트랜잭션 어댑터 컴포넌트를 이용한 레거시 시스템의 랩핑에 관한 연구

황 선 명[†] · 김 상 영[††] · 김 정 아[†††] · 진 영 택[††††]

요 약

컴포넌트 기반 소프트웨어 설계는 개발된 컴포넌트의 조립을 통한 재사용으로 소프트웨어를 생성하는 것을 목표로 하고 있다. 이때 재사용되는 컴포넌트들은 용도에 맞게 개조되어야 한다. 본 연구는 이러한 개조 방법을 트랜잭션 어댑터(TA : Transaction Adaptor)라는 개조 컴포넌트를 이용하여 기존의 컴포넌트 또는 기존의 레거시 시스템을 재사용 하는 것에 대한 연구이다. TA를 이용한 개조방법은 클라이언트와 호스트 시스템 사이에 TA 컴포넌트를 사용하여, XML데이터를 스트림 형태로 변환하여 전송함으로서 레거시 시스템을 재사용 한다. 또한 응용 프로그램의 개발에 XML 데이터에서 필요로 하는 정보로 가공하는데 룰서버(RS : Rule Server)를 이용함으로서 개발과 유지보수에 많은 효과를 얻을 수 있다. 이러한 TA와 RS를 이용한 재사용 방법은 클라이언트 플랫폼이나 호스트의 종류에 관계없이 TA가 XML로 데이터 변환처리하고, RS가 데이터의 처리를 수행하여 레거시 시스템을 재사용할 수 있다.

A Study on Wrapping of Legacy System using a Transaction Adaptor Component

Sun-Myung Hwang[†] · Sang-Young Kim[††] · Jung-Ah Kim[†††] · Young-Tak Jin[††††]

ABSTRACT

The purpose of CBSD(Component Based Software Development) is to develop software using a reuse component and components assembly. The reused components have to be adapter for satisfaction of requirement specification. This paper represents a component adaption method which reuse previously developed component or legacy systems, using an adaptation component called by TA(Transaction Adaptor). The adaptation using TA transmits XML data that is translated into stream type, from a client to on host system. And we introduce RS(Rule Server) which gets some information from XML data for application program development. Through the interaction of TA and RS, we can improve effectiveness of component development and maintenance and we can reuse legacy system.

키워드 : CBSD, 컴포넌트(Component), 개조(Adaptation), 재사용(Reuse), 트랜잭션 어댑터(Transaction Adaptor), 룰서버(Rule Server)

1. 서 론

CBSD(Component Based Software Development) 기반 어플리케이션 개발은 처음부터 소프트웨어를 개발하기보다는 컴포넌트를 선택(Selection), 개조(Adaptation) 및 조립(Assembly)의 과정을 통해 이루어진다. 컴포넌트를 단순하게 본다면 컴포넌트는 어플리케이션 플러그인(Plug-in) 방식으로 활용되어 "있는 그대로(As-is)방식"으로 재사용(Black-Box Reuse)될 수 있다[1]. 그러나 많은 연구에서 밝혀진 바와 같이 "있는 그대로 재사용"이란 매우 어렵고 대부분의 어플리케이션 개발 과정에서 실현성이 없는 것으로 보고되고 있다. 즉 컴포넌트가 정의하고 있는 인터페이스와 이를 사용하고자 하는 컴포넌트 또는 다른 어플리케이션의 요구사항이 다른 경우 컴포넌트는 어떤 방식으로든 개조가 필요한 것이 일반적이다. 조립시에 나타나는 가장 빈번한 문제는 구입한 컴포넌트의 인터페이스가 현 어플리케이션 개발에서 요구되는 인터페이스와 다르거나, 인터페이스는 동일하나 실제의 행위가 요구하는 기능과 상이한 경우이다[2]. 이러한 문제점을 극복하는 방법으로는 통합할 때 기존의 컴포넌트나 시스템을 변경하거나 구입한 컴포넌트를 수정하거나 기존의 시스템과 컴포넌트 사이에 개조기(Adaptor)를 두어 해결하는 것이 일반적이다.

※ 본 연구는 한국과학재단 목적기초연구(R01-2001-000-00343-0(2003))지원으로 수행되었음.
† 종신회원 : 대전대학교 컴퓨터공학과 교수
†† 준 회 원 : 대전대학교 대학원 컴퓨터공학과
††† 종신회원 : 관동대학교 컴퓨터교육과 교수
†††† 종신회원 : 한발대학교 정보통신·.컴퓨터공학부 교수
논문접수 : 2004년 2월 27일, 심사완료 : 2004년 4월 20일

컴포넌트를 개조해야 한다는 관점으로 본다면 기존의 재사용에서의 재사용 단위를 개조하는 것과 같은 맥락으로 볼 수 있을 것이다. 기존의 재사용 단위의 개조는 상속이나 복사를 통한 수정과 같은 White-box 수정과 래핑(wrapping)기법과 같은 Black-box 수정 방식으로 구분할 수 있다.

2. 관련 연구

2.1 CBD 아키텍처

CORBA, DCOM, EJB 등의 시스템 기반 공용 컴포넌트의 보편화된 사용과 특정 핵심 영역에서 비즈니스 컴포넌트의 도입이 크게 증가함에 따라 컴포넌트 기반의 소프트웨어 개발(CBD : Component Based Development)은 대부분의 비즈니스 시스템 개발에서 최상의 경쟁력을 갖춘 전략적 방법론으로 인식되고 있다. 실제로 현재의 분산 응용 시스템들은 대부분 3계층에 기반하고 있어, 개발과 운영에서의 개별 개체들의 독립성과 상호운영성 및 동적 변화에 대한 민첩한 대응성 확보를 위해 컴포넌트 기술의 필요성이 더욱 증대되고 있다. 특히, 표준 플랫폼으로써 인터넷과 이와 관련된 웹 서비스의 성장 및 Java 중심으로의 구축 환경의 변화는 CBD의 핵심을 종래의 기능적 로직을 수행하는 컴포넌트의 생성을 위한 재사용 관점의 재사용 컴포넌트 개발에서 사용하고 공급해야만 하는 서비스가 무엇이며 어떻게 이를 가능하게 할 것 인지의 컴포넌트에 의한 재사용에 대한 관심으로 이동시키고 있다[22].

따라서 인터넷 상의 다양한 소프트웨어 부품 즉, 컴포넌트들을 쉽고 기종에 상관없이 사용하고 "Plug and Play" 할 수 있는 기술 및 지원 환경은 CBD의 가장 큰 기능으로서 필연적으로 요구된다. 즉, 소프트웨어 개발에서 모든 단위 모듈을 부품화함으로써 소프트웨어의 산업화와 대량화를 통한 재사용의 실현을 위한 정점으로 "컴포넌트"가 중심이 되며 이미 형성되어지고 있는 컴포넌트 시장에서의 활용 극대화를 통한 긍정적인 순환을 유도를 위한 기술이며 사업적으로 일치된 합의가 이루어지고 있다.

현재, 이슈가 되고 있는 컴포넌트 기술이나 이전의 객체지향 및 3R(Repository, Reengineering, Reuse), 도메인 공학 모두 "소프트웨어 재사용"이라는 고전적인 문제를 현실화하기 위한 접근들이다. 하지만, 특정 개발 단계에 국한된 재사용 범위나 화이트 박스(White Box) 기반의 상호 의존성 유발, 그리고 이질적인 도메인과 플랫폼에 대한 통합의 어려움 등으로 각 기술들이 제시하고 기대하던 효과 창출을 실현시키지는 못하였다. 따라서 컴포넌트 기술 역시 웹 기술의 절대적 보급에 따른 구현과 공급이 분리된 프로세스를 지향하고 있지만, 불투명한 모델링과 명세 정보와 이로부터 발생되는 컴포넌트의 이해와 적용이라는 필수적인 단계로의 전환이 어려운 상태이다.

CBD는 컴포넌트의 선택과 획득 그리고 통합에 중점을 두어야 한다. 하지만 컴포넌트 관련 기술의 실행에 있어 가장 큰 어려움은 생산된 컴포넌트의 활용 기술 즉, 통합 문제이다. CBD 시스템에서 요구되는 컴포넌트의 서비스들이 무엇이며 이들이 어떠한 관련성으로 통합되어져 공통 도메인에서 재사용 되어질 수 있는가를 설명할 수 있는 표준적인 가이드라인이나 환경적 지원 기술은 실용적인 측면에서 거의 전무한 상태이다. 대부분의 CBD 시스템들은 비즈니스 솔루션을 갖는 단일 개의 컴포넌트만이 시스템의 기능적 모듈로서 이용할 뿐 컴포넌트의 조립이나 서비스 인스턴스의 공유는 여전히 간과되는 실정이다.

따라서 새로운 시스템 개발을 위해 보편화된 구조성과 이에 포함된 요소들 간의 행위적인 관련성에 대한 청사진을 제시하는 소프트웨어 아키텍처 기술은 CBD의 진보적인 성장을 위해 적용되어져야만 한다. 특히 컴포넌트의 수직적, 수평적 위상에 대한 지침과 연관 컴포넌트들 간의 관련성을 제시함으로써 컴포넌트의 활용을 위한 디폴트 프로세스를 형성하는 컴포넌트 아키텍처는 개별 컴포넌트의 결합성에 관한 관점이 아니라 소프트웨어 아키텍처에서 정립된 기술들을 도입하며 CBD 적용 시스템의 아키텍처 기술로 확장되어져야만 한다. 이러한 접근 기술은 기존 컴포넌트 아키텍처와 소프트웨어 아키텍처가 결합되어져 서비스 지향의 통합과 구축의 자동화 환경을 제시하는 컴포넌트 프레임워크로 가능하다[21].

컴포넌트 프레임워크는 인터페이스와 이벤트, 확장 가능한 부분 등을 포함하여 컴포넌트를 정의하고 컴포넌트의 실행과 전개의 수단이 되는 컴포넌트 모델을 기반으로, 동적인 결합이 가능하도록 인터페이스의 집합과 상호 규칙들을 비즈니스 도메인에 따라 다양성과 공통성의 관점에서 아키텍처를 바탕으로 분류, 지원하는 통합 환경이다.

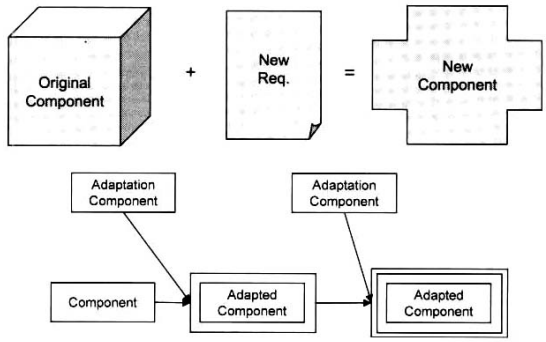
2.2 기존의 개조기법

본 연구에서의 개조란 컴포넌트가 어떤 이유로든 미리 정의된 인터페이스나 행위를 변경해야 할 경우를 의미한다. 간단하게는 인터페이스의 이름이 변경되거나 파라미터의 형식이 변경되는 것을 들 수 있고, 복잡하게는 새로운 인터페이스가 추가되는 경우도 볼 수 있다.

컴포넌트 개조의 개념은 (그림 1)과 같으며 일반적으로 다음의 방법으로 가능하다[5, 6].

① 개조기(Adaptor)기법 : 인터페이스의 차이를 보이는 두 컴포넌트 사이에 개조기 패턴을 활용하여 개조기 컴포넌트를 정의하여 해결한다. 이 개조기 컴포넌트가 차

이나는 두 컴포넌트 사이의 인터페이스 불일치를 해결한다.



(그림 1) Component Adaptation 기본 개념

② Wrapper기법 : 이는 인터페이스의 차이를 없애는 새로운 컴포넌트를 정의하고 이 컴포넌트가 기존의 컴포넌트를 포함하여 관리하는 방식이다[3, 4]. 지금의 바이너리 컴포넌트 기술에 있어서 가장 많이 사용하는 기술로서 Wrapping 컴포넌트는 아주 적은 변경이 필요한 경우에 생성되고, 기존의 컴포넌트에 정의된 기능이 필요하면 요구를 전달하기만 한다. 즉 Wrapping은 이름의 변경 또는 파라미터의 변경 등의 간단한 변경에 있어서 적용 가능하다. 그러나 빈번한 개조가 일어날 경우 개조된 컴포넌트의 크기가 기하 급수적으로 커질 수 있다는 단점이 있다.

③ Superimposition에 의한 개조 : 기본 개념은 개조를 해야하는 컴포넌트와 개조를 처리하는 컴포넌트 두 개를 분리하되, 둘을 하나로 묶어서 긴밀하게 동작시키도록 하려는 것이다[5]. 이는 개조의 대상이 되는 컴포넌트와 개조를 처리하는 컴포넌트 모두 독립적으로 재사용하려는 목적을 갖는다. 이 방법은 컴포넌트의 개조 유형이 다양하지 않고 정해진 범위 내에서 이루어진다는 전제 하에 가능한 방법이다. 이를 위해 언어를 처리하는 새로운 모델을 제시하였으며, 이 기법의 지원을 위해서는 기존의 언어에서 이루어지는 메소드 호출을 별도로 처리하는 기반 시스템을 구축해야 하기 때문에 일반적이지 못하다.

④ Binary Adaptation 기법 : UCSB(University of California at Santa Barbara)에서 개발한 방법으로 동적 로더(Dynamic Loader)를 개발하여, 로딩된 바이너리 컴포넌트를 직접 수정하는 방식이다[6, 7]. 이를 위하여 Java의 클래스 로더를 수정하여 UC Santa Barbara 대학에서 개발한 Delta file 컴파일러를 사용해야 한다는 제약점이 있다. 즉 일반적인 환경에서는 사용 불가능하다. 이러한 개념이 자바에서는 가능하나, EJB(Enterprise JavaBeans)컴포넌트의 경우는 로더를 변경하는 것이 명세에 금지되어 있으므로 EJB에 바로 적용하기는 어려운 기법이다.

⑤ 컴포넌트 자체를 수정하는 기법 : 바이너리 컴포넌트로부터 컴포넌트의 reflection 기법을 이용하여 컴포넌트 소스코드를 역으로 추출 한 후 컴포넌트 자체를 수정하여 새로운 컴포넌트를 만들어내는 방법이다.

이러한 컴포넌트 개조 기법이 만족해야 하는 사항을 정의하면 다음과 같다[8].

① Transparent : 컴포넌트가 Transparent 하다는 것은 개조 전의 원 컴포넌트(Original Component)를 사용하는 개발자나 개조후의 컴포넌트를 사용하는 개발자 모두에게 개조의 사실이 알려져서는 안된다는 것이다. 뿐만 아니라 개조에 필요 없는 컴포넌트는 당연히 추가적인 노력 없이 접근 가능해야 한다.

② Black-Box : 소프트웨어 개발자는 컴포넌트를 재사용하기 전에 컴포넌트에 대한 기능성을 충분히 파악해야 하지만, 가능하면 최소한의 정보만으로 가능해야 한다. 이러한 요구 사항은 컴포넌트의 개조과정에 원 컴포넌트가 어떻게 구현되었는지에 대한 정보가 활용되지 않음을 가정한 것이다. 즉 인터페이스를 통한 컴포넌트의 개조만으로 국한된다.

③ Composable : 개조된 컴포넌트 역시 다른 컴포넌트와 다시 합성될 수 있어야 하며 개조된 컴포넌트는 다시 개조될 수도 있어야 한다.

④ Homogeneous : 기존의 컴포넌트를 사용하는 코드는 여전히 개조된 컴포넌트를 사용할 수 있어야 한다.

⑤ Ignorant : 기존의 컴포넌트는 개조에 대한 아무런 사전 지식이 없어야 한다.

⑥ Identity : 기존의 컴포넌트는 개조와 관계없이 독립적으로 식별 가능해야 한다. 즉, 추후의 또 다른 개조에 사용 될 수 있어야 한다.

⑦ Architectural focus : 기존의 컴포넌트와 개조된 컴포넌트를 포함하여 개발할 어플리케이션의 아키텍처에 대한 명세를 제공할 수 있어야 한다. 이때 기존의 컴포넌트와 개조된 컴포넌트의 명세는 달라야 한다.

3. TA&RS Component 시스템

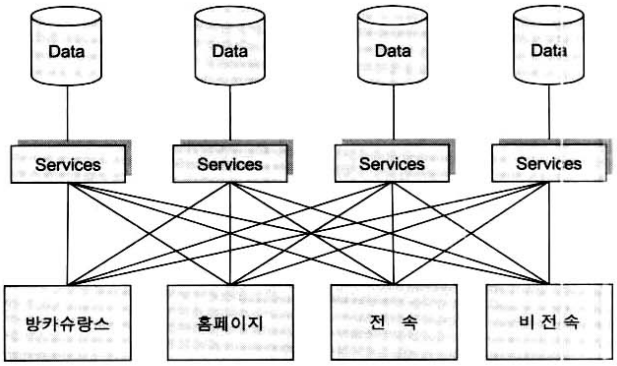
3.1 래거시 시스템의 래핑

최근에 J2EE, Dot Net기반의 컴포넌트 기반 소프트웨어 개발 기술이 발전하면서 많은 기업들이 새롭게 개발하는

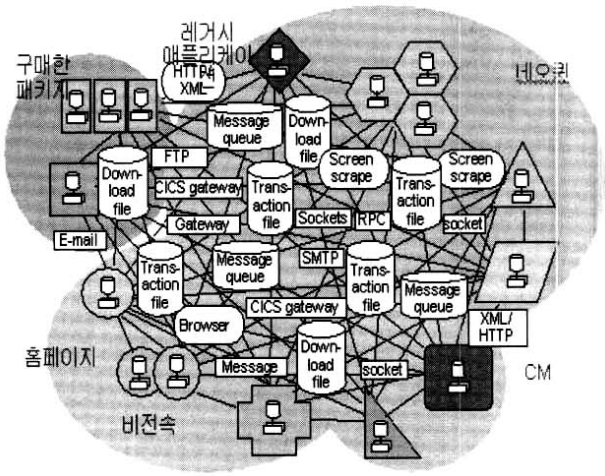
시스템에서 신기술을 적용하고 있다. 그러나 아직도 기업의 중요한 시스템들은 대부분 COBOL로 개발되어 메인 프레임 기반에서 개발되고 있다. 레거시 시스템은 수십 년간 운영되어 오면서 그 서비스의 신뢰성, 안정성, 사용성이라는 품질 요소를 만족한다. 반면 COBOL 언어가 갖고 있는 한계와 레거시 시스템 개발 당시의 개발 기법의 한계 때문에 모듈의 확장성(Extendibility), 이식성(portability), 상호운영성(interoperability), 재사용성의 측면에서는 품질이 떨어진다는 단점을 갖고 있다. 이러한 단점에도 불구하고 레거시 시스템은 조직의 매우 중요한 자산임에는 틀림없다. 그러므로 레거시 시스템의 안정적 서비스는 새로운 웹 또는 컴포넌트 기반의 아키텍처와 통합되어야 한다. 레거시 시스템은 다른 장비, 다른 외부 애플리케이션으로부터 서비스 요청을 처리할 수 있는 기반을 갖추고 있다. 다시 말하면 레거시 시스템을 리소스 레이어로 보면 새로운 애플리케이션 레이어로부터의 요청된 서비스를 제공할 수 있다는 것이다. 이로써 레거시 시스템이 제공하고 있는 이미 검증되고 안정화된 서비스는 새롭게 개발할 필요 없이 그 자체를 컴포넌트로 간주하는 것이 가능하다.

컴포넌트라고 하면 안정된 인터페이스를 제공하여 다른 요소들과 인터페이스 기반의 통합이 가능해야 한다. 현재 레거시 시스템은 인터페이스 기반의 컴포넌트가 아니므로, 레거시 시스템의 모듈을 컴포넌트의 구현으로 보고 우리는 정형화된 인터페이스만 정의하면 재사용 가능한 대상이 될 수 있다. 기존에도 레거시 시스템을 하나의 리소스로 보고 다양한 클라이언트 애플리케이션에서 레거시 시스템을 재사용해 왔다. 이 과정에서의 문제점은 각 시스템마다 레거시 시스템을 접근하여 서비스를 얻어갔기 때문에, 클라이언트 애플리케이션과 레거시 시스템간의 강한 종속 관계가 생기게 되었고 한 클라이언트 변경 요청에 의한 레거시의 변경으로도 다른 클라이언트 애플리케이션에 영향을 줄 수 있는 구조였다. 즉 (그림 2)와 같이 N개의 레거시 서비스별로 애플리케이션의 접근 방법이 달라지는 N×N의 복잡한 결합도를 갖고 있다. 레거시를 컴포넌트로 보고 인터페이스를 정의한 후 이 단일 인터페이스로의 접근으로 일원화하는 래퍼(Wrapper)를 통해 진정한 의미의 컴포넌트화가 가능하다.

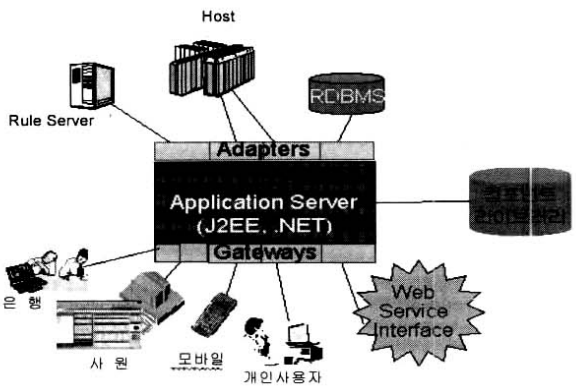
(그림 3)과 같이 전체 시스템의 아키텍처 및 구성도의 부재로 인하여 상위 수준의 아키텍처 및 시스템 구성도가 없기 때문에 시스템을 통합적 관점에서 파악할 수 없는 단점이 있다. 이러한 문제점으로 인하여 아키텍처상의 구성 요소로서 소프트웨어 모듈할당이 이루어지지 않고 비즈니스 모델, 화면, 제어의 독립적 모듈화가 이루어지지 않으며 다양한 요소들의 통합 전략 부재로 프로그램간의 종속성이 강하게 이루어짐으로서 체계적인 소프트웨어 설계가 이루어지지 않는다. 그리고 정형화된 유지보수 프로세스가 없고 다양한 업체별 개발 방법이 존재하기는 하나 이를 통합적으로 관리할 전략이 없는 관계로 체계적인 시스템을 관리할 방법이 없다.



(그림 2) N 서비스 × N 데이터



(그림 3) 아키텍처의 부재로 인한 복잡한 시스템



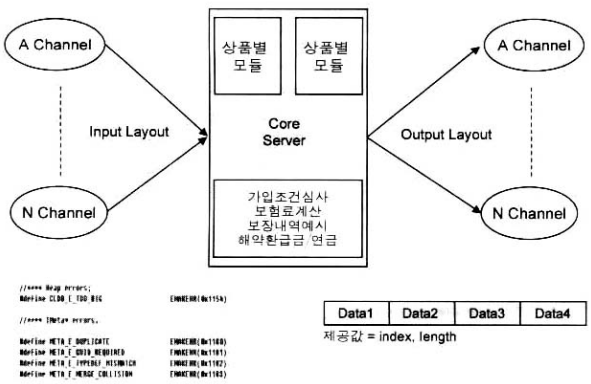
(그림 4) CBD 아키텍처를 적용한 시스템

이렇게 복잡한 시스템을 컴포넌트 아키텍처 정의 및 명세(Component Architecture Definition & Specification

Definitione)를 통하여 컴포넌트 생성, 획득, 조립을 위한 표준 참조 모델을 정의하고 이에 따른 메타 데이터와 컴포넌트 명세 방법을 결정함으로써 컴포넌트의 수요, 공급, 활용의 핵심 지침을 마련한다. 이렇게 컴포넌트 아키텍처를 적용할 경우 ① 기존의 자산에 대한 재사용 문제 ② 다양한 CBD 아키텍처로 인한 기존의 클라이언트 재사용성에 대한 문제가 발생할 수 있는데 이것은 (그림 4)와 같이 Adaptor와 Gateway와 같은 개조기 기술을 통해 해결할 수 있다.

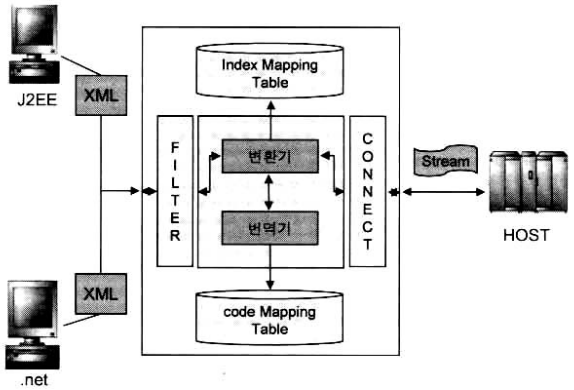
3.2 TA(Transaction Adaptor) Component

본 연구에서는 레거시 시스템에 접근하기 위한 방법으로 레거시 인터페이스를 XML로 래핑하는 방법을 택하였다. 이를 위해서는 레거시 시스템 컴포넌트를 XML로 래핑하는 래퍼(Wrapper)가 필요하다. 클라이언트는 XML 인터페이스를 통해서 레거시 시스템 컴포넌트를 접근하지만, 결국 레거시 시스템의 인터페이스는 (그림 5)와 같이 스트림(stream) 방식일 수 밖에 없다. 클라이언트와 리소스간의 인터페이스 불일치를 해결하기 위한 중간 래퍼가 필요한데, 본 연구에서는 이를 트랜잭션 어댑터(Transaction Adaptor)로 구현하였다. TA의 가장 큰 역할은 컴포넌트 클라이언트 인터페이스인 XML과 리소스 인터페이스인 스트림간의 구조와 의미를 대응시키고 변환하는 매핑의 역할이다. 또한 XML에 정의된 실제 리소스 컴포넌트를 호출하고 그 결과를 클라이언트에게 반환하는 역할도 수행해야 한다. TA자체도 컴포넌트로 구현하였으며, 구조와 의미를 매핑하기 위한 매핑 정보는 컴포넌트 속성(property)으로 분리하여 다양한 종류의 서비스에 대응할 수 있도록 하였다. 물론 인터페이스 래핑의 기준은 컴포넌트의 인터페이스이다. 즉 레거시 시스템이 정의한 인터페이스 방식을 클라이언트가 활용할 수 있는 형태의 XML 스키마로 변환해야 한다.

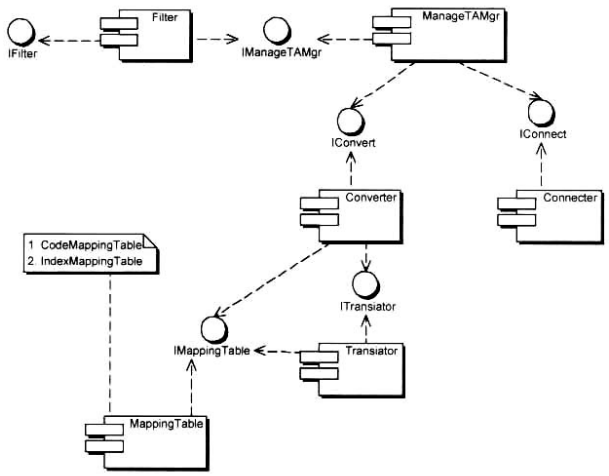


(그림 5) 레거시 시스템의 stream방식 인터페이스

TA서버는 (그림 6)과 (그림 7) 같이 클라이언트 애플리케이션과 데이터를 송수신하는 통신부, 트랜잭션 어댑터 및 제어부로 구성하였다. 제어부는 통신부 및 트랜잭션 어댑터를 통합 관리하는 영역을 의미한다. 트랜잭션 어댑터는 수신된 XML데이터를 필터부(Filter), 번역기(Translator), 변환기(Converter) 및 커넥터(Connector)를 거쳐 레이아웃 데이터로 변환하여 호스트 서버인 레거시 시스템으로 전송한다.



(그림 6) TA 시스템 구조도



(그림 7) TA 아키텍처 다이어그램

트렌젝션 어댑터에서는 클라이언트로부터 전송받은 XML 데이터를 업무의 성격 또는 레거시 컴포넌트의 구현 방식에 독립적으로 레이아웃 데이터로 변환한다. 그러므로 예전 방식의 직접 인터페이스 방식에 비해서는 레거시 시스템과의 인터페이스 불일치의 문제를 TA가 통합 처리함으로서 TA 프로그램 코드 자체의 재사용 및 운영 관리 측면에서 매우 효율적이다.

클라이언트 애플리케이션에서는 입력받은 데이터를 XML 데이터 형식으로 변환하여 TA 서버로 전송한다. 통신부를 통해 전송받은 XML데이터는 필터부에서 태그필터를 삭제하고 순수한 입력 데이터를 걸러낸다. 번역기에 의해 코드 맵핑 테이블을 참조하며 코드화 작업을 수행한다. 코드화된 데이터는 변환기를 거치며 인덱스 매핑 테이블을 참조하여

레이아웃 데이터로 변환을 수행한다. 접속부는 레거시 시스템에 접속한 후 변환된 레이아웃 데이터를 전송한다. 레이아웃 데이터는 레거시 시스템 상에서 별도의 데이터 처리 작업 없이 업무에 대한 처리를 진행하여 그 결과를 TA로 전송한다.

인덱스 매핑 테이블은 TA 시스템의 변환기(Converter)가 사용하는 일종의 인덱스 사전(Index Dictionary)이다. 변환기는 인덱스 매핑 테이블을 이용하여 Layout과 XML 사이의 변환을 수행한다.

TA 컴포넌트가 제공하는 기능은 <표 1>과 같이 크게 4가지로 분류할 수 있다. 이러한 기능을 가진 TA 컴포넌트를 통해 기 개발되어진 레거시 시스템을 컴포넌트화하여 재사용할 수 있으며 여러 종류의 호스트에 대한 접속을 단일화하여 호스트의 부하를 낮추고, XML을 이용하여 향후 해당 도메인에서 재사용시 투명한 개발이 이루어지며 클라이언트의 플렛폼에 독립적인 환경을 제공할 수 있다.

〈표 1〉 TA 컴포넌트의 기능

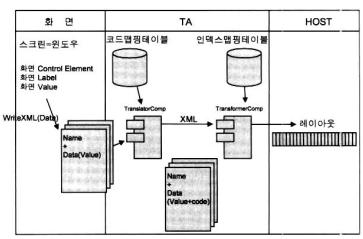
분 류	기 능
레거시 시스템의 컴포넌트화	- 호스트 비즈니스 서비스를 컴포넌트화 하고 안정된 인터페이스를 제공 - Wrapping 기법에 의한 레거시 시스템의 컴포넌트화 - TA 컴포넌트를 통한 호스트 비즈니스 서비스의 재사용 확대 - TA 컴포넌트를 통해 다양한 클라이언트의 호스트 접속 요구사항 처리
호스트 접속의 단일화	- TA 컴포넌트를 통해 다양한 클라이언트 호스트 접속 요구사항 처리 - TA 컴포넌트를 통한 호스트 비즈니스 서비스의 재사용 확대 - TA가 호스트로 보내지는 불필요한 요청을 줄일 수 있음
호스트 레이아웃과 코드에 따른 효율성 의미해석	- 레이아웃 구조에 따른 전송 XML생성 - 레이아웃 스트림에 따른 전송 XML 값 생성
클라이언트 플랫폼에 독립적	- 다양한 CBD 플랫폼을 제공 - 프로토콜별 Filter 제공(SOAP, HTTP, RMI)

TA를 이용하여 클라이언트쪽에서 호스트쪽으로 데이터를 처리하는 방법은 (그림 8)과 같이 각 채널별 클라이언트쪽에서 데이터를 받아서 코드맵핑 테이블을 참조하여 변환 컴포넌트가 XML로 변환하고 이것을 인덱스 맵핑 테이블을 참조하여 Stream 변환 컴포넌트가 해당 데이터에 맞는 형태의 입력 Stream으로 변환하여 호스트에 전달한다. 이와는 반대로 호스트쪽 데이터를 클라이언트쪽으로 전달받을 경우에는 (그림 9)와 같이 호스트쪽 Stream을 인덱스 맵핑 테이블을 참조하는 컴포넌트(TA)가 XML형태로 변환하고 이 데이터를 인덱스 맵핑 테이블을 사용하는 컴포넌트(TA)를 통해 해당 채널 클라이언트가 요구하는 형태의 데이터로 변환하여 전달한다. 이렇게 함으로서 기존에 개발되어진 클라이언트 프로그램이나 새로 개발되어질 클라이언트 프로그램에서 사용되어질 수 있고 기존에 중명되어진 레거시 시스템의 자원을 재사용할 수 있도록 할 수 있다.

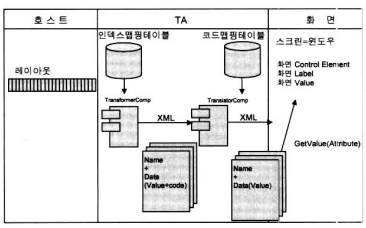
〈표 1〉 TA 컴포넌트의 기능

분 류	기 능
레거시 시스템의 컴포넌트화	- 호스트 비즈니스 서비스를 컴포넌트화 하고 안정된 인터페이스를 제공 - Wrapping 기법에 의한 레거시 시스템의 컴포넌트화 - TA 컴포넌트를 통한 호스트 비즈니스 서비스의 재사용 확대 - TA 컴포넌트를 통해 다양한 클라이언트의 호스트 접속 요구사항 처리
호스트 접속의 단일화	- TA 컴포넌트를 통해 다양한 클라이언트 호스트 접속 요구사항 처리 - TA 컴포넌트를 통한 호스트 비즈니스 서비스의 재사용 확대 - TA가 호스트로 보내지는 불필요한 요청을 줄일 수 있음
호스트 레이아웃과 코드에 따른 효율성 의미해석	- 레이아웃 구조에 따른 전송 XML생성 - 레이아웃 스트림에 따른 전송 XML 값 생성
클라이언트 플랫폼에 독립적	- 다양한 CBD 플랫폼을 제공 - 프로토콜별 Filter 제공(SOAP, HTTP, RMI)

이러한 TA Component를 이용하여 기존의 레거시 시스템의 재사용은 (그림 10)과 같이 처리되어 진다.



(그림 8) 클라이언트 XML데이터를 Stream으로 변환 고정

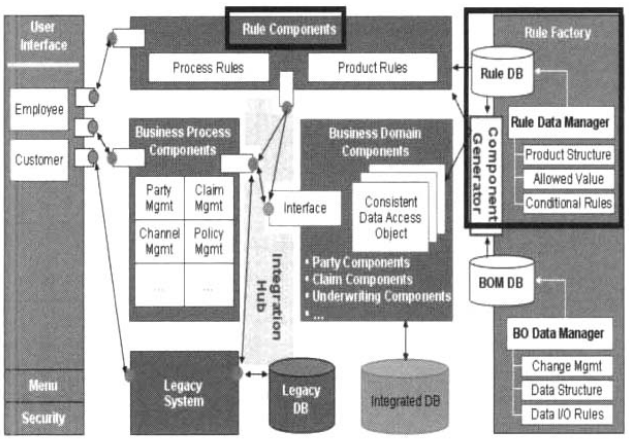


TA를 이용한 재사용 프로세스는 업무분석과 Application

설계, Application 조립, 인터페이스 확장의 4가지 분류를 가지고 있다. 레거시 시스템을 분석하여(business 분석) 필요로 하는 인터페이스를 획득하고 이를 맵핑할 수 있는 인터페이스를 정의한 후 이를 채널 응용 프로그램과 매핑하는 방법으로 재사용할 수 있다.

3.3 RS(Rule Server) Component

RS는 TA 컴포넌트를 통해 레거시 시스템의 변환 데이터나 CBD 기반으로 개발된 데이터의 검증 및 원하는 데이터의 추출을 목적으로 사용하는 컴포넌트로서 TA와 병렬적인 관계를 가진다. RS는 도메인 컴포넌트의 참조용으로 사용되어질 수도 있으며 비즈니스 프로세스의 처리에도 사용되어진다. 도메인 컴포넌트의 참조용으로 쓰일 경우에는 TA에 의해 전달되어지는 XML 데이터를 특정 기준에 의한 데이터의 추출이나 프로세스 수행에 필요한 기준정보를 추출하는데 쓰이며 비즈니스 프로세스에 쓰일 때는 프로세스 자체를 룰로 정의하여 사용한다. (그림 11)은 이러한 기능을 구현하는 룰 시스템 아키텍처를 표현한 것이다.



(그림 11) 룰(Rule) 시스템 아키텍처

룰 서버 컴포넌트를 이용하여 비즈니스를 처리하기 위해서는 룰 컴포넌트가 이해할 수 있는 특정한 언어가 필요로 한데 이러한 룰의 기본적인 구문은 <표 2>와 같이 정의하였다. 기본적인 룰의 구문 형태는 동일하나 적용하고자 하는 응용 도메인에 따라 사용자 함수는 수정, 추가, 삭제되어져야 한다.

일련의 룰들의 실행을 통해 하나의 비즈니스 처리가 완성 될 수 있을 때, 즉 여러 룰들이 모여서 하나의 순차적 워크플로워를 형성할 수 있을 때 룰 집합을 실행한다. Biz Component에서 룰의 호출단위로 Rule Execution Set을 사용하며, 동일한 비즈니스 처리가 적용되는 모든 대상은 동일한 개수와 동일한 이름의 Rule Execution Set을 가져야 한다. 그리고 Rule Execution Set도 하나의 룰로 간주하며 업무처리를 위해서 불러져야할 룰들의 목록을 가지고 있게 된다.

〈표 2〉 룰(Rule) 구문 형태 및 설명

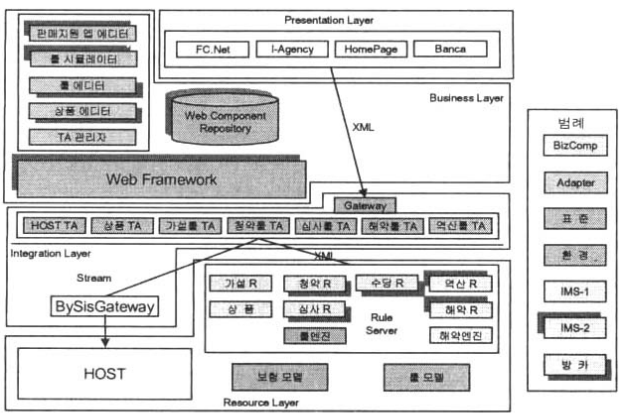
```
Return_type Rule_Header {
            IF (Condition)
            THEN
            ELSE
            ADDITIONALINFO}
```

- Return_type : 반환하는 데이터 타입
- Rule_Header : 룰이름으로서 의미가 구별되는 단어들은 "_"을 붙여 구분한다.
- Body : "{" "}"사이에 처리할 내용을 정의한다.
- IF(Condition) THEN Process1 ELSE Process2 : Condition은 논리연산자와 관계연산자를 사용할 수 있으며 해당 도전이 참일 경우 Process1을 처리하고 그렇지 않을 경우에는 Process2를 처리한다.
- ADDITIONALINFO
  - ADDITIONALINFO : 조건에 따른 처리와 상관없이 사용자에게 보여줄 수 있는 추가적인 정보를 정의하기 위해 사용.
  - ADDITIONALINFO THEN : THEN이 처리될 때 사용자에게 정보를 제공
  - ADDITIONALINFO ELSE : ELSE가 처리될 때 사용자에게 정보를 제공
- 기타 함수 : 해당 도메인별로 필요로 하는 처리함수 정의
  예) 보험도메인 : getAGE, getAccountCount, calculatePremiumFee.
- 기타 함수 처리
  - CALL(함수이름) : 내장함수 및 사용자 함수 호출
  - FIRE(룰이름) : 다른 룰 호출

4. 사례 연구(보험도메인)

4.1 TA를 이용한 보험도메인의 정의

TA를 이용한 보험도메인의 CBD 아키텍처는 리소스 레이어, 인터그레이션 레이어, 프레젠테이션 레이어, 비즈니스 컴포넌트 레이어의 총 4개의 레이어로 구분하여 정의한다. 총괄적인 아키텍처는 (그림 12)와 같으며 각 항목별 설명은 <표 3>과 같다.



(그림 12) 보험 도메인을 위한 CBD아키텍처

① 리소스 레이어

호스트 : 기존 보험 처리계로 레거시 컴포넌트, 메인프레임과 같은 시스템으로 구성되어 있으며 레거시시스템에 의해 안정성을 확보 받은 데이터의 처리가 가능하다.

룰서버 : 가설과 청약 업무를 룰 기반으로 처리하는 컴포넌트로서 TA로부터 변환되어온 데이터를 검증 및 필요로 하는 데이터의 추출에 사용되어 진다.

② 인터그레이션 레이어

Gateway, E-Gate : 호스트로의 연동을 제공하는 외부 Package로서 stream 방식과 EAI 방식의 호스트 접속 프로그램으로서 3rd party 응용 프로그램이다. 이를 통하여 다양한 시스템들과의 유연한 통신을 제공한다.

〈표 3〉 아키텍처 구성 요소별 설명

구 분	이 름	내 용	결 과 물
AL	CM	룰기반 비즈니스 컴포넌트와 호스트TA를 이용한 보험사의 CMR을 위한 보험판매지원 애플리케이션	CM 애플리케이션, 산출물
AL	방카	룰기반 비즈니스 컴포넌트와 호스트TA를 이용한 첫 번째 애플리케이션으로 삼성생명의 방카슈랑스 비즈니스를 지원, 보험사의 보험판매의 최전방에 위치한 애플리케이션으로 핵심적 역할을 수행	
IL	TA Connector -Java	자산이 되는 컴포넌트(비즈니스컴포넌트, 기반컴포넌트)로의 인터페이스를 캡슐화하는 컴포넌트로 컴포넌트의 위치 투명성, 프로토콜 투명성, 데이터 인터페이스 투명성을 보장	클라이언트배포용클래스
IL	TA Locator	클라이언트의 요청에 따라 자산 컴포넌트 중 어떤 컴포넌트에게 요청을 전달해야 하는가를 판단하는 클래스로 컴포넌트의 위치 투명성을 보장	클래스
IL	호스트 TA	레거시 컴포넌트로의 인터페이스를 단일화하는 컴포넌트로 스트림 기반의 인터페이스를 XML기반으로 변경하므로써 비즈니스 가독성을 증대	TA 컴포넌트
BC	가설	판매프로세스의 초기활동인 가입설계에서 이루어지는 채널별 공통의 프로세스를 담당한다. 주로 가입조건심사, 보험료 구하기, 보장내역예시, 연금 및 해약 환급금 구하기, 기타 부가정보 구하기등의 워크플로우를 처리한다.	가입설계컴포넌트
BC	청약	보험에 대한 가입의사를 구체적으로 밝히는 청약프로세스를 담당한다. 가입조건심사, 보험료 구하기, 개인에 대한 인심사, 단일계약에 대한 위험사정심사의 워크플로우를 처리	청약컴포넌트
BC	상품	판매의 단위가 되는 상품별 기본정보를 관리하는 컴포넌트로 판매를 위해 필요한 프로세스의 각 단계별 처리에 필요한 정보와 해석방법을 제공	상품정보조회
BC	가설룰	상품이 판매되기 위해 필요한 처리의 규칙과 판매 개념으로 상품을 해석하는 방법을 정의(가설시)	
BC	청약룰	상품이 판매되기 위해 필요한 처리의 규칙과 판매 개념으로 상품을 해석하는 방법을 정의(청약시)	
BC	보험처리 클래스	각 상품별 설계개념을 바탕으로 보험료, 적립금 연금, 각종 가산금등을 구하는 보험도메인의 핵심	
BC	웹스타일러	가입설계 컴포넌트의 처리결과를 웹스타일러 도구에 의해 작성된 가입설계서로 매핑서비스	매핑처리컴포넌트
BD	보험모델	보험사에서 판매하는 보험에 대한 판매모델을 정의하여 판매에 필요한 정보의 종류와 이를 해석하는데 필요한 구조를 정의한 비즈니스 모델	
TR	룰엔진	정의한 if-then-else의 기본구문과 XML기반의 모델로부터 정의한 표현식을 해석하는 룰 해석기	
	룰모델	비즈니스룰을 정의하는 구문으로 JSR을 준수하고자 노력하였으며, 어떤 모델이든 XML로 정의한 모델을 해석할 수 있도록 정의	
DS	웹인터페이스 생성기	상품에 정의된 정보를 해석하여 각 상품별로 화면구성 요소와 각 요소가 처리해야 하는 기본조건 심사코드를 생성, 이를 위해 가설 웹인터페이스 생성에 필요한 기본 컴포넌트를 저장소에 구축	웹인터페이스생성기, 가설웹컴포넌트
DU	상품룰에디터	상품모델 룰모델에 따라서 직접 상품과 룰을 등록하고 수정하는 환경	상품/룰에디터
DU	상품조회 시스템	서버마다 등록된 상품목록을 조회하고 상품의 판매방법을 조회, 판매지원시스템을 서비스하는 유지보수인력이 상품에 대한 상담을 진행할 때 참조용이면서 현업에서 정확한 값이 저장되었는지를 확인하는 용으로 사용	상품조회시스템
DU	TA관리자	호스트 레이아웃을 통해 클라이언트 프로그래머가 사용한 XML을 생성한다.	TA관리자

주) **AL** : 애플리케이션레이어, **IL** : 인터그레이션레이어, **BC** : 비즈니스컴포넌트, **BD** : 비즈니스도메인모델, **TR** : 테크니컬참조모델, **DS** : 개발자지원환경, **US** : 사용자지원환경

TA 컴포넌트 : 리소스 레이어의 컴포넌트와의 인터페이스 불일치를 처리하는 레이어로서 비즈니스 트랜잭션 단위로 인터페이스의 불일치를 처리하는 컴포넌트이며 이를 통해 기존의 레거시 시스템에서 사용되어지는 데이터나 비즈니스 프로세스를 처리할 수 있다.

③ 비즈니스 레이어

각 채널로 별도의 비즈니스 처리를 담당하는 컴포넌트이다. 이는 각각의 채널별로 접근하는 데이터나 처리하는 내용의 차이를 처리하는 레이어다. 비즈니스 레이어는 실제 현업에 있는 설계사들이 접근하는 데이터를 채널별로 관리하며 처리하는 곳이다.

④ 프레젠테이션 레이어

고객과 인터페이스가 이루어지는 접점으로서 여러개의 채널로 구성되어 있다. 각각의 채널은 응용 용도에 따라 WWW, EJB, .net등의 프레임워크를 응용하여 여러 가지 환경에서 운용되어진다.

4.2 TA를 통한 레거시 시스템의 재사용

본 연구에서 개발한 TA 서버를 보험 기간계 시스템의 재사용 환경에 적용해 보았다. 보험 업무관리는 보험의 세부 계약 내용을 청약하는 가입설계과정, 가입자의 청약에 대하여 보험사의 처리를 진행하는 업무처리과정 및 보험사가 보험청약 승인 후 보험 관리 업무를 처리하게 되는 보험 업무 관리과정으로 나뉘어진다. 또한 이후 지급, 보전, 대출 등의 다양한 업무 수행이 이루어진다. 이러한 레거시 시스템을 한번에 개방형 환경으로 전환한다는 것은 매우 어렵고 위험한 일이다. 점진적 개방형으로의 전환을 목적으로 새로운 채널에 대한 개발의 생산성 향상을 위해 기존 레거시 시스템의 컴포넌트화 및 컴포넌트 기반 소프트웨어 개발이 필수적이다.

트랜잭션 어댑터는 보험 처리 업무의 형식과 클라이언트 플랫폼에 관계없이 통합된 절차로서 레거시 시스템의 인터페이스를 XML화한 후 이를 레이아웃 데이터로 변환하여 레거시 시스템으로 전송함으로서 TA 서버를 이용한 레거시 시스템의 보험업무관리 시스템의 재사용을 가능하게 하였다. 레거시 시스템을 재사용 가능한 컴포넌트화 함에 있어 표준 인터페이스 기능을 담당하는 수단으로서 TA 서버는 레거시 컴포넌트의 표준 인터페이스로 업무별로 유연하게 추가할 수 있도록 XML을 인터페이스 수단으로 사용하였다.

① 새로운 애플리케이션의 필요한 서비스를 분석한다. 이 과정에서 유즈케이스 모델링, 비즈니스 객체 모델링 또는 컴포넌트 명세화 작업을 수행한다.

② 필요한 컴포넌트 구현을 기존의 레거시 시스템에서 제공하는지에 대한 컴포넌트 및 인터페이스의 식별을 수행한다.

③ 레거시 시스템에 대한 인터페이스를 정의한다. 이 과정에서 필요하다면 레거시 시스템의 재공학을 통한 모듈화를 수행한다. 프레젠테이션 로직과 비즈니스 로직의 분리 및 인터페이스의 안정화를 위한 모듈의 통합 등을 수행 할 수 있다.

④ 레거시 시스템에 대한 인터페이스를 XML로 정의한다.

⑤ XML과 레거시 컴포넌트의 인터페이스간의 매핑 정보를 등록한다.

⑥ 클라이언트 애플리케이션은 XML 인터페이스 기반으로 레거시 컴포넌트를 이용하는 프로그램을 개발한다.

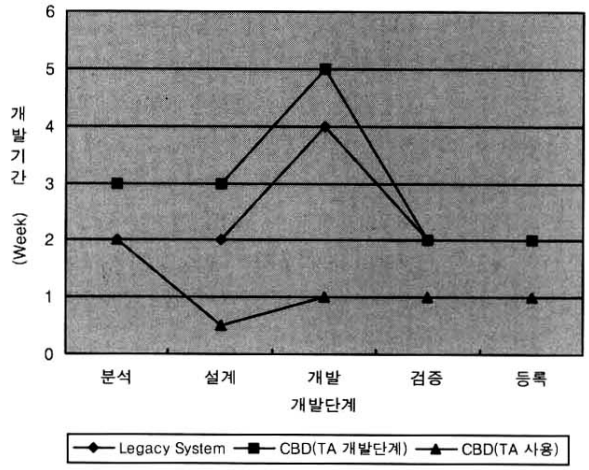
⑦ 클라이언트 애플리케이션과 레거시 시스템을 XML 기반으로 통합하여 서비스 한다.

5. TA&RS를 이용한 시스템의 평가

TA와 RS를 사용하여 개발할 때 표면적으로는 기존의 개발 시스템에 적응되어 있는 개발자들에게 많은 부담을 준다. 그러나 조직의 프로세스에 맞도록 완성되어진 TA와 RS가 완성되어지면 상당한 개발 기간 단축의 효과를 볼 수 있다.

(그림 13)은 TA와 RS에 대하여 보험 도메인에서 상품 등록시스템의 개발에 대한 개발기간을 비교한 그래프이다. 그림에서와 같이 하나의 보험공시가 나왔을때 이를 분석하고 이를 설계할 수 있는 시스템을 개발하기까지는 평균 3달의 소요 시간이 들었다. 이러한 이유는 보험이라는 도메인의 특성상 상품의 등록에 쓰이는 데이터의 오류는 보험 조직에서 치명적으로 적용되기 때문에 시스템의 개발에도 많은 시간이 소요되지만 이를 현업에서 실데이타로 검증하는데에도 많은 시간이 소요된다. 그렇기 때문에 보험 상품이 개발되어도 늦은 전산시스템의 적용으로 인해 적시에 가입자를 받아들이지 못하는 단점이 있었다. 그리고 TA와 RS를 개발하는 단계에서는 오히려 기존 레거시 시스템의 개발과 CBD에 대한 개발로 인력과 시간이 더 많은 시간이 소요되었다. 이 기간은 조직의 규모나 적용대상에 따라 다르겠지만 본 데이터의 대상은 생명보험의 방카슈랑스의 상품에 대한 데이터로서 약 1년 가까운 시간동안 TA와 RS가 개발되는 동안 오히려 더욱 많은 오류와 시간을 소비하게 되었다. 그러한 이유는 기존 레거시 시스템의 분석과 향후 확장될 시스템에 대한 대비로 인한 도메인 분석시간(이는 단순한 방카슈랑스 뿐만 아니라 다른 채널에도 적용을 시키기 위해 조직 전체의 분석이 필요함)이 소요되었다. 그

러나 기반 컴포넌트인 TA가 완성되고 상품등록에 대한 룰 컴포넌트인 RS가 개발이 완료된 시점에서는 오히려 기존 레거시 시스템의 개발 기간보다 약 1/3가량의 시간적 효과를 얻었으며 RS에 의한 자동화된 데이터 오류 검출의 효과를 얻을 수 있다.



(그림 13) 보험상품 등록시스템의 개발단계별 평균 소요시간

〈표 4〉 레거시 시스템 사용과 TA&RS 사용의 장단점 비교

개발환경구분	장 점	단 점
레거시 시스템의 사용한 개발	-기존 데이터와 프로그램의 재사용 -안정성 있는 데이터 활용	-재사용의 한계(모듈/Function 개념의 한계) -다양한 환경에서의 서비스 불가능(WWW, EJB, .net기반) -전산 시스템의 유지보수 비용의 기하급수적 증가 -장기적으로 호스트의 과부하로 인한 많은 호스트 업그레이드 비용 예상
CBD 기반으로 재개발	-기존 데이터와 프로그램의 재사용 -안정성 있는 데이터 활용 -유지보수 비용의 절감 -안정된 서비스 확보	-기존의 안정성 있는 데이터의 확인 불가능(재 작성 또는 컨버전 작업 필요로 많은 비용 필요) -조직에서 기존에 사용하는 전산 시스템의 재활용 불가능
TA&RS 컴포넌트를 이용한 개발	-기존 데이터와 프로그램의 재사용 -안정성 있는 데이터 활용 -기존 데이터와 프로그램의 재사용 -안정성 있는 데이터 활용 -장기적으로 유지보수 비용 절감 -안정된 서비스 확보	-기존의 시스템을 분석하고 TA와 RS를 개발하는 시간과 인력 그리고 추가 비용이 소요 -단기적으로 TA와 RS의 운영을 위한 추가 비용 소요

본 논문에서는 조직에서 기존에 안정화되어 있는 레거시 시스템을 CBD로 전환하려 할 때 기존 데이터나 기존 비즈니스 프로세스를 어떻게 전환하는가에 중점을 두어 TA 컴포넌트와 RS 컴포넌트를 설계하였다. TA와 RS 컴포넌트를 사용함으로서 기존에 안정화되어 있는 데이터나 개발 프로세스를 변화없이 래퍼 컴포넌트를 사용하여 재사용함으로서 CBD 기반의 시스템에 접합하는 방법을 시도하였다. 이렇게 함으로서 조직의 핵심자산 중 일부를 변환 없이 그대로 재사용할 수 있는 방법을 제공하며, 새로운 개발에는 CBD를 적용할 수 있는 방법을 제공하도록 한다. <표 4>에서는 레거시 시스템과 TA&RS를 이용한 CBD 개발의 장단점을 비교하였다. TA&RS를 이용한 시스템은 단기적으로는 조직에게 추가비용을 부담시키지만 장기적으로 운용시 기존의 레거시 시스템의 장점과 CBD의 장점을 모두 이용할 수 있는 시스템이다.

6. 결 론

본 논문에서는 TA 서버와 RS 서버를 이용한 레거시 시스템의 재사용 및 통합 방법을 제안하고 이를 실제 보험처리계 업무의 재사용에 활용해 보았다. TA 서버를 이용한 레거시 시스템의 재사용 및 통합 방법은 업무처리를 하는 사용자의 클라이언트 PC와 업무처리를 수행하는 레거시 시스템 사이에 TA 서버를 구축하고 TA 서버는 클라이언트 애플리케이션으로부터 XML 데이터 형식의 업무 데이터를 수신받아 TA 서버의 트랜잭션 어댑터를 통해 레이아웃 데이터로 변환하여 레거시 시스템으로 전달하기 때문에 업무의 종류와 클라이언트의 플랫폼에 관계없이 트랜잭션 어댑터가 데이터 변환처리를 통일적으로 수행하여 시스템 운영시 중복된 모듈의 재사용 및 업무처리를 효율적으로 수행하는 효과를 제공한다. 이는 XML을 이용하여 RS 서버를 통하여 원하는 프로세스를 정의하여 처리하고 레거시 컴포넌트의 인터페이스를 제공함으로서 안정적이며 확장 가능한 인터페이스 래핑에 의한 레거시 컴포넌트의 재사용을 가능하게 하는 방법이다. 이런 XML 래핑에 의한 레거시 시스템의 재사용은 실질적 개발의 생산성 및 기능의 안정적 서비스에 기여한다.

그러나 본 연구는 보험 도메인을 대상으로 레거시 시스템을 랩핑하여 재사용을 한 연구이기 때문에 다른 도메인에 적용할 경우 결과의 정확성을 예측할 수 없다. 이러한 문제는 향후 RS의 룰 규칙을 수정 보완하는 연구를 계속 진행할 예정이다.

참 고 문 헌

[1] Bradford Kain J. "Component : The Basics : Enabling an Application or System to be the Sum of its parts," Object Magazine, Vol.6, No.2, pp.64-69, April, 1996.

[2] Nierstrasz Oscar, Meijler Theo Dirk, 'Component-Oriented Software Technology,' Object-Oriented Software Composition, Prentice-Hall International, pp.3-28, Dec., 1996.

[3] jim Q. Ning, "Component-Based Software Engineering," IEEE Software, 1997.

[4] jim Q. Ning, "A Component-Based Software Development Model," in Proceedings of 21th Annual International Computer Software and Application Conference, 1996.

[5] Jan Bosch. "Superimposition : A Component Adaptation Technique. Information and Software Technology," Vol.41, No.5, pp.257-273, March, 1999.

[6] Ralph Keller, Urs Holzle, "Implementing Binary Component Adaptation for Java," www.cs.ucsb.edu/oocsb.

[7] Urs Holzle. "Integrating Independently-Developed Components in Object-Oriented Languages," Proceddings of ECOOP'93, Springer Verlag LNCS 512, 1993.

[8] George T. Heineman, "AnEvaluation of Component Adaptation Techniques," Computer Science Department, Worcester PolytechnicInstitute, WPI-CS-TR-99-04.

[9] Johannes Sametinger, "Classification of Composition and Interoperation," OOPSLA'96 Poster Presentation.

[10] Nierstrasz Oscar, Meijler Theo Dirk,"Research Directions in Software Composition," ACM Computing Surveys, Vol. 27, No.2, pp.262-264, June, 1995.

[11] Harry M.Sneed, "Using XML to Integrate existing Software System into the Web," Proceeding of the 26th COMPSAC'02.

[12] MTW Corp, "Legacy Wrapping in a Component Architecture," Technical report'02.

[13] Warren Jan, "The Renaissance of Legacy System," Springer Verlag, 1999.

[14] Paul Asman, "Legacy Wrapping," Proceedings of Plop 2000.

[15] Shengru Tu, et al, "Strategies for Integration of a Non-OOEIS and the J2EE Framework," Proceeding of the 26th COMPSAC'02.

[16] Rahul Sharma, et al., "J2EE Connector Architecture and Enterprise Application," Wesley, Reading, MA, 2001.

[17] Ganti, N., Brayman, W., "The Transition of legacy Systems to a Distributed Architecture," John Wiley&Sons, 1995.

[18] Sneed, H. and E. Nyary, "Downsizing Large Application Programs," Software Maintenance : Rearch ans Practice, Vol.6, No.6, pp.235-247, 1994.

[19] Kim Jung-Ah, "Component Adaptation through Adaptation Component," SERP'03, 2003.

[20] 김정아, "프레임워크 재사용을 위한 컴포넌트 재정의 도구", 한국정보처리학회 소프트웨어공학연구회지, 제2권 제2호, pp. 26-36, 1999.

[21] 차정은, 김철홍, 양영종, 박창순, "CBD환경으로의 레가시 시스템의 재공학 방법론", 한국정보처리학회 소프트웨어공학연구회지, 제5권 제2호, pp.17-26, 2002.

[22] 김정아, "컴포넌트 개조에 의한 재사용 기법", 한국정보과학회 소프트웨어공학회지, pp.19-25, 2002.

[23] 김정아, 김종윤, "TA를 통한 레거시 시스템 재사용", 한국정보처리학회 추계학술발표대회 논문집, 제10권 제2호, pp. 1681-1684, 2003.

김 상 영

e-mail : jayusop@zeus.dju.ac.kr

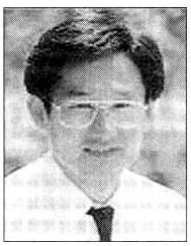
1999년 대전대학교 컴퓨터공학과(학사)

2001년 대전대학교 대학원 컴퓨터공학과 (공학석사)

2001년~현재 대전대학교 대학원 컴퓨터공학과 박사과정

관심분야 : 재사용, CBD, 품질 메트릭스, 테트팅 방법론

황 선 명



e-mail : sunhwang@dju.ac.kr

1982년 중앙대학교 전자계산학과(학사)

1984년 중앙대학교 대학원 전자계산학과 (이학석사)

1987년 중앙대학교 대학원 전자계산학과 (이학박사)

2000년~현재 한국 S/W프로세스 심사인협회(KASPA) 이사

2000년~현재 한국정보처리학회논문지 편집위원

1997년~현재 ISO/IEC JTC7/WG10 한국운영위원

1998년~현재 한국정보통신기술협회 TTA 특별위원

1989년~현재 대전대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 프로세스 모델, 품질 메트릭스, 소프트웨어공학 표준화, 컴포넌트 품질측정, 테스팅방법론 등

김 정 아

e-mail : clara@kwandong.ac.kr

1990년 중앙대학교 대학원 컴퓨터공학과 (공학석사)

1994년 중앙대학교 대학원 컴퓨터공학과 (공학박사)

1994년~1996년 중앙대학교 기술과학연구소 객원연구원

1996년~현재 관동대학교 컴퓨터교육과 부교수

관심분야 : 재사용, CBD, Product Line, MDA, 기술, 프로젝트 관리 및 프로세스 개선 등

진 영 택

e-mail : ytjin@hanbat.ac.kr

1981년 중앙대학교 전자계산학과(학사)

1983년 중앙대학교 대학원 전자계산학과 (이학석사)

1992년 중앙대학교 대학원 컴퓨터공학과 (공학박사)

1983년~1990년 한국 에너지기술연구소 연구원

1999년~2000년 호주 UTS대학교 컴퓨터공학부 교환 교수

1990년~현재 한밭대학교 정보통신·컴퓨터공학부 교수

관심분야 : 소프트웨어공학(객체지향 분석/설계, 설계패턴, 역공학, CBD, CBSE, Product Line)