2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

래거시 Cobol 시스템의 현대화 지원 도구 구현

김철홍 김동관 양영종
한국전자통신연구원
{kch, dgkim, yangyj}@etri.re.kr

A Tool for Modernization of Legacy Cobol System

Cheol-Hong Kim Dong-Kwan Kim Young-Jong Yang
ETRI Computer & Software Technology Lab SE Department
재작성

요 약

래거시 시스템은 기업의 운영에 중요한 시스템이나, 대부분 Cobol이나 Fortran으로 작성되어 있다. 이들 프로그램들은 수년 동안 많은 프로그래머들에 의해서 유지관리 되어서, 소프트웨어에 많은 수정이 가해졌으며, 대부분 설계 문서들이 존재하지 않는다. 이러한 이유들 때문에 래거시 시스템을 유지보수하는데 많은 비용이 소요된다. 따라서 기업의 시스템 운영에 지장을 주지 않고 래거시 프로그램들을 유지보수하기 용이하도록 하는 것이 시급하다. Cobol로 구축된 시스템을 사용하고 있는 업체들이 고려하고 있는 하나의 접근 방법은 래거시 시스템을 객체지향 시스템으로 재공학(Re-engineering)하거나 새로운 분산환경 플랫폼인 EJB 등으로 통합하는 것이다.

본 논문에서는 래거시 Cobol 시스템을 재공학하기 위한 시각화 도구와 래거시 시스템과 EJB 플랫폼을 연계하기 위한 도구를 제시한다.

1. 서론

1980년대 후반 래거시(Legacy)와 관련된 전략은 프로그램의 이해와 기존 코드로부터 핵심 기능을 추출하는데 중점을 두었다. 래거시 전환(Migration)의 목적은 기존 시스템보다 더 견고하고 유지관리가 쉬운 새로운 시스템을 개발하는 것이다. 프로그래머들은 래거시 시스템을 2계층 클라이언트/서버 아키텍처로 변환하는데 많은 노력을 기울였다. 그러나 2계층 아키텍처는 폭주하는 네트워크 체증을 감당하기 어려웠다. 1990년대 초는 3계층 아키텍처가 2계층 아키텍처로 대치되어 클라이언트의 부담을 줄이고 유연성을 제공하였다. 비록 3계층 아키텍처가 네트워크 체증을 줄이는데 큰 성공을 거두었으나, 래거시 코드를 분석하고 재구축 하는데 많은 비용이 들어간다. 1990년대 중반은 객체지향 언어와 컴포넌트 프레임워크 기술이 성숙되어, 래거시 이전에서 래거시 통합으로 중심이 이동하였다. 객체지향 언어는 래거시 코드의 내부를 숨기고 잘 정의된 인터페이스를 통해 외부 시스템과 연계할 수 있는 객체 래퍼(Wrapper)를 제공한다.

최근 Y2K는 우리로 하여금 낡고 문서화되어 있지 않은 래거시 시스템을 재구조화하고 향상시켰다. 인터넷은 기업이 많은 비용과 시간을 투자하여 소프트웨어를 하도록 하였고, 일부 기업들은 기존 시스템의 사용자 인터페이스로 브라우저를 사용하거나 래거시 시스템과 연계하기 위한 새로운 하이브리드 어플리케이션을 개발하였다. 대부분의 포춘 1,000개 기업은 다른 언어뿐만 아니라 Cobol 시스템에도 계속적인 투자를 하고 있으며, 래거시 시스템을 e-비즈니스로 전환해야 할 필요성을 절감하고 있다.

본 논문에서는 이러한 배경 하에서 대부분의 래거시 시스템을 차지하고 있는 Cobol 프로그램을 새로운 목표 환경으로 전환하기 위한 래거시 Cobol 시스템의 현대화 지원 도구를 제시한다. 2장에서는 관련 연구를 설명하고, 3장에서는 전체 시스템 구조와 각 도구에 대하여 설명한다. 4장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련연구

2.1 코드 기반 래거시 기술

코드 기반 래거시 기술은 GUI 재구성과 객체 래퍼 기술로 분류된다. GUI 재구성은 래거시 전환에서 첫 번째 접근 방법이다. 대부분의 래거시 메인프레임 어플리케이션은 "green screen"인 텍스트 기반 3270 터미널을 사용하는데, 스크린 스크레퍼(Scraper)는 GUI 재구성

프로그램으로 메인프레임 출력 패킷에서 문자 데이터를 읽어 GUI 스타일이나 브라우저에서 출력될 수 있도록 변환한다[1].

2.2 래퍼(Wrapper)

래퍼는 래거시 시스템을 광범위한 어플리케이션 프레임워크와 통합시키기 위한 효과적인 기술이다. 객체 래퍼는 래거시 코드의 블랙 박스 변환 기법으로 기존 프로그램에 대한 인터페이스를 제공한다. 래핑 레이어는 소켓, RPC 또는 API를 통해 래거시 시스템과 통신한다. 래퍼는 래거시 시스템을 변경하지 않고 객체 기반 인터페이스를 제공하고, 모든 인터페이스 스크린, API, 커뮤니케이션 어뎁터, 파일 그리고 데이터베이스를 숨긴다. 래핑된 시스템은 재사용 가능한 컴포넌트가 되어, 새로운 기술이 적용된 어플리케이션에 의해 작동될 수 있으므로 래거시 어플리케이션과 데이터의 수명이 연장된다[1].

2.3 래거시 Cobol 프로그램 재공학

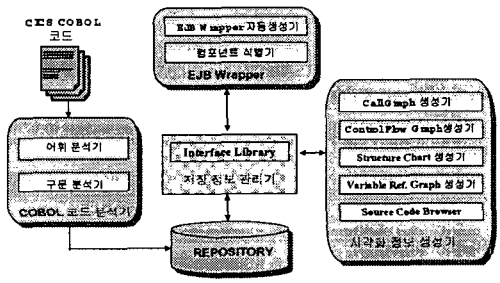
Cobol로 구축된 시스템을 사용하고 있는 업체들이 고려하고 있는 하나의 접근 방법은 래거시 시스템을 객체지향 시스템으로 재공학(Re-engineering)하는 것이다. 그러나 낡은 기술로 개발된 래거시 시스템을 신기술을 적용한 시스템에 통합하는 것은 많은 어려움이 있다. Cobol 프로그램과 객체지향 프로그램을 비교해 보면, 두 개의 프로그램간에는 넓은 간격이 있다. Cobol에서는 지역 변수의 사용이 거의 없기 때문에 캡슐화 수준이 전체 프로그램이지만, 객체지향 언어에서는 변수들은 메소드 또는 클래스 수준에서 캡슐화 할 수 있다. 또한 Cobol에서는 호출되는 프로시져에 매개변수를 전달할 수 없어 프로그램의 일반화(Generality)에 제약을 받으므로 프로그램의 전체가 하나의 모듈 단위가 되며, 객체지향 프로그램에서는 메소드와 프로시져가 완전히 일반화가 되므로 클래스가 모듈 단위가 된다. 이러한 두 언어간의 간격을 좁히기 위해서 래거시 Fortran 프로그램에서 인스턴스 변수와 메소드를 식별하기 위해서 사용된 접근 방법을 제시한다[2].

첫 번째 단계는 래거시 Cobol 프로그램으로부터 인스턴스 변수와 메소드를 자동으로 식별하는 것으로 정적 프로그램 분석인 데이터 흐름 분석, 제어 흐름 분석[3]과 변수 분류[4]를 사용한다. 본 관련 연구에서는 Cobol 프로그램에서 인스턴스 변수와 메소드를 찾기 위해서 데이터 중심 의존(Data-centered Dependence) 모델[5]로부터 시작한다. 이 모델은 각 변수에 대한 *define/use* 정보와 파라그라프에 대한 *call/called-by* 정보를 표현한다. 각 파라그라프를 순회하여 메소드가 될 것인지, 어떤 클래스의 메소드로 포함할 것인지를 결정한다. 이 단계의 산출물은 변수와 메소드를 가지고 있는 클래스들로 구성된 완벽한 객체지향 설계이다. 다음 단계는 객체지향 설계를 개선하고 옵티마이즈하는 것으로 4단계로 수행된다. (1) 어휘상으로 관련이 있는 메소드를 합병하고, (2) 클래스 상호작용 다이아그램으로부터 사이클을 제거하고, (3) 좀 더 작은 클래스들은 합병하고, (4) 큰 메소드들은 더 작게 분할한다. 마지막 단계는 Cobol 코드를 객체지향 프로그램으로 변환한다.

3.래거시 Cobol 시스템의 현대화 지원 도구

3.1 시스템 개요

본 도구는 그림 1과 같이 크게 4개의 서브 시스템으로 구성되어 있다.



(그림 1) 래거시 Cobol 시스템 현대화 지원도구 구성도

Cobol 코드 분석기는 Cobol 소스 프로그램과 화면 정보를 가지고 있는 BMS Map 파일을 파싱하여 정보를 추출한다. 코드 분석기는 어휘 분석을 위해 JFlex를 이용하였고, CICS Cobol의 경우 각 Division에 사용되는 예약어가 각기 다른 용도로 사용되므로 구문 분석은 각 Division을 나누어 파서를 구현하였다. 저장정보 관리기는 EJB Wrapper와 시각화 정보 생성기에서 필요한 정보를 정보저장소로부터 추출하여 라이브러리 형태로 제공한다. EJB Wrapper는 Cobol 프로그램에서 재사용 가능한 기능을 컴포넌트로 식별하고 이를 EJB 환경에서 사용 가능하도록 연계시켜 주기 위한 Wrapper를 자동 생성한다. 시각화 정보 생성기는 재공학 측면에서 Cobol 프로그램을 정적 분석하여 분석된 정보를 시각적으로 보여 준다. 본 장에서는 EJB Wrapper와 시각화 정보 생성기에 대하여 기술한다.

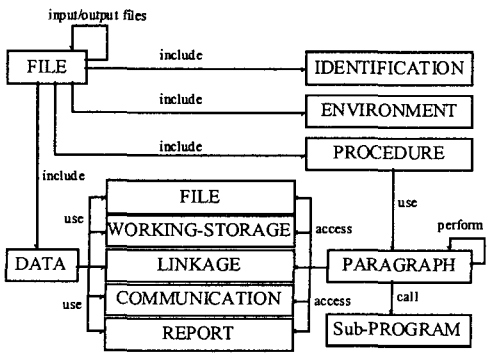
3.2 EJB Wrapper

EJB Wrapper는 컴포넌트 식별기와 EJB Wrapper 자동 생성기로 구성된다. 컴포넌트 식별기는 CICS Cobol 프로그램에서 재사용 가능한 비즈니스 로직이 포함되어 있는 컴포넌트 후보를 식별한다. CISC Cobol 시스템은 시스템의 자원을 효율적으로 이용하기 위해 Pseudo-Conversion 방식으로 프로그램을 하게되는데, 이 방식은 키보드 입력에 따라 처리할 루틴의 시작 파라그라프명을 등록하게 된다. 따라서 재사용 가능한 비즈니스 로직은 이러한 키보드 이벤트와 밀접한 관계를 가지고 있는데, 이벤트와 데이터 흐름을 중심으로 래거시 컴포넌트 후보를 식별한다.

EJB Wrapper 자동 생성기는 EJB 프레임워크와 중간 프레임워크인 래핑 프레임워크을 자동 생성한다. 자동 생성된 래거시 컴포넌트는 유상태(stateful) 세션 빈으로 생성되며 EJB 트랜젝션 서비스를 IBM CICS 시스템과 동일하게 유지하기 위해 트랜젝션 단위로 래핑한다.

3.3 시각화 정보 생성기

일반적으로 Cobol 프로그램의 구성요소는 Files, Division, Section, Paragraph(Procedure), Sentence, Statement, Data-name들로 이루어지며, 프로그램은 이들 7가지 구성요소들간의 관계로 이루어진다. 시각화 정보생성기에서는 이러한 정보들을 이용하여 소스코드보다 더 높은 상위 단계의 다양한 그래픽 표현인 Call Graph(CG), Control Flow Graph(CFG), Structure Chart(SC) 등을 제공하기 위해서 정보의 종류 및 관계, 뷰 형태, 향해 행위, 보고서 종류 등의 분석들이 요구된다.

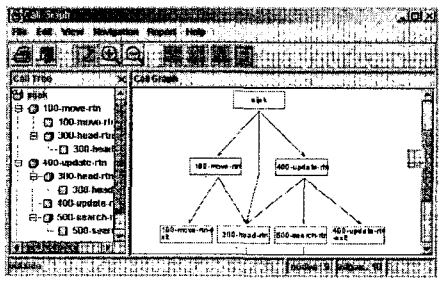


(그림 2) Cobol 프로그램의 개념적 모델

시각화 정보생성기를 위해서 가장 중요한 것은 Cobol 프로그램의 구성요소들 중에서 추상화할 수 있는 정보의 집합을 그림 2와 같이 정의하는 Cobol 프로그램의 개념적 모델이 필요하다. Cobol 프로그램에 대한 개념적 모델은 다수의 추상화 단계에서 사용되는 개체들과 관계성을 정의한 것이다. 그것은 정보의 추상화를 위한 요구 명세서로서 제공된다. 다음은 각 시각화 정보 생성기에 대하여 설명한다.

3.3.1 Call Graph 생성기

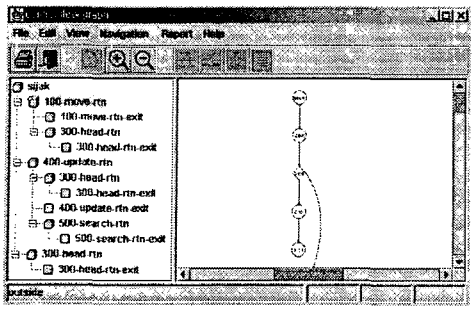
Call Graph(CG) 생성기는 Cobol 프로그램의 Procedure Division에서 정적인 파라그라프 상호간의 호출 흐름 관계를 그림 3과 같이 모형화 하는 도구이다. CG 생성기의 입력 그래프인 G=(V,E)는 파라그라프들간의 호출 관계를 나타내는 두 집합의 쌍으로 이루어졌다. CG 정보 생성기의 그래프 자동 생성 레이아웃 알고리즘에 의해 정점들의 집합인 G=(V)는 두 종류의 파라그라프를 Box 형태로, 그리고 간선들의 집합인 G=(E)는 두 종류의 호출선인 각 Edge로 자동 표현된다.



(그림 3) Call Graph

3.3.2 Control Flow Graph 생성기

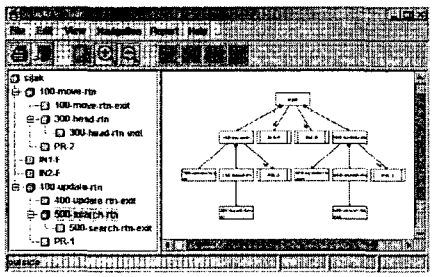
Control Flow Graph(CFG) 생성기는 Cobol 프로그램에서 정적인 파라그라프 내부의 문장 제어 흐름 관계를 그림 4와 같이 모형화 하는 도구이다. CFG 생성기의 입력 그래프인 G=(V,E)는 한 파라그라프 내의 문장 제어 관계를 나타내는 두 집합의 쌍으로 이루어져 있다. CFG 정보 생성기의 그래프 자동 생성 알고리즘에 의해 정점들의 집합인 G=(V)는 제어 구조를 제외한 기본적인 원호의 형태와 제어구조의 마름모 형태로, 그리고 간선들의 집합인 G=(E)는 두 종류의 호출선인 각 edge로 자동 표현된다.



(그림 4) Control Flow Graph

3.3.3 Structure Chart 생성기

Structure Chart(SC) 생성기는 Cobol 프로그램에서 모듈 상호간 호출 관계를 그림 5와 같이 모형화 하는 도구이다. SC 생성기의 입력 그래프인 G=(V,E)는 모듈간의 호출 관계를 나타내는 두 집합의 쌍으로 이루어져 있다. SC 정보생성기의 그래프 자동 생성 레이아웃 알고리즘에 의해 정점들의 집합인 G=(V)는 두 종류의 모듈인 Box 형태로 그리고 간선들의 집합인 G=(E)는 두 종류의 호출선인 각 edge로 자동 표현된다. 그리고 각 vertex에 전달되고 반환되는 데이터 흐름 표시인 데이터 커플은 데이터 이름과 상/하 단 방향성 화살 꼬리 형태로 표현된다.



(그림 5) Structure Chart

3.3.4 Variable Reference Graph 생성기

Variable Reference Graph(VRG) 생성기는 Cobol 프로그램에서 파라그라프들이 어떻게 전역 변수들을 정의/사용하는지를 모형화 하는 도구이다. VRG 생성기의 입력 그래프인 G=(V,E)는 전역 변수와 파라그라프의 정의/사용 관계 및 파라그라프들간의 호출관계를 나타내는 두 집합의 쌍으로 이루어져있다. VRG 생성기의 그래프 자동 생성 알고리즘에 의해 정점들의 집합인 G=(V)는 두 종류의 전역 변수와 세 종류의 함수인 Box 형태로 그리고 간선들의 집합인 G=(E)는 한 종류의 호출선인 각 edge로 자동 표현된다.

4. 결론 및 향후 연구

본 논문은 래거시 Cobol 시스템을 분석하여 프로그램의 정적분석에 필요한 정보들을 시각적으로 보여 주는 시각화 정보 생성기와 CICS Cobol 프로그램을 EJB 플랫폼 환경과 연계하기 위한 EJB Wrapper를 제시하였다. 시각화 정보 생성기는 주로 정적 분석에 초점이 맞추어져 있으며, 향후 객체지향 환경으로 체계적으로 전환하기 위해 객체지향 설계를 생성하는 기법이 더 연구가 되어야 할 것이며, EJB Wrapper의 경우 컴포넌트 식별 방법은 완전한 식별 방법을 제시하지 못하므로 다양한 식별 방법에 대해 앞으로 보다 많은 연구가 필요하다.

5. 참고문헌

[1] Frank P. Coyle, Legacy Integration Changing Perspectives, IEEE Software, pp37-41, Mar/Apr 2000.

[2] Ong C.L. and Tsai W.T.,Class and Object extraction from Imperative Code, J.Object-oriented Prog., pp58-68,Mar/Apr 1993.

[3] Hecht M.S., Flow Analysis of Computer Program,North Holland,New York,NY, 1977

[4] Chen X.P., Tsai W.T., Joiner J.K., Grandamaneni H. and Sun J., Automatic Variable Classification for Cobol Program, In Proceedings of IEEE COMPSAC,1994.

[5] Joiner J.K., Tsai W.T., Chen X.P.,Subramanian S.,Boddu C. and Sun J.,Data-centered Program Understanding,In Proceedings of International Conference on Software Maintenance IEEE,pp272-282,Sept 1994.