레거시 시스템 이해도구를 이용한 비즈니스 로직 추출 기법

송문섭[0] 박창순
한국전자통신연구원
{sirius, cpark}@etri.re.kr

The Extraction Method of Business Logic Using Legacy System Understanding Tool

Moon-Sub Song[0] Chang-Soon Park
Electronics and Telecommunications Research Institute

요 약

최근 컴포넌트 기반 개발이 소프트웨어 개발의 새로운 패러다임으로 대두되고 있는 가장 큰 이유는 컴포넌트의 재사용으로 인해 얻을 수 있는 개발 기간의 단축, 유지·보수의 용이함 등의 장점들 때문이다. 재사용의 개념을 시스템으로 확장하여 생각하면 기존 레거시 시스템을 컴포넌트 기반 시스템으로 바꾸는 것이며 이를 위해서는 레거시 시스템의 비즈니스 로직을 추출하여 컴포넌트화 해야 한다. 본 논문에서는 기존 레거시 시스템에서 컴포넌트 기반 시스템으로 변환하기 위해 필요한 비즈니스 로직 추출 방법으로 레거시 시스템 분석을 통해 얻은 시스템 플로우 그래프, 프로그램 호출 그래프, 패러그래프 흐름 그래프 등의 그래프 등을 이용하여 레거시 시스템을 이해하고 추출 정보로써 핵심 변수와 패턴 식별을 이용한 방법을 제안한 후 구현된 비즈니스 로직 추출기를 이용하여 구체적으로 COBOL 소스 코드에서 비즈니스 로직이 추출되는 과정을 설명하겠다.

1. 서 론

최근 소프트웨어 개발 방법으로 재사용과 효율성에 높일 수 있는 컴포넌트 기반 개발 방법에 많은 관심을 갖고 있으며 많은 기업들이 이런 컴포넌트 기반 시스템을 구축하고자 한다. 그러나 기존 시스템의 중요성으로 인하여 쉽게 컴포넌트 기반 시스템으로 변환하지 못 하는 경우가 있다. 예를 들면 은행의 뱅킹 시스템의 경우 메인프레임과 같은 안정성이 중요하기 때문에 새로운 시스템으로의 전환이 쉽지 않다. 이러한 레거시(Legacy) 시스템의 현대화(Modernization) 문제는 레거시 시스템에 따라 각기 다른 방법을 통해 해결해야 하며, 이런 방법으로는 재개발(Redevelopment), 변환(Transformation), 래핑(Wrapping) 등으로 크게 나눠 볼 수 있다.[1][2][3] 본 논문에서는 이러한 방법들에서 공통적으로 필요한 작업인 주요 모듈인 비즈니스 로직(Business Logic)을 추출하여 컴포넌트화(Componentization)하기 위한 방법을 제안하고, 이를 CICS COBOL 프로그램에 적용하여 EJB Bean 생성에 필요한 비즈니스 로직 추출기에 대해 설명하겠다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 레거시 시스템 현대화 방법에 대해 설명하고 3장에서는 비즈니스 로직 추출 전 단계로 레거시 시스템 이해 단계를 설명한다. 4장에서는 컴포넌트 추출을 위한 비즈니스 로직 정보에 대해 설명하고 5장에서는 비즈니스 로직 추출 방법을 CICS COBOL을 대상으로 설명한 후 6장에서 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

레거시 시스템의 현대화 방법으로는 크게 재개발, 변환, 래핑 등으로 나눠 볼 수 있다. 재개발 방법은 기존 시스템에 대한 이용이 가장 적으며 개발 비용과 기간이 길다. 또한 개발 후 기존 시스템을 대체할 경우 신뢰성과 안전성에 대한 보장이 약하다.[2] 이에 반해 래핑 방법은 기존 시스템은 그대로 두고 새로운 시스템 환경에서 사용 가능하도록 중간에 래퍼(Wrapper)를 둠으로써 신뢰성과 안정성이 좋지만, 시스템의 유연성과 확장성 측면에서 많은 단점이 있다.[2][3] 변환 방법은 레거시 시스템을 새로운 시스템으로 1:1로 매핑(Mapping)해 주는 방법으로 가장 이상적이지만 시스템들간의 환경, 소프트웨어 아키텍처(Architecture), 프로그램의 특성이 다르기 때문에 대부분 구현하기가 힘들다. 프로그램의 구조를 비즈니스 로직과 데이터베이스로 나눠 볼 때 현재 레거시 시스템 현대화는 주로 데이터베이스의 변환 작업에서 이루어지고 있는데 이는 여러 가지 이유가 있겠지만 기존 프로그램에서 비즈니스 로직을 추출하는 것이 어렵기 때문이다.[3][4] 특히 레거시 시스템에 대한 문서(Document)의 부재로 레거시 시스템에 대한 사용자의 이해를 위해 분석하는 작업 또한 난해한 문제이다.[6] 이러한 문제들로 인하여 시스템이 작은 경우에는 재개발을 하거나 큰 시스템의 경우 인터페이스 래핑 등의 방법을 사용하는데 이는 자원의 효율성과 확장성에 문제를 발생시킨다.[1][2][3]

3. 레거시 시스템 이해 단계

레거시 프로그램에서 비즈니스 로직을 추출하기 위해서는 먼저 레거시 시스템의 구조와 기능을 이해하고 있어야 한다. 대부분의 레거시 프로그램들이 문서화되어 있지 않음으로 인하여 시스템에 대한 이해가 부족하여 비즈니스 로직 추출에 어려움이 있을 수 있기 때문이다. 레거시 시스템 이해 단계에서 제공하는 그래프는 다음과 같다.

3.1 시스템 플로우 그래프

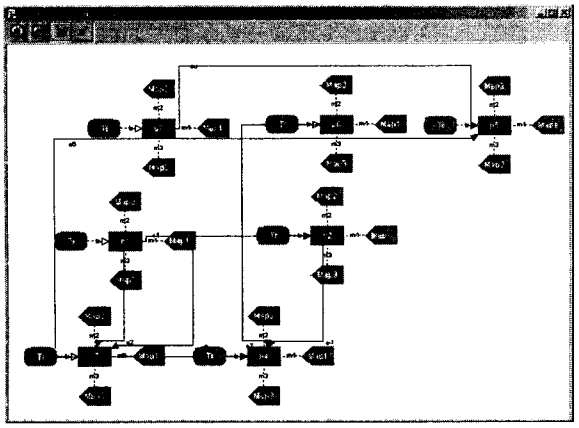

그림 1 시스템 플로우 그래프

시스템 플로우 그래프(System Flow Graph)는 레거시 시스템의 전체 구조에 대한 정보를 나타낸다. CICS COBOL 프로그램에서는 프로그램 명, 맵 파일 명, 트랜잭션 명 등의 정보가 표현된다. 특히 프로그램간의 호출 정보를 통해 시스템의 전체 흐름을 파악할 수 있다. 그림 1은 CICS COBOL 프로그램의 구조를 보여주는 시스템 플로우 그래프의 예이다.

3.2 프로그램 호출 그래프

시스템 플로우 그래프가 시스템 전체의 구조를 보여주는 반면 프로그램 호출 그래프는 하나의 프로그램 내의 패러그래프(Paragraph)들 간의 호출 정보를 보여주는 그래프이다. CICS COBOL의 경우 패러그래프간의 호출은 PERFORM과 GOTO 명령어를 사용하여 다른 패러그래프를 호출한다. 다음 그림 2는 프로그램 호출 그래프의 예이다.

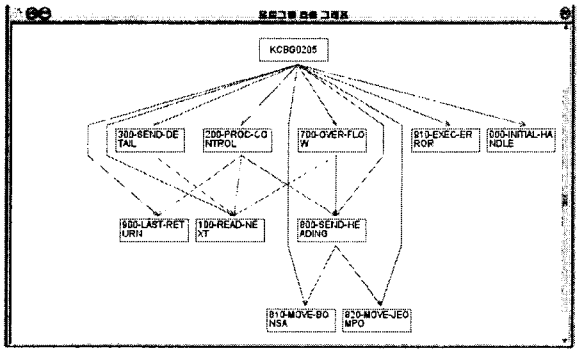

그림 2 프로그램 호출 그래프

3.3 패러그래프 흐름 그래프

패러그래프 흐름 그래프는 패러그래프 내의 문장들간의 흐름 정보를 보여주는 그래프이다. 각 노드들은 소스 코드의 한 라인을 나타내면 화살표는 제어 흐름을 나타낸다. 그림 3은 패러그래프 흐름 그래프의 예이다.

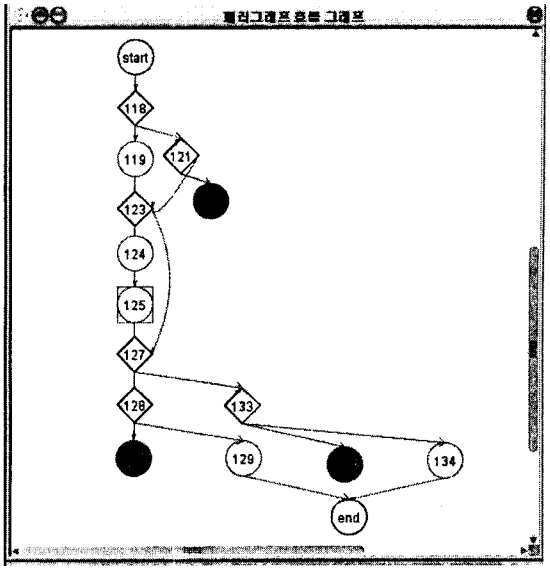

그림 3 패러그래프 흐름 그래프

4. 비즈니스 로직 정보 추출 단계

비즈니스 로직 정보 추출 단계는 비즈니스 로직을 추출하기 위해 필요한 데이터 정보를 분석해 내는 단계이다. 비즈니스 로직 추출에 필요한 데이터는 비즈니스 로직 계산 관련 코드들이 주를 이룬다. 본 논문에서는 비즈니스 로직 정보들을 소스 코드들로부터 식별해 내기 위해서 핵심 변수 식별 과정과 프로그램 패턴 식별 과정을 수행하게 된다. 본 논문의 핵심 변수는 비즈니스 로직 추출을 위해 필요한 변수를 말하며 프로그램 패턴은 조건문과 호출문 등으로 이루어진 일정한 문장 구조를 뜻한다.

4.1 핵심 변수 식별

핵심 변수 식별 과정은 비즈니스 로직 추출의 단서가 되는 핵심 변수(Core Variable)를 프로그램 내에서 찾아내는 과정이다. 핵심 변수의 후보(Candidate)로는 입력 변수, 출력 변수, 주요 변수들이 있다. 주요 변수는 프로그램 내에서 사용된 변수 중에 제어에 관련된 변수, 또는 패턴에 자주 사용된 변수들을 말한다. 비즈니스 로직이 데이터를 계산하여 처리하는 일련의 흐름이므로 핵심 변수를 식별하는 것은 비즈니스 로직을 추출하기 위해 필요한 작업이다. 이렇게 식별된 핵심 변수들은 비즈니스 로직 추출 단계에서 수행하는 Slicing 작업의 시작점이 된다.

4.2 프로그램 패턴 식별

핵심 변수 식별과 더불어 본 논문에서는 CICS COBOL 프로그램의 PROCEDURE DIVISION에서 반복적으로 사용하는 논리 구조를 찾아내 비즈니스 로직 추출에 활용한다. 즉 반복적으로 코딩된 논리 구조를 하나의 패턴으로 인식하게 된다. 이렇게 인식된 패턴을 이용하여 공통 비즈니스 로직 후보를 추출할 수 있게 된다. 코볼 프로그램에서 나타나는 패턴으로는 호출 패턴과 IF 패턴이 있다. 호출 패턴의 피호출 패러그래프와 IF 패턴의 제어 변수는 비즈니스 로직 추출 단계에서 중요 정보로 이용된다.

5. 비즈니스 로직 추출 단계

비즈니스 로직 추출 단계는 추출된 비즈니스 로직 정보를 이용하여 본격적으로 하나의 비즈니스 로직을 구성하는 부분 코드들을 식별한 후 컴포넌트화 할 수 있도록 추출하는 단계이다. 비즈니스 로직 추출의 첫 번째 단계는 CICS COBOL의 DATA DIVISION에서 핵심 변수 후보를 추출하는 과정이다. 즉 DATA DIVISION에 정의된(COPY 파일 포함) 변수들을 입력 변수, 출력 변수, 임시 변수(Temporary Variable), 주요 변수 등으로 분류한다.[7] 다음 단계에서는 프로그램에서 사용된 패턴을 찾기 위해 PROCEDURE DIVISION을 분석한다. PROCEDURE DIVISION에서 사용된 패러그래프들의 문장들을 분석하여 패러그래프들 간의 구조 정보를 획득한다. 다음 단계에서는 획득된 구조 정보를 이용하여 패턴들을 식별한다. 다음 단계에서는 식별된 패턴 중에 호출 패턴을 이용하여 피호출 패러그래프를 중심으로 핵심 패러그래프를 식별한다. 다음 단계에서는 식별된 핵심 패러그래프와 IF 패턴의 제어 변수 등을 이용하여 핵심 변수를 식별한다. 프로그램 내에서 반복적으로 사용되는 패턴은 공통 비즈니스 로직(Common Business Logic)의 후보가 된다. 또한 여러 프로그램에서 같은 프로그램 혹은 패러그래프를 호출 할 경우에도 호출되는 모듈은 비즈니스 로직의 후보가 된다. CICS COBOL 프로그램의 문장 구조는 대부분 분기 패턴이며, 분기 패턴에 사용된 제어 변수(Condition Value)는 핵심 변수와 더불어 다음 단계인 Slicing 작업의 시드 값(Seed Value)으로 사용된다. Slicing 단계에서는 시드 값에 따라 Forward Slicing과 Backward Slicing을 수행한다.[8] 여기서 Slicing은 패러그래프 단위로 이루어진다. 이상과 같은 과정들을 순서도로 표현하면 그림 4와 같다.

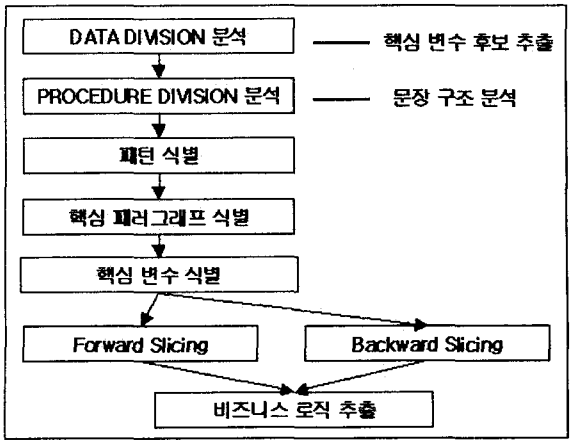

그림 4 비즈니스 로직 추출 순서도

6. 결론 및 향후 연구 과제

본 논문에서는 레거시 시스템 중에서 CICS COBOL을 대상으로 컴포넌트 기반 시스템으로 변환 시에 필요한 비즈니스 로직 추출 기법을 제안하였다. 레거시 시스템에 대한 이해 도구의 기능을 하는 시스템 플로우 그래프, 프로그램 호출 그래프, 패러그래프 흐름 그래프는 레거시 시스템에 대한 문서가 없을 경우에도 프로그램의 흐름을 쉽게 이해할 수 있도록 도와줌으로써 비즈니스 로직 추출뿐만 아니라 시스템 유지 보수 측면에서도 활용할 수 있다. 비즈니스 로직 추출 방법으로는 프로시저에서 처리하는 데이터를 중심으로 핵심 변수를 추출하고 주요 프로시저의 패턴 구조를 식별해 낸 후 이를 이용하여 비즈니스 로직 후보가 되는 패러그래프와 변수를 결정한 후 Slicing을 이용하였다. 이러한 추출 방법은 사용자가 레거시 시스템에 대한 지식을 갖고 있어야한다는 제약 조건은 있으나 프로그램 내의 주요 데이터와 논리 구조를 이용함으로써 컴포넌트로 변환 될 수 있는 비즈니스 로직을 추출할 수 있다. 향후 연구 과제로는 추출된 비즈니스 로직에 관한 검증 및 테스트에 관한 연구와 CICS COBOL 프로그램 이외의 언어에 대해서도 적용 가능한 추출 기법에 대한 연구가 이루어져야 한다.

참고문헌

[1] H. M. Sneed, Extracting business logic from existing COBOL programs as a basis for redevelopment, *Proc. 9th, Program Comprehension(IWPC 2001),* 2001, 167-175.

[2] H. M. Sneed, A Case Study in Software Wrapping, *Proc. IEEE Software Maintenance,* pp86-92, *1998.*

[3] M. S. Lee, The design and implementation of Enterprise JavaBean(EJB) Wrapper for Legacy system, *Systems, Man and Cybernetics, 2001 IEEE International Conf.* 2001, pp.1988-1992 vol.3

[4] H. Huang, Business rule extraction from legacy code, *Proc. 20th, Computer Software and Applications Conf.,* 1996, 922-926.

[5] A. Perkins, Business rules=meta-data, *Proc. 34th, Technology of Object-Oriented Languages(TOOLS 34'),* 2000, 285-294.

[6] J. K. Joiner, Data-Centered Program Understanding, *Proc. Software Maintenance,* 1994, 272-281.

[7] K. Kawabe, Variable Classification Technique for Software Maintenance and Application to The Year 2000 Problem. *Proc. 6th Software Engineering Conf.(APSEC'99),* 1999, 500-506.

[8] Weiser, M.:"Program Slicing", *IEEE Trans. on S.E.,* Vol. 10, No. 4, July, 1984, pp.72.