제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

기존 C++ 애플리케이션의 COM 컴포넌트로의 변환

정후식*, 최일우*, 류성열*
*숭실대학교 컴퓨터공학과
e-mail : dessert@selab.ssu.ac.kr

Transformation from Legacy C++ Application to COM Component

Hoo-Sik Jung*, Il-Woo Choi*, Sung-Yul Rhew*
*Department of Computing Graduate School, Soongsil University

요 약

소프트웨어는 컴포넌트 기반으로 개발되어 가고 있다. 컴포넌트 기반의 시스템은 소프트웨어의 복잡해지고 다양한 요구사항에 대해 부품을 조립하듯 빠르게 대응할 수 있다. 이미 기존에 구축되어 있는 시스템을 이용한다면 새로운 시스템을 구축하는데 비즈니스 모델, 비즈니스 로직과 GUI를 재사용할 수 있고 개발 기간은 매우 단축될 수 있다. 본 논문에서는 MFC 환경으로 구축되어 있는 기존 시스템을 COM 컴포넌트 모델로 변환하기 위한 변환 방법을 제시하고 그에 따른 소스코드의 재사용율을 측정하였다. 본 논문에서 제시하는 방법에 따른 컴포넌트 변환 방법은 신뢰성 있는 기존의 어플리케이션을 COM 컴포넌트 모델로 변환시 매우 높은 재사용율을 보인다.

1. 서 론

소프트웨어는 점점 복잡해지고 다양한 요구사항과 빠른 기간 내에 개발해야 하는 환경으로 변화하고 있다. 이에 컴포넌트 기반 개발을 수행함으로서 소프트웨어를 부품화하여 요구사항에 빠르게 대응할 수 있게 되었다.

컴포넌트 기반 개발(CBD : Component Based Development)은 소프트웨어를 개발의 적시성, 생산성 향상, 유지보수 및 확장성, 품질 향상 등에 많은 장점을 가지고 있다[1].

컴포넌트를 이용한 개발은 어플리케이션뿐만 아니라 웹 어플리케이션까지 반영되고 있다. 신뢰성을 보장받은 기존의 어플리케이션을 대상으로 컴포넌트를 추출하거나 컴포넌트화하는 행위는 비교적 빠른 구축을 이룰 수 있으며, 검증된 비즈니스 로직과 추가적으로 발생할 수 있는 장점을 가질 수 있다[2,3].

컴포넌트는 어플리케이션과 동적으로 결합하고 분리하는 컴포넌트의 기능에서 직접적으로 나온것이다. 이러한 기능을 가능하게 하기 위해서 동적으로 링크되어야 한다. 요구사항에 대하여 어플리케이션이 실행되고 있는 동안에도 교체될 수 있도록 해야 한다. 실행 시에 컴포넌트를 변화시키려면 컴포넌트들이 동적으로 링크되어야 한다. 또한, 컴포넌트가 어떻게 구현되어 있는지에 대한 구체적인 사항은 캡슐화해야 한다. 컴포넌트가 대체될 때 기존의 컴포넌트에 대한 연결을 해제하고 새로운 컴포넌트에 연결을 설정해야 한다. 컴포넌트와 다른 컴포넌트를 사용하기 위하여 연결하는 것을 인터페이스(interface)라고 한다. 동적 링크의 이점을 위해 컴포넌트는 인터페이스를 절대로 변경해서는 안된다. 이를 캡슐화되어야만 한다. 어떻게 구현되어 있는지 구체적인 것은 인터페이스에 반영되어서는 안된다.[4]

본 논문에서는 MFC로 구축되어 있는 기존 어플리케이션을 COM(Component Object Model) 컴포넌트 모델로 변환하기 위한 방법을 제시하고

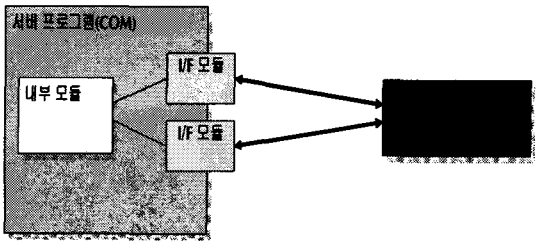
그에 따른 재사용율을 측정하였다.

2. 관련 연구

2.1 COM 컴포넌트 모델 구조

COM 이란 모듈 간의 통신 인터페이스로, OLE의 가장 기본이 되는 컴포넌트이다. OLE 란 객체간에 서로 연결하여 필요한 부분을 상호 교환하며 프로그램을 구동하는 방식을 의미한다.[4]

[그림 1]은 일반적인 COM의 개념도를 보이고 있다.



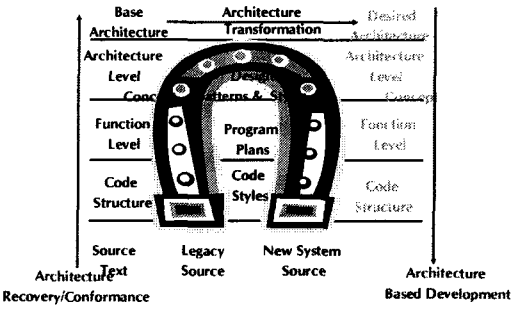
[그림 1] COM의 개념도

이러한 COM 과의 통신은 인터페이스에 의한 것인데 이 인터페이스의 변화는 곧 COM를 사용할 수 없게 된다는 뜻이다.

즉, 내부 모듈을 바꾸는 것은 아무런 제한이 없지만 인터페이스를 바꾸는 일은 발생하여서는 안된다. 따라서 COM을 개발할 때는 확장성을 고려하여 앞으로의 요구사항을 충분히 생각하여 인터페이스를 정해야 한다.

2.2 레가시 시스템으로부터 정보 추출

새로운 컴포넌트 개발을 할 때 기존에 이미 구축되어 신뢰성있는 시스템의 경우 어플리케이션으로부터 재사용 가능한 부분을 추출할 수 있다. 기존의 어플리케이션으로부터 정보를 추출하는 것은 개발 기간 감축과 새로운 프로그램의 신뢰도 향상을 가져 올 수 있다. 기존 레가시 시스템으로부터 얻을 수 있는 정보들은 [그림 2]와 같다[5]. 코드, 유저 인터페이스, 구조적부분, 디자인, 도메인 모델까지 모두 재사용이 가능하다. 모든 단계에서 재사용이 가능하지만 재사용했을 경우 재사용하지 않고 개발했을 경우와 비교를 해서 리소스나 비용면에서 얼마만큼 효율적인지 판단을 해야 한다.



[그림 2] 재공학 기반 개발 아키텍처

3. COM 컴포넌트 모델로 변환 기법

본 논문에서는 H 사에서 개발한 ERP 어플리케이션으로서 클라이언트 서버 환경에서 현재 사용되고 있는 시스템이며, 기존 시스템의 성능 및 재사용성을 고려하여 컴포넌트 기반의 시스템으로 변경하고자 한다.

기존 어플리케이션은 MFC로 개발되어 있으며, 클라이언트 부분은 표준 MFC 라이브러리를 사용하지 않았으며, 기능 단위 사용자 인터페이스 중심으로 모델링 하였다.

3.1 레가시 시스템을 이용한 역공학 단계

초기 개발사에서 제공한 개발 제안 요청서(RFP)와 제안서, 프로젝트 진행 계획서, 요구사항 명세서, 상세 설계서, 개발 코드, 데이터베이스 스키마, 운영자 매뉴얼을 획득했다. ERP 어플리케이션 중에서 향후 재사용 빈도가 높은 인사관리 모듈을 대상으로 하였다.

고수준의 컴포넌트를 획득하기 위한 재구성 활동으로 기존 인사 관리 코드로부터 식별할 수 있는 업무 기능의 산출물과 UI를 통해 식별되는 도메인 분석을 하였다. 비즈니스 로직을 추출하기 위해 Top-Down 방식으로 모델링을 하여 이를 통해 재구성을 한다.

3.2 COM 컴포넌트 모델 변환 매핑

COM 표준 컴포넌트 모델의 변환 대상을 H사의 ERP 어플리케이션 중에서 인사관리 모듈의 사원 정보 관리 도메인을 대상으로 하며, 기존 레가시 소프트웨어에 대한 역공학 산출물 과 재구성 활동을 통한 컴포넌트 식별 기준을 중심으로 사례 연구를 한다. [표 1]은 컴포넌트 식별단계에 따른 COM 컴포넌트 구성요소와의 매핑을 나타낸다[4].

[표 1] COM 컴포넌트 구성요소 변환 매핑 관계

항목	컴포넌트 식별 단계 구성 요소	COM 컴포넌트 구성 요소
변환지침	유스케이스 모델	COM 표준 객체
	유스케이스와 클래스 연관관계	COM 객체(IUnknwon 을 상속받는 구조체)
	초기 컴포넌트	COM 클라이언트 객체, COM 서버 객체(Factory 클래스)
	인터페이스	COM 기반 및 표준 인터페이스
	컴포넌트 연관 관계	COM 객체간 연관 관계
	컴포넌트 계층도	COM 클라이언트와 COM 서버로 구현
	컴포넌트 명세	COM 컴포넌트 명세

3.3 MFC 소스의 재사용

MFC에서는 기본적으로 View클래스과 Doc클래스, App클래스 등으로 구성되어 있다. 기존 H사의 ERP 어플리케이션에서는 View 클래스에 포함되어 있는 UI 관련 로직 부분과 데이터 구조를 선언해 놓은 구조체 헤더파일 부분이 재사용 가능하다는 것을 확인하였다.

[표 2] 데이터관련 구조체 변환 예

기존 소스

```
typedef struct tagFS_FGINDIVI
{
        long    lEmpCode;          //사원코드
        long    lDepCode;          //소속코드
        short   nPostCode;         //직책코드
        char    szIDCardNo[16];
//타임카드코드
        short   nAttFlag;
//근태계산대상
        short   nPayFlag;
//급여계산대상
        short   nDmyFlag[3];       //예비
                ..................................
                ..................................
        short   nZIPCode1;         //우편번호1
        short   nZIPCode2;         //우편번호2
        char    szAddress1[41]; //주소1
        char    szAddress2[41]; //주소2
        short   nEmployeeType; //관리범위
        short   nApprovalFlag;
//자기승인허가
} FS_FGINDIVI;
```

변환된 소스

```
create   proc usp_InsertIndivi
        @lEmpCode as int
,       @lDepCode as int
,       @nPostCode as smallint
,       @szIDCardNo as varchar(16)
,       @nAttFlag as smallint
,       @nPayFlag as smallint
,       @nDmyFlag0 as smallint
,       @nDmyFlag1 as smallint
        ..................................
        ..................................
,       @nZIPCode1 as smallint
,       @nZIPCode2 as smallint
,       @szAddress1 as varchar(41)
,       @szAddress2 as varchar(41)
,       @nEmployeeType as smallint
,       @nApprovalFlag as smallint
```

기존 ERP 어플리케이션은 데이터를 저장하거나 읽을 경우 구조체를 이용하였다. [표 2]는 기존 레가시 어플리케이션에서 데이터에 접근하기 위하여 사용되는 구조체들을 선언해 놓은 헤더 파일의 내용을 데이터 레벨의 스토어드 프로시저로 변경하는 예를 나타낸 것이다. 구조체내에 있는 변수들은 테이블의 필드명과 일치하게 선언되어 있다.

[표 3] 비즈니스 로직 변환 예

기존 소스

```
pEdit = (CEdit*)GetDlgItem(IDC_A1E_EMPCODE);
pEdit->GetData(&pDoc-
>m_setIndivid.m_data.lEmpCode);
pCombo =
(CComb*)GetDlgItem(IDC_A1C_DEPCODE);
pCombo->GetCurCode(&pDoc->m_setIndivi.m_data.
lDepCode);
pCombo = (CComb*)GetDlgItem(IDC_A1C_POST);
pCombo->GetCurCode(&pDoc->m_setIndivid.m_data.
nPostCode );
pEdit = (CEdit*)GetDlgItem(IDC_A1E_CARDNO);
```

변환된 소스

```
parm = adoCmd-
>CreateParameter(_bstr_t("lEmpCode"), adInteger,
adParamInput, 8 , lEmpCode);
parm = adoCmd-
>CreateParameter(_bstr_t("lDepCode"), adInteger,
adParamInput, 8 , lDepCode);
parm = adoCmd-
>CreateParameter(_bstr_t("nPostCode"), adInteger,
adParamInput, 4 , nPostCode);
parm = adoCmd->CreateParameter
( _bstr_t("szIDCardNo"), adVarChar, adParamInput,
16, szIDCardNo);
```

기존 어플리케이션은 에디트, 콤보, 라디오 등의 UI 컨트롤로부터 얻은 값을 해당 테이블과 매핑되는 구조체를 통해 데이터 베이스에 저장된다.

UI 컨트롤로부터 얻어지는 값들은 COM 컴포넌트에 전달되어 지는 인자와 매핑될 수 있으므로 [표 3]과 같이 변환되어질 수 있다.

[표 3]은 인사관리 공통부분에서 UI를 구성하는 View 클래스로부터 도메인 로직을 포함하는 부분을 컴포넌트에서 사용할 수 있는 형태로 추출하는 과정의 예를 보여준다.

3.4 평가

[표 4] 인사관리 공통 모듈 변환 평가표

단위 : LOC

모듈명	관련파일		소스 크기	재사용
인사관리 공통	기존파일	FGINDIVI.h	136	34
	변환 파일	usp_InsertIndivi	85	
인사기본 정보관리	기존파일	Vg1000 view class	1019	123
	변환파일	HPersonalMgmt.Modify	297	
인사자력 관리	기존파일	Vp1400 view class	2495	442
	변환파일	HPersonalInfoMgmt.Modify	1433	
인사근무 사항	기존파일	Vp1450 view class	4198	1346
	변환파일	HPersonalWorkMgmt.Modify	2934	

COM 컴포넌트를 제작할 경우 MFC로 제작되어 있는 기존 어플리케이션의 기존 소스를 COM 컴포넌트 기반의 새로운 시스템에서 35%정도를 재사용할 수 있었다.

4. 결론 및 향후 연구

기존 어플리케이션을 컴포넌트 기반으로 변화시키려면 신뢰성, 개발기간 등 많은 어려움이 따른다. 컴포넌트 기반 개발시 기존의 레가시 어플리케이션으로부터 재사용성을 추출하여 개발을 한다면 보다 빠르게 시스템을 구축할 수 있고 신뢰성 있는 시스템을 개발할 수 있다.

본 논문에서는 컴포넌트 기반 개발에서 생산성을 확보하고, 신뢰성 있는 어플리케이션을 개발하기 위한 방법으로 기존에 구축되어 있는 어플리케이션에서 재사용율을 측정하고 컴포넌트화 하는 기법을 제시하고자 하였다. COM 컴포넌트 모델을 선택함으로 윈도우기반의 어플리케이션에서 언어에 상관없이 공통적으로 사용할 수 있도록 하였다.

COM 컴포넌트 변환 기법은 기존의 모든 리소스를 바탕으로 어플리케이션을 분석, 변환하여 재사용한다. MFC로 제작되어진 기존 어플리케이션의 경우 모델의 재사용뿐 아니라 소스의 재사용성율도 높기 때문에 기존 소스에 대한 재활용도 매우 높은 것을 확인하였다.

비즈니스 로직을 재사용할 수는 있으나 많은 부분이 전문가의 해석이 필요한 수동으로 이루어지고 있다. 이에 필요한 부분을 자동으로 제작화 할 수 있는 CASE Tool이 있다면 소프트웨어의 재사용을 원활히 할 수 있으리라 기대된다.

5. 참고문헌

[1] Wilkes, Lawrence, Understanding Component Based Development, Addison-Wesley, June, 2000.

[2] Luqi, Jiang Guo, "Toward Automated Retrieve for a Software Component Repository", IEEE Conference and Workshop on Engineering of Computer-based Systems, March, 1999.

[3] C.W.Krueger, "Software Reuse", ACM Computing Survey, Vol. 24, No. 2, pp.131~184, June, 1992.

[4] Microsoft, The Component Object Model Specification, at URL: http://www.microsoft.com/com/resources/comdocs.asp

[5] Requirements for Integration Software Architecture And Reengineering Models, 1998. IEEE.