로 이루어지며, 핵심변수를 다루는 로직이 포함되어야 한다.



[그림 6] 서비스 가능로직 식별

위의 [그림 6]과 같이 코드 객체화 단계의 연관관계를 통해서 식별된 객체들 중 핵심 변수를 포함하는 루틴 혹은 객체간의 관계를 통해서 서비스의 로직이 될 수 있는 객체들의 범위를 구할 수 있다.

3.3. SOA 서비스 생성단계

SOA 서비스 생성단계는 인터페이스를 정의하는 과정과 서비스를 조립하는 과정으로 나누어진다. SOA 서비스의 인터페이스는 WebService, RMI, RPC, SOAP, XML-RPC, TP connection, EJB Call, Message Queue 등의 인터페이스 기술로 구현되며 네트워크를 통해 서비스에 연결한 클라이언트에게 서비스 기능을 제공한다. 인터페이스를 추출하는 내용은 서비스를 추출하는 내용과 비슷하지만 비즈니스 시스템의 기반환경에 영향을 받기 때문에 본 논문에서는 다루지 않도록 한다.

서비스를 조립하는 과정은 이전의 단계들을 통해서 추출된 데이터와 서비스 로직, 그리고 인터페이스를 하나의 컴포넌트로 만드는 과정이다.

4. 결론 및 향후 연구 과제

본 논문에서는 COBOL언어로 개발된 Legacy 시스템으로부터 SOA기반 환경으로 변환시 필요한 서비스 추출 방법을 제안하였다. 이런 방법을 사용하면 COBOL 프로그램으로부터 쉽게 코드를 객체화 할 수 있으며 이를 통해 SOA의 서비스로 변환이 가능하다.

하지만 이렇게 추출된 서비스는 기능적인 면보다 재사용성과 독립성 등을 중점으로 고려되었기 때문에 추출된 서비스가 제대로 기능하는지 확인이 필요하다. 또한 추출과정 중에서 객체화 과정은 자동화가 가능하지만 서비스를 식별하는 과정에서는 서비스가 올바로 정의되었는지 사용자에 의한 개입이 필요하게 된다.

향후 연구과제로는 서비스를 사용자의 개입이 없어도 올바르게 식별하도록 하는 방법과 추출된 서비스에 관한 검증 및 테스트에 관한 연구, 그리고 COBOL 프로그램 이외의 언어에 대해서도 적용 가능한 추출 기법에 대한 연구가 이루어져야 한다.

참고문헌

[1] 김철홍외 2명, “A Re-engineering Methodology for Componentization of Legacy System", 한국IT서비스학회지, 2003

[2] OSOA(Open Service Oriented Architecture), " http://www.osoa.org"

[3] Thomas Erl, "Service-Oriented Architecture : Concepts, Technology, and Design", Prentice Hall, 2005.

[4] Dirk Krafzig, Karl Banke, Dirk Slama, “Enterprise SOA: Service Oriented-Architecture Best Practices”, Prentice Hall PTR, 2004

[5] Joiner J.K, Tsai W.T, Chen X.P, Data-Centered Program Understanding, Proc. Software Maintenance, 1994, 272-281.