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

자바 코드 메트릭 측정 도구의 설계 및 구현

오현철°, 김태균, 윤석진*
부산외국어대학교 컴퓨터공학과
*한국전자통신연구원
e-mail : {hcoh°, ktg}@saejong.pufs.ac.kr, sjyoon@etri.re.kr

Design & Implementation of Metrics Evaluation Tool for Java Code

Hyun-Chul Oh°, Tae-Gyun Kim, Seok-Jin Yoon*
°Dept. of Computer Engineering, Pusan Univ. of Foreign Studies
*ETRI

요 약

본 논문은 기존에 작성된 자바 프로그램이나 현재 구현 중인 자바 프로그램을 대상으로 메트릭 정보를 측정하는 자동화 도구의 설계 및 구현 결과를 논한다. 이러한 도구의 구현을 위해 필요한 가장 핵심적인 기능은 자바 코드에 대한 분석 기능이다. 본 논문의 내용은 한국전자통신연구원의 컴포넌트 공학 연구팀 주관으로 EJB(Enterprise Java Beans) 기술을 기반으로 컴포넌트를 개발하기 위한 환경인 COBALT(Component Based AppLication devlopment Tool) 시스템의 부 시스템으로 구현된 자바 코드 메트릭 측정 도구의 설계 및 구현 결과를 다룬다. 본 논문에서 구현된 자바 코드 메트릭 측정 도구을 통하여 클래스의 적절한 분할, 클래스 멤버 자원의 적절한 배치, 상속 트리의 적절한 조직 등을 이룰 수 있다.

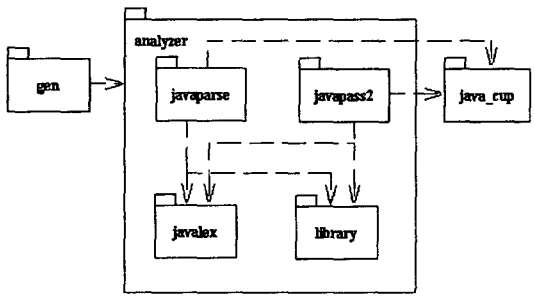
1. 서론

객체 지향 기법이 제공하는 자료추상화, 상속, 동적 바인딩, 다형 개념과 같은 기술이 구조적 기법에 비하여 소프트웨어의 개발을 좀더 체계적으로 이룰 수 있게 하며 소프트웨어의 재사용성을 증대시키고 있음은 잘 알려진 사실이다. 그러나 객체지향 기술만으로는 소프트웨어의 개발을 하드웨어 개발 방식으로 수행하고자 하는 소프트웨어 엔지니어들의 바램이 충족될 수 없다. 기존의 객체 지향 기법이 가지는 가장 큰 한계점은 소프트웨어의 재사용이 원시 코드 단계에서만 이루어질 수 있다는 것이다. 즉 기존 소프트웨어의 재사용을 통해서 새로운 소프트웨어를 개발하려는 개발자는 기존 소프트웨어 원시 코드들을 수집하여 자신의 요구사항에 맞도록 변경한 후, 컴파일 작업과 링크 작업을 수행해야 한다. 기존 소프트웨어의 기능을 재사용하기 위해서 반드시 링크 작업을 수행해야 하는 문제점은 소프트웨어 부품의 계속적인 추가 버전을 최종 개발자가 쉽게 이용할 수 없게 한다. 이러한 문제점을 해결하기 위해 제안된 기술이 소프트웨어 컴포넌트 기술이다. 소프트웨어 컴포넌트 기술은 원시 코드 단위가 아닌 실행 단위의 소프트웨어 부품들을 실행시에 연결할 수 있도록 하는 기술로서 정해진 인터페이스를 충족하는 소프트웨어 부품들을 조립하여 새로운 기능의 소프트웨어를 구축할 수 있도록 한다. 소프트웨어 컴포넌트 관련 기술은 1990 년대 중반 이후로 심도 있는 연구가 진행되고 있으며 Microsoft 사의 ATL/DCOM, Active X, MTS(Microsoft Transaction Server)나 Sun 사의 EJB(Enterprise Java Beans), OMG 의 CORBA 기술들이 컴포넌트와 관련한 현재의 핵심 기술이다. 한국전자통신 연구원의 컴포넌트 공학 연구팀에서는 2000 년 이후로 EJB 를 이용한 소프트웨어 컴포넌트 기술 기반 연구를 진행중이며 개발 환경으로 COBALT 시스템을 구현 중이다. COBALT 시스템은 자바로 구현되었으며 UML 을 이용한 영역 모델링 기능, 컴포넌트 조립 및 배치 기능, 자바 언어 분석 기능 및 컴포넌트 추출 기능 등을 갖고 있다. 본 논문은 COBALT 의 기능 중에서 자바 언어 분석 기능과 메트릭 측정 기능을 개발하는 과정

에서 얻어진 설계 및 구현 결과에 대하여 다룬다. 자바 언어 분석 기능은 JLEX/CUP 을 이용하여 구현되었으며 메트릭 정보 제공 사용자 인터페이스 구현을 위한 JDK(Java Development Kit) 1.3 과 Swing 을 이용하였다. 본 논문의 2 장에서는 시스템의 설계에 대하여 논하며 3 장에서는 구현 결과를 제시하고 4 장에서는 결론과 함께 차후 연구 방향에 대하여 기술한다.

2. 시스템 설계

본 논문에서 구현하고자 하는 도구는 여섯 개의 패키지들로 구성되며 이들 간의 상호 관계는 (그림 1)과 같다.



(그림 1) 자바 역공학 도구의 패키지 구조

(그림 1)의 패키지 구조는 역공학 기능 구현 시의 패키지 구조와 동일하다. 그 이유는 메트릭 정보 측정 기능의 구현을 위해서는 역공학 기능을 확장하는 방식으로 작업이 이루어질 수 있기 때문이다. (그림 1)에 나타나있는 각 패키지의 기능은 다음과 같다.

• gen 패키지 : 이 패키지는 전체 도구의 통합 GUI 를 구성한다. 본 논문에서 구현된 기능들은 gen 패키지에 속하는 통합 GUI 의 메뉴 선택에 의해 관련 함수가 호출됨으로써 수행된다.

• analyzer 패키지 : 이 패키지는 본 논문의 결과로 작성된 프로그램들을 포함하는 주 패키지이로서 다음에 기술된 각각의 기능에 따라 분리되어 설계된 javalex, javaparse, java_cup, javapass2, library 패키지들을 포함한다.

• javalex 패키지 : 이 패키지는 자바 프로그램에 대하여 렉시칼 분석을 실행할 때 필요한 기능을 수행하는 클래스들을 포함하는 패키지이다. 이 패키지에서는 입력 스트림을 이용하여 자바 프로그램으로부터 한 문자씩 입력을 받아 구문 분석을 위한 토큰을 생성하여 line 및 연산자의 정보를 계산한다.

• javaparse 패키지 : 이 패키지는 자바 프로그램에 대하여 구문 분석을 수행한 후 파스 트리를 만들어 주는 기능을 구현하기 위해 필요한 클래스들을 포함하는 패키지이다.

• javapass2 패키지 : 이 패키지는 이중 패스 알고리즘 중에서 두 번째 패스를 담당하며 렉시칼 분석과 구문 분석을 수행한 후 메소드 단위의 메트릭 정보를 추출하기 위해 이용되는 파스 트리를 구축한다.

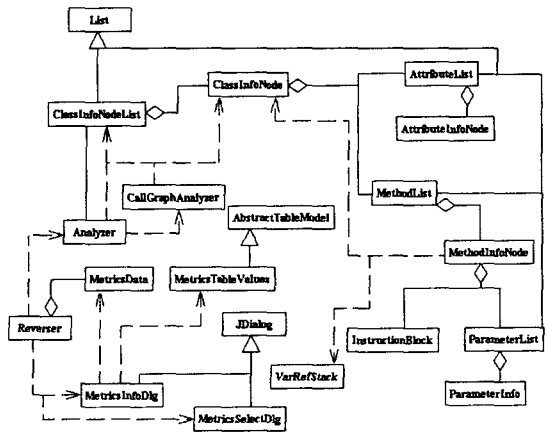
• library 패키지 : 이 패키지는 파싱된 결과를 저장하는 자료구조와 메트릭 측정 기능의 구현을 위해 보편적으로 필요한 클래스들을 포함하는 패키지이다.

• java_cup 패키지 : 이 패키지는 클래스의 내용들이 고정되어있는 패키지로 자바 CUP 을 사용하기 위해 런 타임 시에 링크 되어야 하는 패키지이다.

본 본문에서 구현된 도구는 이중 패스를 통하여 자바 코드를 파싱한다. 원시 코드 파싱 과정을 두 단계로 나눈 이유는 메소드 단위의 결합도와 응집도 관련 데이터를 수집하기 위해서이다. 결합도와 응집도 메트릭 생성을 위한 주된 작업은 자바 멤버 함수의 본체에 포함된 다른 객체에 대한 메시지 호출처리이다. 그런데 멤버 함수의 본체를 처리할 때 필요한 정보는 참조된 클래스들의 메소드 정의이므로 참조된 클래스에 대한 메소드 정보가 미리 존재하지 않으면 이를 구성하기 위한 역공학 작업에 매우 복잡한 제어 구조가 이용되어야 한다. 이러한 문제점을 해결하기 위해 두 단계의 파싱 과정이 이용되고 있으며 두 단계의 패스에서 분담된 역할은 다음과 같다.

• 패스 1 : 이 단계에서는 모든 클래스의 데이터 멤버, 멤버 함수 정의를 처리한다. 아울러 추출된 파스 트리에 상속 관계에 대한 정보를 반영함으로써 패스 2 에서 상속을 고려한 멤버 함수 탐색 작업이 가능하도록 한다.

• 패스 2 : 이 단계에서는 주로 멤버 함수의 본체를 파싱하면서 다른 객체에 대한 메시지 호출 정보를 추출한다. 패스 1 에서 메시지 정의에 대한 정보는 이미 수집되어 있는 상태이기 때문에 패스 2 과정에서 함수 호출을 발견하면 호출된 함수 정의에 대한 레퍼런스와 실인자를 포함하는 함수 호출 문에 대한 정보를 저장한다.



(그림 2) 라이브러리 패키지에 클래스들간의 관계

(그림 2)의 클래스 다이어그램은 패스 1 과 패스 2 과정에서 사용되는 클래스들 간의 상호 관계를 모델링 한것으로 이 그림에 속하는 클래스들의 역할을 개략적으로 설명하면 다음과 같다.

• Reverser : 역공학 도구 서브 시스템의 관점에서 메인 프로그램 역할을 하는 객체이다. 이 객체의 임무는 크게 세 가지인데 그 하나는 통합 GUI 의 메뉴 선택에 대한 해당 기능을 수행하는 인터페이스를 제공하는 것이고, 다른 하나는 자바 코드 분석 이후에 통합 GUI 가 제공하는 인터페이스를 이용하여 다이어그램들을 작성하는 것이며, 마지막 임무는 수집된 메트릭 정보를 보여주는 것이다.

• Analyzer : 첫 번째 패스를 위한 구문 분석을 수행하는 파서 객체이다. 주된 목적은 모든 클래스를 파싱하여 각 클래스를 위한 데이터 멤버와 멤버 함수 정의를 추출하는 것이다. 이 객체의 수행 결과로부터 클래스 단위의 메트릭 정보를 수집할 수 있으며 외부 시스템에 정보를 제공하는 인터페이스를 갖추고 있다.

• CallGraphAnalyzer : 두번째 패스를 위한 구문 분석을 수행하는 파서 객체이다. 이 객체는 첫 번째 패스의 결과를 이용하여 멤버 함수 본체로부터 복잡도, 결합도, 응집도와 관련된 메트릭 정보를 추출할 수 있는 함수 호출 정보를 얻는다.

• ClassInfoNode : 하나의 클래스에 대하여 분석된 모든 정보를 관리한다. 이 객체를 통하여 해당 클래스에 속하는 모든 멤버의 자세한 정보의 자세한 정보를 얻을 수 있을 뿐만 아니라 개별 클래스에 대한 메트릭 측정값을 얻을 수 있다.

• ClassInfoNodeList : 분석된 모든 클래스 객체들을 관리하는 컨테이너 객체이다.

• AttributeInfoNode : 특정 클래스에 속하는 하나의 데이터 멤버에 대한 모든 정보를 관리한다.

• MethodInfoNode : 특정 클래스에 속하는 하나의 멤버 함수에 대한 모든 정보를 관리한다.

• ParameterInfo : 특정 멤버 함수에 속하는 형식 인자에 대한 정보를 관리한다.

• VarInfoNodeList, MethodInfoNodeList, ParameterList : 이들은 다수의 VarInfoNode, MethodInfoNode, ParameterInfo 객체들을 각각 관리하는 컨테이너 객체들이다.

• InstructionBlock : 모든 멤버 함수의 부속 객체로 소속되며 명령어 블록 단위의 함수 정보를 관리한다.

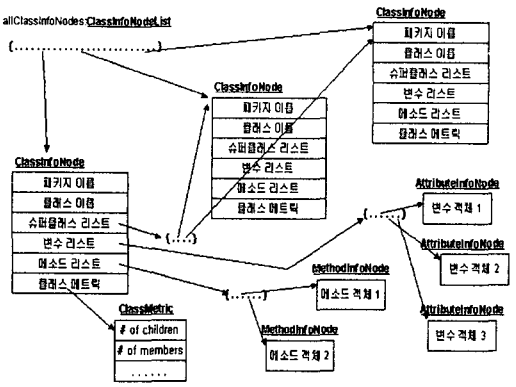
• VarReferenceStack : 함수 호출 관계를 분석할 때 참조되는 객체의 scope 를 관리하기 위한 사용된다.

• MetricsData : 해당 클래스에 대한 메트릭 정보를 저장한다

• MetricsInfoDlg : 측정된 메트릭 정보를 테이블 형태로 사용자에게 보여주는 위해 사용된다.

• MetricsTableValues : MetricsInfoDlg 대화상자 내부에서 사용되는 테이블의 값을 관리하는 것으로 측정된 메트릭의 모든 정보를 저장한다.

• MetricsSelectDlg : 메트릭 종류 주에서 사용자 원하는 것들을 선택하기 위해 사용된다.



(그림 3) 분석된 클래스 정보들을 저장하는 자료구조

(그림 2)에 나타나 있는 클래스들은 역공학 적용 후 만들어진 파스 트리를 정보 사용이 용이하도록 가공하기 위한 목적을 갖는다. (그림 3)의 내용은 시스템 실행 시에 (그림 2)의 객체들이 어떠한 방식으로 정보를 저장하는가를 보여주는 자료구조이다.

3. 시스템 구현

본 도구에서는 47 가지의 메트릭 값을 측정하는 기능이 구현되었다. 이들 모두는 다음과 같이 10 가지 부류로 구분된다.

• 기본 메트릭 : 특정 데이터를 카운트함으로써 얻어지는 값들이다

• 응집력 매트릭 : 클래스 내부에 속하는 멤버들끼리의 관계가 얼마나 밀접한가를 측정하는 값들이다.

• 복잡도 메트릭 : 시스템을 구성하는 클래스들의 복잡도를 구하는 메트릭들이다.

• 결합도 메트릭 : 클래스 사이의 연관 정도를 측정하는 값들이다.

• 할스테드 메트릭 : 할스테드의 소프트웨어 과학 메트릭에서 정의된 것들이다.

• 캡슐화 메트릭 : MOOD 에서 정의된 것들로서 프로젝트 단위로 수집되는 것들이다.

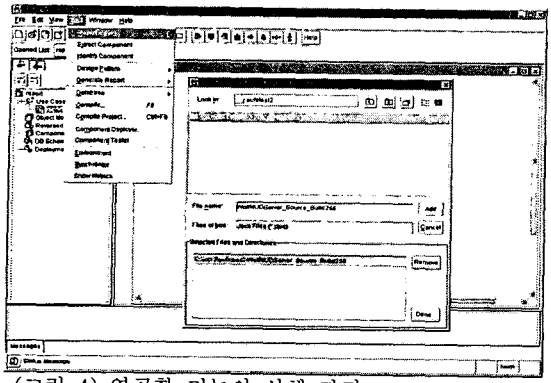
• 상속 메트릭 : 상속과 관계되는 값들이다.

• 최대값 메트릭 : 특정 데이터의 최대치를 구함으로서 얻어지는 것들이다.

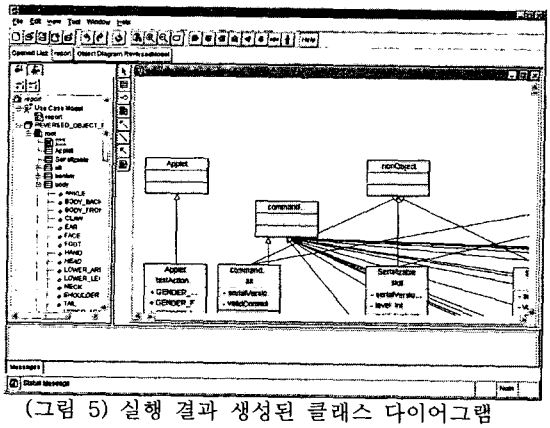
• 다형성 메트릭 : 상속 관계에서 오버라이드된 멤버와 관계된 값을 측정한 것이다.

• 비율 메트릭 : 특정한 데이터들의 비율을 구한 값들이다.

위 메트릭 측정 기능의 구현 결과를 실행 화면 위주로 제시한다. 메트릭 측정 기능의 실행을 위해서는 우선 역공학 기능을 실행해야 한다. (그림 4)의 내용은 역공학 실행을 위한 메뉴와 대화 상자 이용과정을 보여주며 (그림 5)의 내용은 역공학 실행 결과를 보여준다.

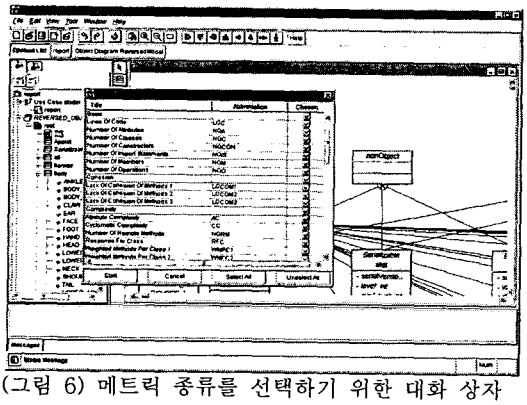


(그림 4) 역공학 기능의 실행 과정



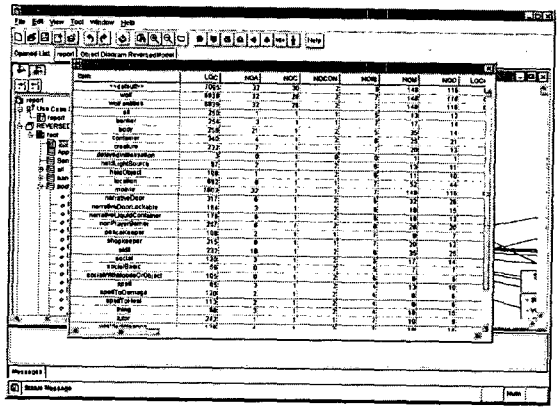
(그림 5) 실행 결과 생성된 클래스 다이어그램

(그림 5)의 화면에 추출된 클래스 다이어그램은 예제로 작성된 자바 클래스들을 대상으로 시험한 것이다. 역공학 기능의 실행 이후에 메트릭 정보 추출 기능의 실행이 가능하며 이는 <Tool>/<Show Metrics> 버튼을 선택함으로써 가능하다. 메트릭 측정 기능을 실행하면 우선 (그림 6)와 같은 형태의 대화 상자가 나타나는데 이 대화 상자를 이용하여 사용자는 그가 원하는 메트릭 종류들에 대해서만 선택적으로 정보를 제공받을 수 있다.



(그림 6) 메트릭 종류를 선택하기 위한 대화 상자

(그림 6)의 대화 상자에서 <Start> 버튼을 선택하면 최종적으로 (그림 7)과 같은 상태가 된다. (그림 7)은 예제 프로그램에 속하는 클래스들로부터 추출된 모든 메트릭을 선택한 경우에 측정된 메트릭 값들이다. (그림 7)에서와 같이 본 도구에서 구현된 47 가지의 메트릭 정보를 한눈에 파악하기는 쉽지 않으며 테이블의 스크롤 바를 이동 시켜야지만 사용자가 원하는 메트릭 값을 확인 할 수 있다. (그림 6)의 메트릭 종류 선택용 대화 상자는 사용자가 관심을 갖고 있는 메트릭 값들만을 추출할 수 있도록 하기 때문에 사용하기가 편리하다.



(그림 7) 메트릭 값들을 보여주는 대화상자

4. 결론

본 논문에서는 자바 코드 메트릭 측정 도구의 설계 및 구현에 대하여 기술하였다. 자바 코드 메트릭 측정 도구는 자바로 구현된 시스템의 품질 보증 작업을 위한 필수적인 도구이다. 본 도구의 구현을 통하여 구현된 기능은 크게 세 가지이며 다음과 같다.

- 자바 프로그램 분석 기능
- 메트릭 데이터 수집 및 계산 기능
- 메트릭 정보 제공 사용자 인터페이스 기능

현재 구현 결과물에 존재하는 문제점은 시스템의 안정성에 관한 것이다. 추후의 계속적인 연구를 통하여 시스템의 신뢰성이 확보될 수 있도록 보완할 계획이다.

참고문헌

[1] P. Allen, A.C. Wills, "*Objects, Components, and Frameworks with UML: The Catalysis Approach*," Addison-Wesley, 1998.

[2] Steven Atkinson, "Modeling Formal Integrated Component Retrieval," Fifth International Conference on Software Reuse, June 1998.

[3] S. R. Chidamber, C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE TOSE, Vol. 20, No. 6, June, 1994.