레거시 시스템을 위한 EJB 래퍼 설계 및 구현

이 문 수, 양 영 종
한국전자통신연구원(ETRI)
(mslee, yjyang)@etri.re.kr

The Design And Implementation Of EJB Wrapper For Legacy System

Moon-Soo Lee[0] Young-Jong Yang

Dept. of ETRI-Computer & Software Technology Lab.

요 약

레거시 시스템은 수년간 각 기업의 비즈니스 환경에 따라 최적화되어 있다. 국부적인 영역에만 적용되어진 레거시 시스템은 웹과 같은 분산환경에서 더 이상 사용하기 어려워졌다. 따라서 레거시 시스템으로부터 중요한 비즈니스 로직을 식별하여 재사용할 수 있는 메커니즘이 필요하다.

본 논문에서는 레거시 시스템을 컴포넌트 래핑 기술을 이용하여 엔터프라이즈 자바 빈즈(EJB)로 생성하는 지원도구를 설계 및 구현하였다. 본 논문은 비즈니스 로직을 이벤트와 데이터 분석과 관계를 이용여 여 레거시 컴포넌트를 식별하는 방법을 제시하고 이러한 방법으로 추출된 컴포넌트를 EJB환경에서 재사용 가능한 프레임웍 기반의 래퍼 구조를 제시하고자 한다.

1. 서론

오늘날 중요한 비즈니스 정보를 가지고 있는 과거의 모놀리딕 어플리케이션은 초창기 프로그램 언어인 COBOL과 PL/I과 같은 언어를 사용하여 개발 되어왔다. 이러한 어플리케이션들은 소프트웨어 유지 관리(Maintenance)를 통하여 계속 변경되어 왔지만 문서화는 그다지 잘 이루어지지 못했다. 또한 프로그램 인력의 부족은 시스템의 유지 관리 비용을 계속 증가시켰다. 웹 서비스와 같이 계속적인 기업 요구 사항의 맞추고 유지 관리 비용을 줄이기 위해 레거시 시스템의 진화(evolution)은 불가피하게 되었다.

소프트웨어 진화의 해결책은 재개발, 래핑, 이전(Migration), 변환(Transformation)과 같은 몇 개로 나누어 볼 수 있다.[1,2] 이들 중에서 래핑은 가장 비용이 최소화 하면서 효과적인 방법이 될 수 있다. 래핑은 단 기간에 최소의 비용으로 레거시 시스템의 장점인 안정성과 신뢰성을 그대로 유지 할 수 있다는 이점을 가지고 있다. 하지만 스크린 스크래핑(Screen Scraping)과 같은 소프트웨어 래핑은 임시 변동의 해결책일 뿐 비즈니스 요구 사항이 변경되거나 추가될 경우에는 오히려 더욱 시스템을 복잡하게 만들 수 있다. 스크린 스크래핑은 주로 UI에 관련된 부분의 현대화(Modernization)에 이용되었기 때문에 더 이상 재사용되지도 진화를 할 수 없게 되었다.

인터넷과 같은 분산 환경에서의 소프트웨어 재사용에 대한 요구는 컴포넌트 기반의 소프트웨어 개발(CBD)방법이 주목을 받게 되었다.. 썬(Sun)은 트랜잭션, 영구성(persistence), 보안등 기반 정보 서비스를 표준화된 인터페이스를 제공하는 엔트프라이즈 자바 빈즈(EJB) 아키텍쳐를 발표하였다. EJB는 초창기에는 기존의 CICS나 SAP, MQseries등과 같은 시스템에 대한 내용은 간과되었다. 하지만 결국 최근에 썬(Sun)은 J2EE 커넥터 아키텍쳐를 통하여 레거시 시스템을 수용하게 되었다.

이 논문에서는 레거시 시스템에서 기능 중심으로 재사용 가능한 부분을 식별하여 스크린 스크래핑의 문제점을 보완한 EJB 컴포넌트 래핑 도구를 구현한다. 2장에서는 시스템의 전체 구조를 설명하고 3장에서는 레거시 시스템으로부터 제안된 식별 알고리즘을 통하여 재사용 가능한 부분을 식별한다. 4장에서는 레거시 컴포넌트 래퍼의 구조를 설명하고 5장에서는 결론과 향후 연구방향을 제시한다.

2. 시스템 전체 구조

컴포넌트 래퍼를 생성하는 프로로타입 도구는 그림 1과 같다. 이 도구은 전체적으로 3개의 모듈로 구성되어 있다.

코드분석기는 기존 COBOL 소스 프로그램과 화면 정보를 가지고 있는 BMS Map파일을 파싱하여 정보를 추출한다. 분석기는 어휘분석을 위해 JFlex를 이용하였고 CICS COBOL의 경우 각 Division에 사용되는 예약어가 각기 다른 용도로 사용되므로 구문 분석은 각 Division을 나누어 파서를 구현하였다.

컴포넌트 후보 식별기는 기존 시스템에서 재사용 가능한 부분의 비즈니스 로직이 포함되어 있는 컴포넌트 후보를 식별해낸다. 컴포넌트 후보는 제안된 식별 방법을 이용한다.

EJB 래퍼 생성기는 EJB 프레임웍과 중간 프레임웍인 래핑 프레임웍을 자동 생성해 준다. 자동 생성된 레거시 컴포넌트는 유상태(stateful) 세션 빈으로 생성되며

EJB 트랜잭션 서비스를 IBM CICS 시스템과 동일하게 유지하기 위해 트랜잭션 단위로 래핑한다.

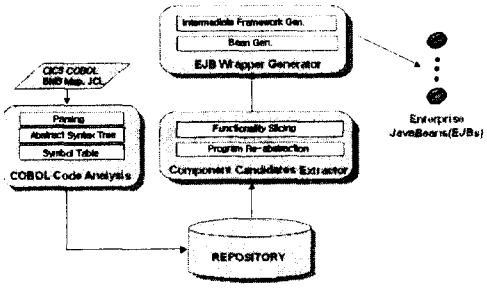

그림 1. 전체 시스템 구조

3. 레거시 컴포넌트 식별

비즈니스 로직 식별을 위해 소스코드 이해 및 기능 재사용에 관해 많은 연구가 되어왔다. 그 결과는 주로 유지보수[7,8,9,10,11]와 데이터 흐름, 제어흐름과 호출관계등과 같은 재 문서화[6,7]에 사용되었다.

비즈니스 로직 식별 방법에는 입력,출력, 치환, 조건 등 4개가지 요소로 이루어져 있으며 이들 요소들의 집합을 식별하는 것이다.[9] 다른 접근 방법은 의존성(dependency)분석, 변수 분류, 프로그램 슬라이싱을 이용한 데이터 중심으로 식별하는 것이다.[11]

CICS COBOL 시스템은 시스템의 자원을 효율적으로 이용하기 위해 Pseudo-Conversation방식으로 프로그램을 하게 된다. Pseudo-Conversation은 그림 2와 같이 각 키보드 입력에 따라 처리할 루틴의 시작 파라그래프명을 등록하게 된다. 따라서 재사용 가능한 비즈니스 로직은 이러한 키보드 이벤트와 밀접한 관계를 가지고 있다.

```
*-----------------------------------------------------------------
PROCEDURE       DIVISION.
*-----------------------------------------------------------------
START-OF-PROGRAM.
  EXEC CICS HANDLE CONDITION
            MAPFAIL (MAPFAIL-ERROR)
            DUPREC (DUPREC-ERROR)
            PGMIDERR(PGMIDERR-ERROR)
            NOTOPEN (OPEN-ERROR)
            ERROR (UNDEFINED-ERROR)
            END-EXEC.

  EXEC CICS HANDLE AID
            ENTER (ENTER-KEY)
            PF3   (PF3-KEY)
            PF4   (PF4-KEY)
            PF10  (PF10-KEY)
            PF11  (PF11-KEY)
            ANYKEY (ANY-KEY)
            END-EXEC.

  IF KD-SUDO = 'Y'
    MOVE  LOW-VALUE  TO DAMF4210
                :
                :
*-----------------------------------------------------------------
ENTER-KEY.
*-----------------------------------------------------------------
  PERFORM  KEY-EDIT  THRU  KEY-EDIT-EXIT.
  IF EDIT-FLAG NOT = SPACE
    PERFORM SEND-ALARM  THRU  SEND-ALARM-EXIT
    MOVE  PGMMSG-04 TO MSG0
    PERFORM SEND-MAPSET THRU  SEND-MAP-EXIT
    PERFORM RETURN-TRANSID THRU RETURN-TRANSID-EXIT.
                :
                :
```

그림 2. Procedure Division 선언 부분

본 논문의 식별 방법은 이벤트와 데이터 흐름을 중심으로 레거시 컴포넌트 후보를 식별하는 것이다. 그 과정은 정리하면 다음과 같다.

Step1. 변수나 파라그래프와 같은 식별자(Identifier)들을 분류를 한다. COBOL에서 모든 변수는 전역변수로 선언되어 있다. 따라서 변수의 가시성을 구별하는 메커니즘을 가지고 있지 않는 COBOL 프로그램을 입력 변수, 출력 변수, 제어변수 그리고 파라그래프 이름들로 식별자 분류하는 것은 매우 중요하다.

Step2. 그림 3과 같이 파라그래프와 제어 변수들의 관계를 나타내는 Call tree를 생성한다. Call tree의 시작점은 ENTER, PF3나 PF4등과 같이 키보드 이벤트가 된다. 파라그래프의 흐름은 프로그램 재추상화(Re-abstraction)를 통해 프로그램의 계층적 구조관계를 적용하여 표현하게 된다.

그림 3. Call tree 그래프

Step3. Step1에서 분류된 변수 중 사용자와 관련을 가지는 입력과 출력 데이터들의 흐름을 이용하여 레거시 컴포넌트 후보들을 식별해낸다. 식별된 후보들을 하나의 흐름을 가지고 있으면 입력이나 출력 데이터를 가질 수 있다. 이러한 입출력 데이터는 EJB Remote 인터페이스에 set메소드나 get메소드를 이용하여 정의된다.

Step4. 소스 코드내의 HANDLE 명령문을 제거한다. 레거시 CICS시스템은 이벤트나 에러처리를 위해 HANDLE AID, HANDLE CONDITION나 HANDLE ABEND(그림 2)와 같은 HANDLE 명령문을 사용한다. 에러 처리 부분은 상당히 복잡하기 때문에 수작업으로 소스코드를 수정해야 한다.

Step5. COMMAREA 영역을 수정한다. COMMAREA는 외부프로그램이 CICS 프로그램을 호출하여 데이터를 전달하는 공통 메모리 영역이다. 따라서 입출력 데이터를 주고 받기 위해서는 COMMAREA부분에 레코드와 필드가 정의 되어야 한다. 우리는 COMMAREA 부분은 그림 4와 같이 세 부분으로 나뉘어진다. 첫번째 부분은 호출될 프로그램들을 위한 기존 메모리 정의가 된다. 두 번째는 입출력 정보를 가지고 있는 Map의 Symbolic 부분을 정의한다. 마지막 부분은 예외처리와 같은 내부 프로세서의 인터셉트를 위한 메시지와 수정된 HANDLE 명령문과 관련을 가지고 있는 추가 필드에 대해 정의하는 부분이다.

그림 4. 수정된 COMMAREA

4. 레거시 래퍼 구조

소프트웨어 래핑은 기존 시스템을 이전할 때 가장 안정적이고 효율적인 해결책이다. 하지만 래핑은 일시적인 방편일 뿐 더 이상 재사용하지 못하게 된다. 이는 프레임웍에 대한 고려보다는 UI부분의 현대화(Modernization)에만 집중되어 있기 때문이다.

기존 시스템을 EJB시스템과 조합하는데 있어서 각 프레임웍의 격차(gap)과 레거시 시스템의 접근성(accessibility)에 대한 해결이 매우 중요하다.[5] 제안된 레거시 래퍼 아키텍처는 중간 프레임웍을 이용하여 그림5와 같이 계층적 구조로 이루어 진다.

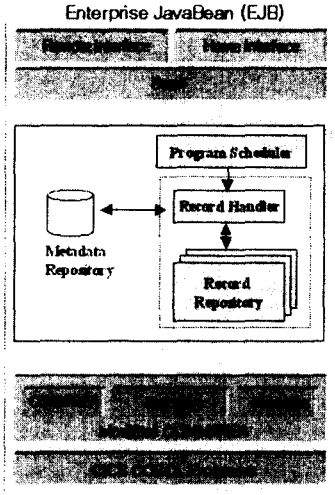

그림 5. 레거시 래퍼 구조

래퍼 아키텍처는 EJB 프레임웍, 중간 프레임웍, 레거시 프레임웍으로 이루어져 있고 프레임웍의 격차를 해결하기 위해 중간 프레임웍이 들어가게 되고 Map정보와 프로그램 인터셉트를 위한 메시지부분을 추가하여 레거시 프레임웍의 접근성을 용이하게 해준다.

중간 계층의 프레임웍은 다음 네 부분들로 구성된다.

레코드 저장소는 ASCII코드와 EBCDIC코드의 상호 변환하는 기능과 EBJ시스템이 CICS 시스템으로 데이터를 전달하기 위한 일시적인 저장소 역할을 담당한다.

메타 저장소는 COBOL의 필드들이나 레코드들에 대한 메타 정보를 가지고 있다. 메타정보는 상위 레코드와 필드의 조합으로 고유한 키를 가지게 되고 그에 따른 메타 객체를 해쉬테이블(Hashtable)로 가지고 있다. 그리고 메타 저장소는 런타임으로 메타정보를 제공한다.

레코드 핸들러는 EJB로부터의 명령을 해석하여 메타 저장소로부터 런타임으로 메타정보를 얻는다. 그리고 레코드 저장소로부터 원하는 정보를 추출해낸다.

프로그램 스케쥴러는 하나의 트랜잭션에 포함되어 있는 연결성을 가지고 있는 프로그램들의 항해정보를 가지고 있다.

5. 결론

본 논문은 컴포넌트 기반 소프트웨어 개발에 있어서 IBM CICS COBOL의 레거시 시스템을 래핑하여 재사용할 수 있기 위해 컴포넌트 후보 식별방법을 제안하고 프레임웍 기반의 래퍼 구조를 설계하였다. 제안된 프레임웍 기반의 래퍼는 소프트웨어의 재사용성을 높여 주고 프레임웍 조합 시 생기는 문제점을 해결하였다. 현재 지원 도구는 이벤트와 데이터 중심으로 컴포넌트 후보를 식별한다. 하지만 제안된 식별 방법은 완전한 식별 방법을 제시하지 못하므로 다양한 식별 방법에 대해 앞으로 보다 많은 연구가 필요하다.

6. 참고문헌

1. J. Bisbal and D. Lawiess and Bing Wu, Legacy Information Systems : Issues and Directions, IEEE Sotware, pp 103-111, 1999

2. Harry M. Sneed and Rudolf Majnar, A Case Study in Software Wrapping, IEEE Software, pp86-92, 1998

3. Melody M. Moore, Rule-Based Detection for Reverse Engineering User Interfaces, IEEE Proceedings of WCRE'96, pp.42-48, 1996

4. Michael Mattsson and Jan Bosch, Framework Composition: Problem, Causes and Solutions, IEEE Computer, pp.203-214, 1998

5. Elizabeth Burd and Malcolm Munro, Analysing Large COBOL Programs: the extraction of reusable modules, IEEE Computer, pp.238-243, 1996

6. Elizabeth Burd and Malcolm Munro, Extracting Reusable Modules from Legacy code: Considering the Issues of Module Granularity, Proceedings of WCRE'96, pp.189-196, 1996

7. Harry M. Sneed and Erika Nyary, Extracting Object-Oriented Specification from Procedurally Oriented Programs, IEEE Computer, pp.217-226, 1995

8. H.Huang and W.T.Tsai, Business Rule Extraction form Legacy Code, IEEE Computer, pp.162-167, 1996

9. Harry M. Sneed and Katalin Erdos, Extracting Business Rules from Source Code, IEEE Compute, pp. 240-247, 1996

10. J. K. Joiner and W. T. Tsai, Data-Centered Program Understanding, IEEE Computer, pp.272-281, 1994

11. Jon Beck and David Eichmann, Program and Interface Slicing for Reverse Engineering, IEEE Computer, pp.509-518, 1993