제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

COBOL 시스템을 위한 EJB 래퍼 컴포넌트 지원 도구 개발

정효택, 김동관
한국전자통신연구원 컴퓨터.소프트웨어연구소
S/W.컨텐츠연구부 S/W 재사용연구팀
e-mail : {htjung, dgkim}@etri.re.kr

COBOL2EJB : A Wrapper Component Supporting Tool for COBOL System

Hyo-Taeg Jung, Dong-Kwan Kim
Dept. of Software.Contents Technology, CSRL, ETRI

요 약

최근 컴퓨팅 환경이 변화함에 따라 현재 운용되고 있는 레거시(Legacy) 시스템을 웹 환경과 같은 새로운 환경에서도 운용할 수 있도록 시스템을 현대화(Modernization) 하고자 하는 연구가 활발히 진행되고 있다. 특히 설계 패턴, 프레임워크 등의 소프트웨어 재사용 기술과 함께 컴포넌트 기술이 개발, 보급됨에 따라 기존 시스템을 컴포넌트로 변환하거나 연계하는 방법들이 연구되고 있다. 본 논문에서는 IBM 메인 프레임에서 운용되고 있는 CICS 코볼 시스템을 EJB 래퍼 컴포넌트로 연계하는 일련의 프로세스를 지원하는 연계 도구에 대한 내용으로 도구의 구성과 기능을 상세히 설명하고 있다.

1. 서론

기존에 운용되고 있는 레거시(Legacy) 시스템은 웹과 같은 새로운 컴퓨팅 기술 및 환경에 적응하기 위하여 진화(Evolution) 되어야 한다. 본래 시스템의 진화는 데이터베이스의 한 필드를 갱신하는 것부터 시스템 전체를 재구현(re-implementation) 하는 것까지를 이르는 매우 광범위한 용어로 사용된다. 시스템의 진화는 유지 보수(Maintenance), 대치(Replacement), 현대화(Modernization)로 크게 나눌 수 있다[1]. 시스템 유지 보수는 시스템의 구조에는 아무런 변경 없이 시스템의 버그를 수정하거나 기능을 향상시키는 등 점진적이고(incremental) 반복적인 (iterative) 과정으로 이루어지며 단점으로는 시간이 경과할수록 비용이 증가한다는 것이다. 시스템 대치는 시스템의 현대화에 비용이 너무 소요되거나 혹은 시스템 자체의 문서가 부족하거나 기술이 낙후되어 확장이 불가능 할 경우에 적당한 방법으로서 처음부터 시스템을 재구축하는 과정으로 시스템의 안정성을 확보하지 못하는 단점이 있다. 시스템 현대화는 유지 보수와 대치의 중간적인 입장에서 시스템을 재구조화 하거나 특정 기능을 향상시키거나 새로운 기술을 접목하는 등 기존 레거시 시스템 전반에 걸쳐 변경을 가하는 방법이다.

시스템을 현대화하는 방법으로는 크게 화이트 박스 현대화(White-box Modernization) 와 블랙 박스 현대화(Black-box Modernization) 으로 구분할 수 있다. 화이트 박스 현대화는 시스템 내부의 역공학 과정을 통하여 프로그램을 이해하고 이를 바탕으로 시스템이나 코드를 재구조화 함을 말하며, 블랙 박스 현대화는 시스템의 내부 로직과는 관계없이 입출력과 같은 시스템 인터페이스를 이해하고 이를 바탕으로 레거시 시스템을 래핑 등의 방법으로 레이어링(Layering) 함을 말한다. 블랙 박스 현대화는 종종 실제적이지 않거나 화이트 박스 현대화처럼 시스템 내부의 모듈에 대한 철저한 이해가 필요한 경우가 있다.

대상 시스템에 따른 현대화 기법으로는 크게 사용자 인터페이스의 현대화(User Interface Modernization), 데이터의 현대화(Data Modernization), 기능(로직)의 현

대화(Functional(logic) Modernization) 등으로 구분할 수 있다. 데이터를 현대화하는 기법으로는 ODBC, JDBC 등과 같이 데이터베이스 게이트웨이를 사용하거나 XML 을 이용하여 통합(Integration)하거나 데이터를 분산시키는 데이터 복제(Data Replication) 등의 기법이 있다. 기능(로직)을 현대화하는 기법으로는 CGI 확장을 이용한 기법, 객체 기반의 래핑 기법, 컴포넌트 래핑 기법 등이 있다.

시스템 현대화를 위해서는 레거시 시스템의 비즈니스 로직 일부를 컴포넌트화 하는 것이 일반적인데, 그 방법으로는 전이(Legacy Migration), 래핑(Wrapping with Components Interface), 재구조화(Legacy Restructuring), 향상(Legacy Enhancement), 통합(Integration via Interfaces and Component Interfaces) 등이 있다[2].

본 논문에서는 IBM 메인 프레임에서 운용되고 있는 CICS 코볼 시스템을 EJB 래퍼 컴포넌트로 연계하는 일련의 프로세스를 지원하는 연계 도구(일명 COBOL2EJB)에 대한 내용으로, 2 장에서는 시스템의 전체적인 구성을 설명하고, 3 장 ~ 6 장은 각 서브 모듈의 기능과 특성을 소개한다. 7 장에서는 시스템 환경을 소개하고 8 장에서는 결론 및 향후 연구 방향을 제시한다.

2. COBOL2EJB 시스템 구성

COBOL2EJB 연계 도구는 현재 IBM 메인 프레임에서 운용되고 있는 CICS 코볼 시스템을 EJB 래퍼 컴포넌트로 연계하는 일련의 프로세스를 지원하는 연계 도구로서, COBOL 코드 분석기, 시각화 정보 생성기, EJB 래퍼 컴포넌트 생성기, 컴포넌트 시험기로 구성되어 있다 (그림 1).

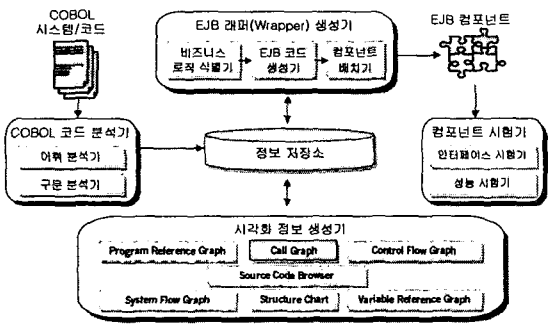

그림 1. 시스템 구성도

COBOL2EJB 연계 도구의 동작 순서를 살펴 보면, 먼저 COBOL 시스템을 입력으로 받아 소스 코드를 분석하고, 시각화 정보 생성기에서 제공하는 여러 그래프를 활용하여 시스템에 대한 이해를 높인다. EJB 래퍼 컴포넌트를 생성, 실행하기 위하여 컴포넌트화 할 대상 비즈니스 로직을 식별하고, EJB 코드를 생성한 다음 이를 웹 어플리케이션 서버에 배치한다. 생성된 컴포넌트는 컴포넌트 시험기를 통하여 인터페이스와 성능을 테스트한다.

3. COBOL 코드 분석기

CICS 코볼 시스템은 소스 코드 이외에도 관련된 맵화일(Map file), 카피북(Copy book), JCL 등 여러 요소로 구성되어 있다. 코볼 코드 분석기에서는 이러한 요소들을 입력으로 받아 어휘 분석, 구문 분석 과정을 거쳐 AST, 심볼 테이블 등을 생성하고 이를 저장한다 (그림 2). 코드 분석기에서 분석된 정보는 시각화 정보 생성기 및 EJB 래퍼 컴포넌트 생성기 등에서 사용된다.

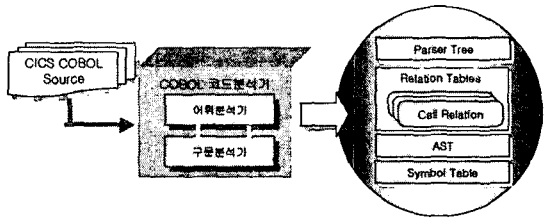

그림 2. COBOL 코드 분석기 구성도

4. 시각화 정보 생성기

시각화 정보 생성기는 코드 분석기에서 분석한 정보를 이용하여 입력 시스템에 대한 정보를 각종 그래프를 통하여 시각적으로 제공해 줌으로써 입력 시스템에 대한 이해를 증진 시킨다. 제공되는 그래프는 시스템 레벨의 시스템 흐름 그래프, 프로그램 참조 그래프가 있으며 프로그램 레벨에서는 호출 그래프, 구조도, 제어 흐름 그래프, 변수 참조 그래프 등이 있으며 그외 소스 코드 브라우저, 맵화일 뷰어 등이 있다.

4.1. 시스템 흐름 그래프(System Flow Graph)

시스템 흐름 그래프는 시스템 레벨에서 프로그램간의 논리적 흐름을 보여 준다. 소스 프로그램, 화면 정보를 가지고 있는 맵화일, 그리고 트랜잭션 ID 로 구성되어 있어서, 트랜잭션과 관련된 프로그램 및 그 흐름을 파악할 수 있다 (그림 3 의 왼쪽 위).

4.2. 프로그램 참조 그래프(Program Reference Graph)

프로그램 참조 그래프는 시스템 레벨에서 프로그램간의 물리적 구조를 보여 준다. 소스 프로그램과 관련된 맵세트, 카피북, 그리고 플랫 데이터베이스 등으로 구성되어 있다 (그림 3 의 왼쪽 아래).

4.3. 호출 그래프 (Call Graph)

호출 그래프는 하나의 프로그램내에서 패러그래프간의 호출 관계를 보여 준다. 구성 요소로는 프로그램과 패러그래이며 호출 관계는 Perform 과 Goto 호출로 구분된다 (그림 3 의 오른쪽).

4.4. 제어 흐름 그래프 (Control Flow Graph)

제어 흐름 그래프는 한 패러그래프 내에서 스테이트먼트의 흐름을 보여 준다. 그래프내의 숫자는 소스 코드의 라인 번호를 나타내며 마름모는 IF 문의 분기점을 표시한다 (그림 4 의 왼쪽).

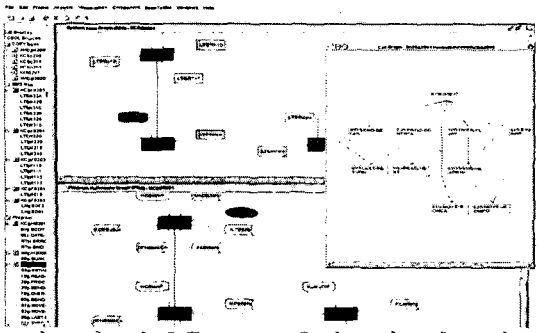

그림 3. 시스템 흐름, 프로그램 참조 및 호출 그래프

4.5. 구조도 (Structure Chart)

구조도는 프로그램 내에서 패러그래프와 변수들의 관계를 보여 준다. 구성 요소는 프로그램, 패러그래프, 변수이며 화살표의 방향은 변수가 패러그래프에서 사용되거나 선언됨을 표시한다 (그림 4 의 오른쪽 위).

4.6. 변수 참조 그래프 (Variable Reference Graph)

변수 참조 그래프는 임의의 변수가 사용되는 패러그래프들을 보여 준다. 즉 변수가 어느 패러그래프에서 사용되었는지를 표시한다 (그림 4 의 오른쪽 아래).

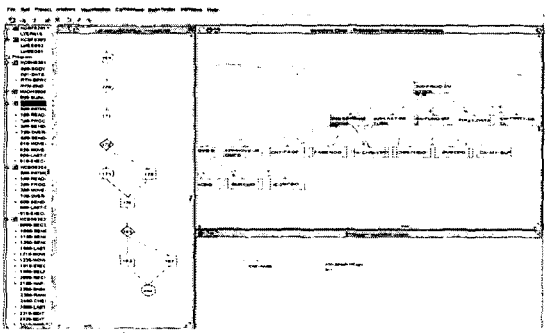

그림 4. 제어 흐름 그래프, 구조도, 변수 참조 그래프

4.7. 소스 코드 브라우저 (Source Code Browser)

소스 코드 브라우저는 소스 코드를 트리 형태로 보여 준다. 내용을 펼치거나 접을 수 있어서 특정 부분으로 쉽게 이동이 가능하며 선별하여 보기에 용이하다 (그림 5 의 왼쪽).

4.8. 맵화일 뷰어 (Map file Viewer)

맵화일 뷰어는 화면 정보를 가지고 있는 맵화일을 분석하여 어떤 화면인지를 보여 준다. (그림 5 의 오른쪽).

5. EJB 생성기

EJB 생성기는 컴포넌트화 대상 비즈니스 로직을 식별하는 식별기, 식별된 비즈니스 로직을 EJB 컴포넌트로 생성하는 생성기, 생성된 EJB 컴포넌트를 웹 어플리케이션 서버에 배치하는 배치기로 구성되어 있다.

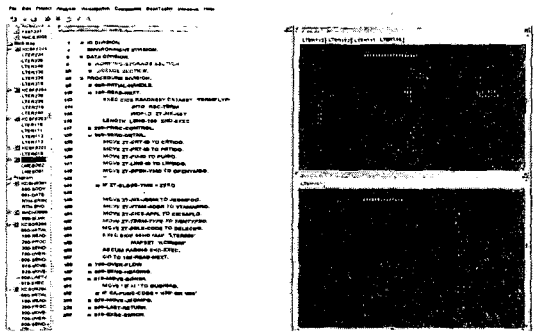

그림 5. 소스 코드 브라우저와 맵 뷰어

5.1. 비즈니스 로직 식별기

비즈니스 로직을 이해하고 식별하는 여러 방법들이 연구되어 소개되었지만[3][4][5], 식별은 물론 검증도 쉽지 않다.

비즈니스 로직 식별기가 변수들을 입력, 출력, 외부 변수로 분류하게 되면 사용자는 재사용하고자 하는 비즈니스 로직의 유형을 가중치를 사용하여 선택하여 슬라이싱 판별 기준으로 설정한다 (그림 6 왼쪽 위). 다음 단계로 사용자는 프로그램 단위, 패러그래프 단위로 후보 로직을 선택하면 (그림 6 오른쪽 위, 왼쪽 아래) 식별기는 호출 트리(Call Tree)를 이용하여 선택한 패러그래프의 실행 경로를 보여 주며 (그림 6 오른쪽 아래) 내부적인 영향 분석(Impact Analysis)를 통하여 재사용 가능한 워크플로우를 자동으로 식별해 준다.

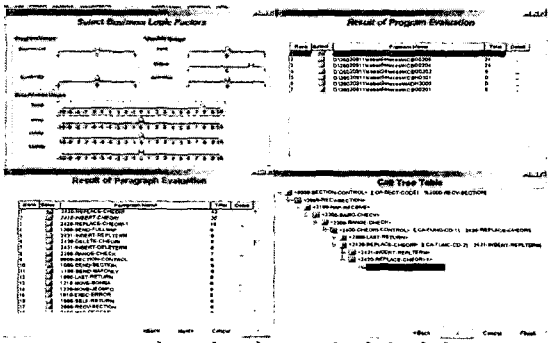

그림 6. 비즈니스 로직 식별 과정

5.2. EJB 코드 생성기

EJB 코드를 생성하기 위해서는 먼저 EJB 래퍼 컴포넌트의 인터페이스를 정의하게 되는데, 래퍼 컴포넌트의 인터페이스는 식별된 비즈니스 로직의 화면 정보를 이용하게 된다. 다음은 식별된 비즈니스 로직과 EJB 컴포넌트를 연계할 수 있는 연결 코드인 레코드 프레임워크를 자동으로 생성하게 된다. 또한 생성될 EJB 컴포넌트의 리모트 인터페이스 이름과 CTG(CICS Transaction Gateway), CICS 서버, 트랜잭션 ID 등 메인 프레임의 각종 환경 정보를 설정함으로써 EJB 소스 코드가 자동으로 생성된다. 마지막으로 EJB 코드와

연계 코드를 컴파일 함으로써 웹 서버에 배치할 준비가 완료되며, 이러한 모든 기능을 EJB 코드 생성기가 실행하여 준다.

5.3. EJB 배치기

생성된 EJB 컴포넌트를 웹 어플리케이션 서버에 배치하는 역할을 한다.

6. 컴포넌트 시험기

컴포넌트 시험기는 서버에서 운용되는 EJB 컴포넌트에 대한 인터페이스 테스트와 성능 테스트를 수행하는데, 중요한 특징으로는 Java Reflection 을 이용하여 테스트 케이스를 정의하고, 패턴 및 XML 을 이용하여 테스트 데이터를 생성하며, 테스트 클라이언트 프로그램을 자동으로 생성하며, 쓰레드 및 Individual JVM 을 이용하여 성능 테스트를 한다.

시험기에서는 EJB 컴포넌트의 최소, 최대, 평균 응답시간을 수치와 차트로 제공한다. 또한 테스트 케이스를 실행하여 반환되는 값이 예상되는 값과 다를 경우 차이를 보여 주며, 동시 클라이언트 수에 따른 컴포넌트의 응답 시간을 그래프로, 또한 응답시간의 분포를 Hi-Lo 차트를 통하여 보여 준다 (그림 7).

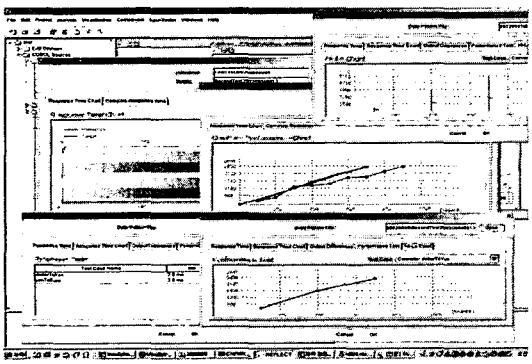

그림 7. 컴포넌트 시험기의 결과 리포팅

7. COBOL2EJB 시스템 환경

COBOL2EJB 연계 도구는 (그림 8)과 같은 목표 환경을 지원한다. 즉 생성된 EJB 래퍼 컴포넌트는 웹 어플리케이션 서버에 배치되어 CTG 를 통하여 CICS 메인 프레임에 있는 Legacy COBOL 시스템과 연동하게 된다.

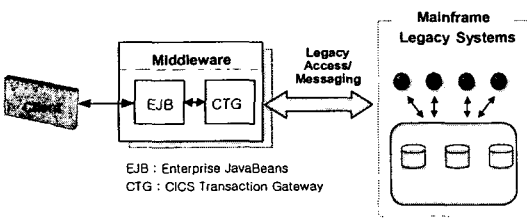

그림 8. 목표 시스템 환경

COBOL2EJB 의 개발 및 운용 환경은 다음과 같다.

- 메인 프레임 : OS/390 v2.7, CICS v1.3
- 미들웨어 : Windows 2000/NT 서버, IBM CTG, IBM Websphererv3.5
- 개발언어 : JDK 1.2 이상

8. 결론 및 향후 과제

COBOL2EJB 연계 도구는 IBM 메인 프레임에서 운용되고 있는 CICS 코볼 시스템을 EJB 래퍼 컴포넌트로 연계하는 일련의 프로세스를 지원하는 연계 도구로서, COBOL 코드 분석기, 시각화 정보 생성기, EJB 래퍼 컴포넌트 생성기, 컴포넌트 시험기로 구성되어 있다. 래퍼 컴포넌트를 생성하여 기존 레거시 시스템과 연동함으로써 시스템의 변경 없이도 새로운 환경에서 쉽게 운용함은 물론 새로운 기능에 대한 추가도 컴포넌트를 확장함으로써 가능하다.

향후 과제로는 시스템의 비즈니스 로직을 확장하거나 시스템의 구조를 변경하는 등 시스템을 재구조화 (Restructuring) 할 수 있는 기능을 추가하거나, 대상 시스템을 CICS 코볼에 한정하지 않고 다양한 코볼 시스템으로 확장하거나, 혹은 웹 프로그램을 컴폰넌트로 변환하는 도구와의 통합 등을 고려할 수 있다.

참고문헌

[1] Santiago Comell-Dorda, Kurt Wallnau, Robert C. Seacord, and John Robert, "A Survey of Legacy System Modernization Approaches", Technical Note, CMU/SEI-2000-TN-003, Apr. 2000

[2] "Creating Components from Legacy Applications", CBDi Forum Journal, Dec. 1998

[3] J.Q. Ning, "Recovering reusable components from legacy systems by program segmentation", Proc. Reverse Engineering, pp.64-pp.72, 1993

[4] H.Huang, "Business rule extraction from legacy code", Proc. 20Th Computer software and Applications Conference, pp.922-pp.926, 1996

[5] H.M.Sneed, "Extracting business rules from source code", Proc.4th Program Comprehension, pp.240-pp.247, 1996