트렌젝션 어뎁터 컴포넌트를 이용한 래핑에 관한 연구

김상영* 정지환* 김정아** 황선명*
대전대학교 컴퓨터공학과*
관동대학교 컴퓨터교육과**
e-mail:jayusop@zeus.dju.ac.kr

A Study wrapping using Transaction Adaptor component

Sang-Young Kim* Ji-Hwan Jung* Jung-Ah Kim** Sun-Myung Hwang*
Dept. of Computer Engineering, Daejeon University*
Dept. of Computer Education, Kwandong University**

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 어뎁터(Transaction Adaptor)라는 개조 컴포넌트를 이용하여 기존의 컴포넌트 또는 기존의 레거시 시스템을 재사용하는 것에 대한 연구이다. TA를 이용한 개조방법은 클라이언트와 호스트시스템 사이에 TA컴포넌트를 사용하여, XML데이터를 스트림 형태로 변환하여 전송함으로서 레거시 시스템을 재사용한다. 이러한 TA를 이용한 재사용 방법은 클라이언트 플랫폼이나 호스트의 종류에 관계없이 TA가 XML로 데이터 변환처리하여 레거시 시스템을 재사용할 수 있다.

1. 서 론

CBSD 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 통합 할 때 기존의 컴포넌트나 시스템을 변경하거나 구입한 컴포넌트를 수정하거나 기존의 시스템과 컴포넌트 사이에 개조기(Adaptor)를 두어 해결하는 것이 일반적이다.

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

본 연구는 한국과학재단 목적기초연구(R01-2001-000-00343 -0(2003))지원으로 수행되었음

2. 관련 연구

2.1 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.

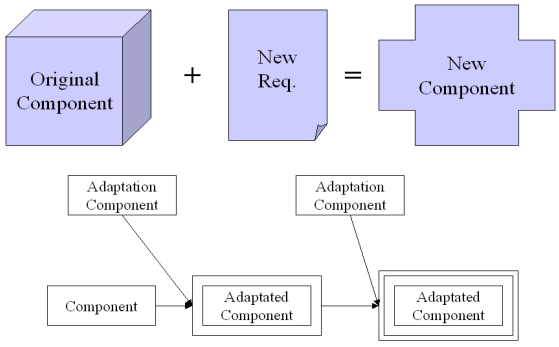
1) 개조기(Adaptor)기법 : 인터페이스의 차이를 보이는 두 컴포넌트 사이에 개조기 패턴을 활용하여 개조기 컴포넌트를 정의하여 해결한다. 이 개조기 컴포넌트가 차이나는 두 컴포넌트 사이의 인터페이스 불일치를 해결한다.

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 동작 시키도록 하려는 것이다[5]. 이는 개조의 대상이 되는 컴포넌트와 개조를 처리하는 컴포넌트 모두 독립적으로 재 사용하려는 목적을 갖는다. 이 방법은 컴포넌트의 개조 유형이 다양하지 않고 정해진 범위 내에서 이루어진다는 전재하에 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 reflextion 기법을 이용하여 컴포넌트 구현 내용을 역으로 추출 한 후 컴포넌트 자체를 수정하여 새로운 컴포넌트를 만들어 내는 방법이다.



(그림 1) Component Adaptation기본 개념

3. TA Component 시스템

3.1 TA(Transaction Adaptor) Component

본 연구에서는 레거시 시스템에 접근하기 위한 방법으로 레거시 인터페이스를 XML로 래핑 하는 방법을 택하였다. 이를 위해서는 레거시 시스템 컴포넌트를 XML로 래핑 하는 래퍼(Wrapper)가 필요하다. 클라이언트는 XML 인터페이스를 통해서 레거시 시스템 컴포넌트를 접근하지만, 결국 레거시 시스템의 인터페이스는 스트림(stream)방식일 수 밖에 없다. 클라이언트와 리소스 간의 인터페이스 불일치를 해결하기 위한 중간 래퍼가 필요한데, 본 연구에서는 이를 트렌잭션 어뎁터(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 매핑 하기 위한 매핑 정보는 컴포넌트 속성(property)로 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 XML스키마로 변환해야 한다.

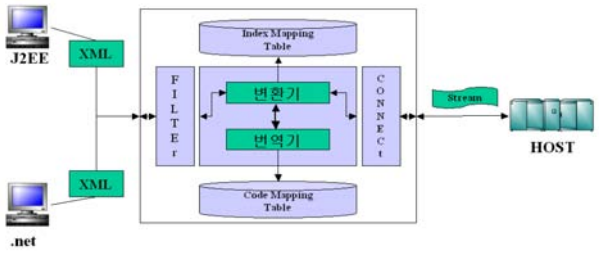
TA서버는 클라이언트 애플리케이션과 데이터를 송수신하는 통신부, 트랜젝션 어뎁터 및 제어부로 구성하였다. 제어부는 통신부 및 트랜젝션 어뎁터를 통합 관리하는 영역을 의미한다. 트랜젝션 어뎁터는 수신된 XML데이터를 필터부(Filter), 번역기(Translator), 변환기(Converter) 및 커넥터(Connector)를 거쳐 레이아웃 데이터로 변환하여 호스트 서버인 레거시 시스템으로 전송한다.

트렌젝션 어뎁터에서는 클라이언트로부터 전송받은 XML데이터를 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 TA프로그램 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 Layout 과 XML사이의 변환을 수행한다.

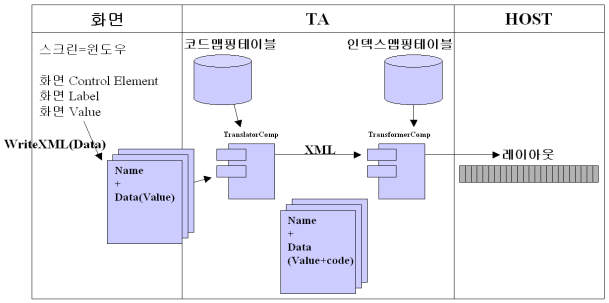
TA 시스템 구조도는 (그림 2)과 같다.



(그림 2) TA 시스템 구조도

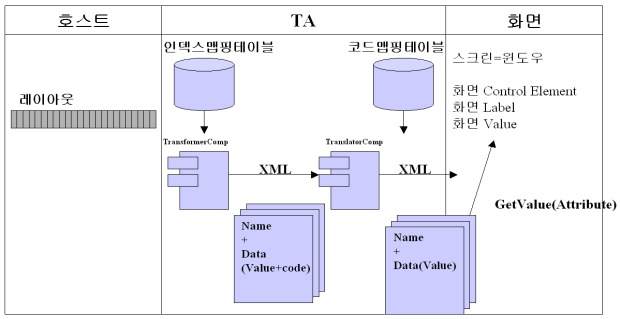
TA컴포넌트를 통해 기 개발되어진 레거시 시스템을 컴포넌트화하여 재사용할 수 있으며 여러 종류의 호스트에 대한 접속을 단일화하여 호스트의 부하를 낮추고, XML을 이용하여 향후 해당 도메인에서 재사용시 투명한 개발이 이루어지며 클라이언트의 플렛폼에 독립적인 환경을 제공할 수 있다.

TA를 이용하여 클라이언트쪽에서 호스트쪽으로 데이터를 처리하는 방법은 (그림 3)와



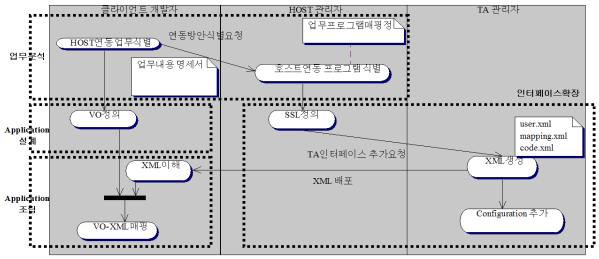
(그림 3) 클라이언트 데이터를 Stream으로 변환 과정

같이 각 채널별 클라이언트쪽에서 데이터를 받아서 코드맵핑 테이블을 참조하여 변환 컴포넌트가 XML로 변환하고 이것을 인덱스맵핑테이블을 참조하여 Stream변환 컴포넌트가 해당 데이터에 맞는 형태의 입력 Stream으로 변환하여 호스트에 전달한다. 이와는 반대로 호스트쪽 데이터를 클라이언트쪽으로 전달받을 경우에는 (그림 4)와 같이 호스트쪽 Stream을 인덱스맵핑테이블을 참조하는 컴포넌트(TA)가 XML형태로 변환하고 이 데이터를 인덱스맵핑테이블을 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 증명되어진 레거시 시스템의 자원을 재사용할 수 있도록 할 수 있다.



(그림 4) Stream을 클라이언트 데이터로 변환 과정

이러한 TA Component를 이용하여 기존의 레거시 시스템의 재사용은 (그림 5)와 같이 처리되어 진다.



(그림 5) TA기반 레거시 재사용 프로세스

TA를 이용한 재사용 프로세스는 업무분석과 Application 설계, Application조립, 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.

4. 결 론

본 논문에서는 TA Component를 이용한 레거시 시스템의 재사용 및 통합 방법을 제안하였다. TA를 이용한 레거시 시스템의 재사용 및 통합 방법은 업무처리를 하는 사용자의 클라이언트 PC와 업무처리를 수행하는 레거시 시스템 사이에 TA서버를 구축하고 TA서버는 클라이언트 애플리케이션으로부터 XML데이터 형식의 업무 데이터를 수신받아 TA서버의 트랜잭션 어댑터를 통해 레이아웃 데이터로 변환하여 레거시 시스템으로 전달하기 때문에 업무의 종류와 클라이언트의 플렛폼에 관계없이 트랜잭션 어댑터가 데이터 변환처리를 통일적으로 수행하여 시스템 운영시 중복된 모듈의 재사용 및 업무처리를 효율적으로 수행하는 효과를 제공한다. 이런 XML 래핑에 의한 레거시 시스템의 재사용은 실질적 개발의 생산성 및 기능의 안정적 서비스에 기여한다

참고문헌

[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, December, 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, 41(5):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.