제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

레거시 시스템을 포함한 자바 컴포넌트 설계 및 구현

백수진, 정화영, 송영재
경희대학교 컴퓨터공학과
e-mail : croso@cvs2.khu.ac.kr

Design and Implement the Java Component including the Legacy System

Su-Jin Baek, Hwa-Young Jeong, Young-Jae Song
Dept of Computer Engineering, Kyung-Hee University

요 약

컴포넌트 개발 방법론이 일반화되면서 기존 소프트웨어의 재사용과 유지 보수에 대한 필요성이 요구되었다. 그러나, 현재 대다수의 단위 컴포넌트 개발은 주로 각각의 산업 분야별로 컴포넌트 자체를 새롭게 개발하고 있는데 비해 기존의 레거시 시스템의 재사용은 많이 이루어지지 않고 있다. 레거시 시스템을 사용하는 기존 연구방법으로는 래핑 방법이나 변환 규칙을 적용하여 컴포넌트로 생성하거나 도입하려는 연구가 진행되고 있으나 이러한 기법은 전문가적인 부가 정보를 필요로 한다. 따라서 많은 부가정보나 수정 없이 쉽게 컴포넌트로 이용할 수 있는 기법이 요구된다.

본 논문에서는 프로그램의 재사용을 위해 이미 존재해 있는 레거시 시스템을 자바 기반의 어플리케이션과 JNI를 사용하여 연계하고, 범용적인 컴포넌트 모델인 자바빈즈를 채택하여 자바빈즈 컴포넌트로 변환하는 구조를 제안한다.

1. 서론

컴포넌트 개발 방법론이 소프트웨어의 대량 생산을 가능하도록 제공함으로써 기업에서는 성공적으로 프로젝트를 진행하기 위한 기존 소프트웨어의 재사용과 유지 보수의 필요성을 느끼게 되었다. 이를 위해서는 전반적인 소프트웨어의 설계나 원시 코드의 생성 등에서 유지 보수를 위한 소프트웨어의 이해는 물론, 컴포넌트로 되어 있지 않은 프로그램을 컴포넌트화 함으로써 재사용성을 높여주는 일 등이 필요하다. [1]

그러나 지금까지의 컴포넌트 개발은 컴포넌트 자체를 새롭게 개발하여 사용한데 반해 기존의 레거시 시스템의 자체의 재사용은 많이 개발되지 않고 있다. 기존 연구에는 레거시 시스템을 변환 규칙과 디자인패턴을 이용하여 새로운 컴포넌트로 생성하거나 래핑방법을 사용하여 컴포넌트화하는 방법이 연구되고 있다. 그러나 이러한 기법을 적용하기 위해서는 컴포넌트 변환이나 생성을 위한 많은 전문가적인 부가 정보를 필요로 한다. [2]

따라서, 본 논문에서는, 기존의 레거시 시스템을 많은 부가정보 없이 컴포넌트로 변환하는 기법을 제안한다. 즉, 프로그램의 재사용을 위해 이미 존재해 있는 레거시 시스템을 자바 기반의 기술인 JNI와 RMI를 통해 연계하여 자바빈즈로 재사용 가능한 컴포넌트로 변환하는 구조를 설계 및 구현한다.

2. 관련연구

본 장에서는 레거시 시스템과 레거시 시스템을 컴포넌트로 만들기 위해 사용되는 JNI, 자바 기반의 컴포넌트 모델인 JavaBeans에 대해 소개하고 Legacy system을 사용한 기존의 연구 방법에 대해 기술한다.

2.1 Legacy system

레거시 시스템은 조직의 데이터 처리를 위해 오랜 기간동안 개발되고 만들어진 코드와 데이터, 기술들을 말한다. 이것은 기본적인 프로세서, 중요한 데이터, 백본 네트워크, 관련기술, 개발에 사용된 과거의 언어들을 포함한다. 레거시 코드와 데이터는 조직의 중요한 자원이며, 새로운 시스템을 만들거나 통합시 반드시 고려되어야 한다. 그러나 레거시 시스템은 일체적이고, 수직적으로 통합된 어플리케이션들이며, 사용자 독자적인 방법을 적용한 패쇄된 시스템이다. 또 소프트웨어의 구조를 이해하기가 어려워서 개발자나 유지 보수 요원이 시스템 구조를 불완전하게 이해하게 되며, 재사용과 확장성의 기능이 부족하고 시스템 개발 및 배포시 많은 시간이 소모된다. 유지 보수 및 시스템 진화에 많은 비용이 드는 단점을 가지고 있다. 이러한 단점을 극복하기 위한 일반적인 방법으로는 어댑터와 래핑 기술 등이 있다. [3]

2.2 JNI

JNI(Java Native Interface)는 자바가상머신에서 실행되는 자바코드가 C, C++, 어셈블리 같은 다른 언어로 작성된 어플리케이션이나 라이브러리들과 동작하도록 한다. 다시말하면, JNI가 자바가상머신 내에 포함됨으로써 자바가상머신이 호스트 운영체제 상의 입출력, 그래픽스, 네트워킹 그리고 쓰레드와 같은 기능들을 작동하기 위한 로컬 시스템 호출을 수행할 수 있다.

JNI 프레임워크는 자바코드가 자바 오브젝트를 사용하는 것과 같은 방법으로 네이티브 메소드가 자바 오브젝트를 활용한다. 네이티브 메소드는 배열, 문자열을 포함하는 자바 오브젝트를 생성한 후 업무를 수행하기 위해 이런 오브젝트를 검사하고 사용할 수 있다. 또한 자바 어플리케이션 코드에 의해 생성된 오브젝트를 수정할 수 있다. 이런 수정된 오브젝트들은 자바 어플리케이션에서 유효하다. 그래서 네이티브 언어쪽과 어플리케이션의 자바쪽 모두 생성, 수정, 자바 오브젝트에 접근이 가능하고 양쪽간의 오브젝트들을 공유한다. 또한 네이티브 메소드들은 쉽게 자바 메소드를 호출할 수 있다.[4][5]

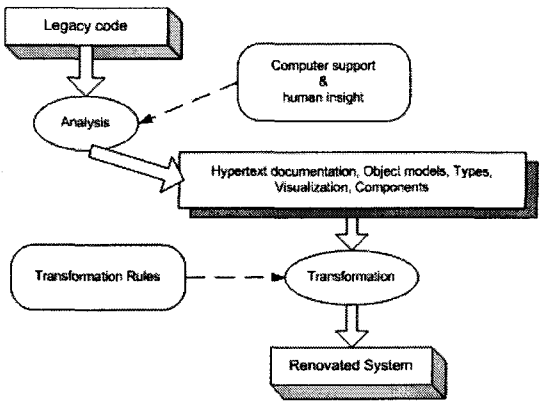
2.3 자바 기반의 컴포넌트 기술

컴포넌트 기반 개발은(Component Based Development) 기존의 소프트웨어를 통째로 개발하던 기존의 방식과 달리, 부품 역할을 하는 컴포넌트를 기능별로 개발하고 이중 필요한 것만을 선택하여 조립하는 소프트웨어 개발 방법이다.[6] 이 방법을 적용한 자바 기반 컴포넌트 기술에는 자바빈즈가 있다. 자바빈즈는 자바 프로그래밍 언어를 사용하여 개발된 이식성 좋은, 플랫폼 독립적인 소프트웨어 컴포넌트 모델이다. 자바빈즈의 목적은 어떤 플랫폼에서도 재사용할 수 있는 소프트웨어 컴포넌트를 만드는 것이다. 이는 개발자가 비주얼한 소프트웨어 컴포넌트인 자바빈을 자신의 애플리케이션에 맞게 적용하거나 수정하여 원하는 애플리케이션을 빨리 만들 수 있게 한다.

자바빈은 개발자 도구에서 비주얼하게 수행되어 질 수 있는 재사용 가능한 소프트웨어 컴포넌트이다.[7]

2.4 레거시 시스템을 이용한 기존 연구 기법

레거시 시스템을 이용한 기존 연구 방법인 Arie[2]의 변환과정은 다음(그림 1)과 같다.



< 그림 1 레거시 시스템의 분석과 변환 >

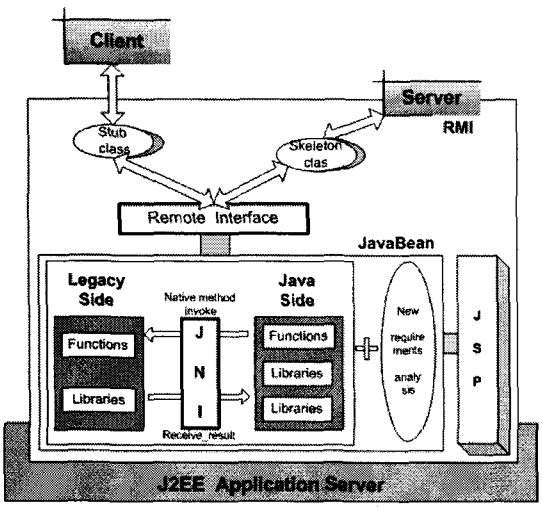
이러한 방법에서의 레거시 시스템은 Hyper 문서, 오브젝트 모델, 타입, 시각적 요소, 컴포넌트 추출 및 변환하기 위한 전문가적인 정보를 가짐으로써 새로운 시스템으로 재개발된다.

3. 레거시 시스템을 포함한 자바 컴포넌트 설계 및 구현

본 논문에서는 레거시 시스템을 자바 기반 컴포넌트인 자바빈즈에 포함시킴으로써 컴포넌트 기반 개발 기법에 적용될 수 있도록 하였다.

제안한 자바 컴포넌트 시스템은 (그림 2)과 같이 레거시 시스템을 Java 기반 시스템으로 변환하기 위

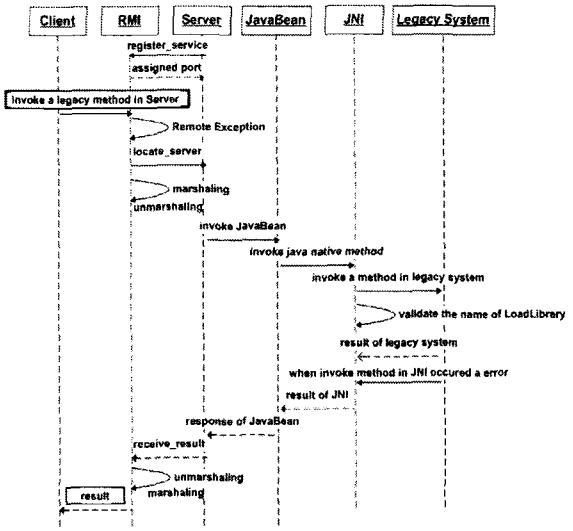
하여 JNI를 이용하였다. JNI를 통하여 변환되어진 레거시 시스템은 원격 메소드 호출 기법인 RMI를 통하여 접속되고 이를 Java 컴포넌트 기반인 JavaBean에 포함함으로써 컴포넌트화한다. 또한 레거시 정보 분석 외에 추가적인 정보를 포함시켜 개발자가 많은 수정을 거치지 않고 사용할 수 있도록 설계되었다. 그리고 JavaBean의 호출 결과는 View Logic을 담당하는 JSP에서 Browser를 통해 사용자에게 나타낸다.



< 그림 2 JNI를 이용한 자바빈즈 컴포넌트의 전체구조 >

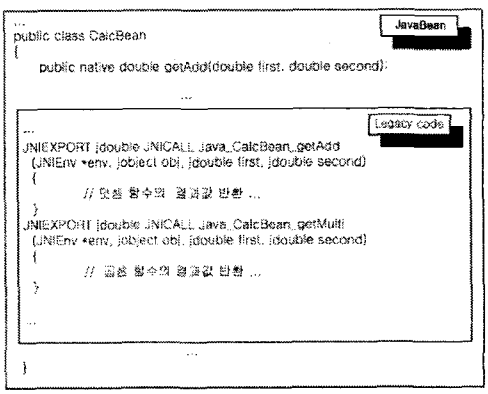
(그림 3)는 클라이언트가 서버에 있는 자바빈을 호출하는 시퀀스 다이어그램으로 자바빈에 포함된 레거시 시스템의 함수를 호출하여 원하는 값을 얻으며 클라이언트는 RMI를 이용하여 서버에 접근한다. 서버에서는 자바빈를 호출하게 된다. 자바로 작성된 프로그램은 자바 네이티브 인터페이스를 통해 레거시 시스템에 접근하여 함수를 호출하고, 결과값을 반환 받는다. 레거시 시스템에서 얻은 반환값은 다시 네이티브 인터페이스를 통해 자바빈에 반환된다. 서버에서는 자바빈즈에서 얻은 결과값을 클라이언트에게 되돌려주게 됨으로써 사용자는 자바빈즈 컴포넌트에서 원격의 레거시 시스템을 호출하여 그 결과를 얻을 수 있다.

본 논문에서 제안한 레거시 시스템을 자바 기반으로 컴포넌트화한 것을 보여주기 위하여 간단한 계산기 프로그램을 작성하였다. 대상 프로그램은 사용자의 입력을 받아 사칙연산을 수행하여 결과값을 보여



< 그림 3 레거시 시스템 호출 시퀀스 다이어그램 >

주는 프로그램이다. 윈도우즈 프랫폼에서 지원되는 C를 사용하여 개발되었다. C언어를 자바 기반의 컴포넌트로 변환하기 위해서는 우선 C언어로 작성된 코드와 자바언어로 작성된 코드가 연계될 수 있도록 자바에서 네이티브 메소드를 선언하였다. 자바 네이티브 메소드는 자바 오브젝트를 생성한 후 서비스를 수행하기 위해 오브젝트를 검사하고 이를 자바빈에서 사용할 수 있도록 제공된다. 수정된 오브젝트들은 자바 어플리케이션에서 유효하기 때문에 네이티브 언어쪽과 어플리케이션의 자바쪽 모두 생성, 수정, 자바 오브젝트에 접근이 가능하고 양쪽간의 오브젝트들을 공유한다. 아래 (그림 4)의 Legacy

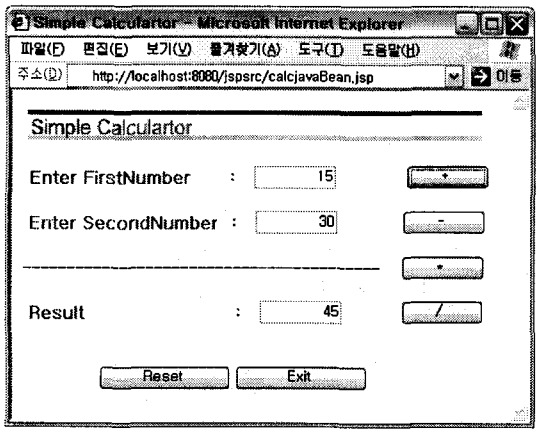


< 그림 4 자바 기반의 컴포넌트화하기 위한 레거시 시스템 코드 >

code 부분은 C로 작성된 레거시 코드를 자바와 연계될 수 있도록 코드 변형을 하였고, JavaBean 부분은 네이티브 메소드를 선언함을 보여주고 있다. 자바빈에서 C로 작성된 함수를 호출하기 위해서 아래와 같은 코드가 작성된다.

```
public double getBeanAdd(double first, double second)
{
    result = getAdd(first, second);  // c언어로 작성된 getAdd 함수 호출
    return result;
}
```

자바빈에서 선언된 함수에서 C로 작성된 getAdd 함수를 호출하여 결과값을 얻는다. 다음으로는 레거시 정보 분석 그밖에 환경 및 필요한 추가적인 정보를 포함하여 자바빈을 완성시킨다. (그림 5)는 자바 기반의 컴포넌트로 완성되었음을 보이기 위하여 JSP를 사용한 실행화면이다.



< 그림 5 레거시 시스템을 포함한 자바빈의 실행결과 >

위의 화면에서 원하는 수 두개를 입력하고 오른쪽에 나타나있는 +, -, *, / 중 하나의 버튼을 선택하여 누르면 아래 Result에 결과값이 출력된다. 여기서 계산하는 함수들(getAdd, getMulti, getSub, getDiv)은 레거시 시스템에 구현되어진 것이다. 이 함수들에 접근하기 위해서는 자바빈에서 자바 네이티브 인터페이스를 통해 불리워진다.

4. 결론 및 향후 연구과제

본 논문에서는 레거시 시스템을 JNI, RMI, JavaBeans를 사용하여 자바 기반의 컴포넌트를 만들 수 있는 구조를 설계 및 구현하였다. 이것은 기존 레거시 시스템을 개발자가 원하는 컴포넌트로 적용하기 위해서 많은 부가정보 없이 자바 기술만을 이용하여 컴포넌트로의 전환을 이룸으로써 컴포넌트의 장점인 재사용과 확장성의 기능을 가진다.

향후 연구과제로는 제안한 구조를 이용하여 컴포넌트 명세구조 및 아키텍처 기반의 컴포넌트 조립 합성이 이루어져야 한다.

참고문헌

[1] Jeffrey Voas, "Maintaining component-based systems," IEEE Software, July/August 1998.

[2] Arie van Deursen, Ben Elsinga, Paul Klint, Ron Tolido, "From Legacy to Component : Software Renovation in Three Steps", CAP Gemini, 2000.

[3] A. J. O'Callaghan, "MIGRATING LARGE-SCALE LEGACY SYSTEMS TO COMPONENT-BASED AND OBJECT TECHNOLOGY : The Evolution of a Pattern Language" , De Montfort University, Leicester, United Kingdom, CAIS, Vol 2, Article 3, July 1999

[4] Beth Stearns, Sun Microsystems Inc., "Java Native Interface", http://java.sun.com/docs/books/tutorial/native1.1

[5] Sun Microsystems Inc., "Java Native Interface Specification", May 16, 1997 http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html

[6] Alan W.Brown, Sterling Software, "From Compoent Infrastructure To Component-Based Development", ICSE Workshop, 1998

[7] Sun Microsystem Inc., "JavaBeans Tutorial", http://java.sun.com/products/javabeans/docs/javaBeansTutorial-Nov97