2000년 한국정보처리학회 춘계 학술발표논문집 제7권 제1호

객체지향 메트릭스에 의한 자바 프로그램 테트팅 도구의 구현

김상영, 황선명
대전대학교 컴퓨터공학과
e-mail:jayusop@zeus.taejon.ac.kr
sunhwang@dragon.taejon.ac.kr

Developing Testing Tools Using Object-Oriented metrics for programs implemented by Java

SangYoung Kim, SunMyung Hwang
Dept of Computer Engineering, Taejon University

요 약

최근 주목을 받고 있는 자바는 객체지향성을 가장 잘 반영한 언어이며 많은 연구 분야에서 사용되고 있지만 기존의 테스팅 도구들이 대부분 C++을 기본 대상 언어로 사용하기 때문에 자바에 적용하기 위한 테스팅 방법과 도구들이 부족한 것이 현실이다. 때문에 기존의 테스팅 도구들에 자바 프로그램을 적용하기는 까다로운 일이 아닐 수 없다.

본 논문에서는 기존에 연구되어진 객체지향 소프트웨어 메트릭스들을 자바에 적용하는 방법과 자바에 적용할 수 있는 새로운 메트릭스들과 자바프로그램의 구조를 보다 쉽게 파악하기 위한 분석지원 도구의 설계 및 구현에 대하여 연구하였다.

1.서 론

어느 제품에 있어서도 품질의 중요성은 아무리 강조해도 지나치지 않는다. 소프트웨어 테스팅은 소프트웨어의 품질을 확보하고 결함을 찾아내기 위해 수행되는 일련의 작업이다. 테스팅은 개발된 소프트웨어의 풀질에 대한 평가와 품질 향상을 위한 수정 작업들을 포함한다[10].

소프트웨어 테스팅에는 개발자는 물론이고 사용자, 독립적인 테스터 등이 참여한다. 소프트웨어에 대한 테스팅은 소프트웨어 품질 보증을 위한 마지막 단계이며, 따라서 품질 보증을 위한 마지막 보루라 할 수 있다. 이 단계에서 수정되지 않는 오류는 사용자가 사용하며 발견되어 이를 수정하는데 이때 많은 비용이 소요된다. 테스팅이 제대로 수행되지 않았을 때 소용되는 비용과 품질 저하, 고객의 불만족 등은 잘 계획되고 철저한 테스팅이 얼마나 중요한 것인가를 설명해주고 있다[13][14].

본 논문에서는 최근 각광을 받고 있는 자바 프로그램을 입력으로하여 프로그램을 정적 분석하고 메트릭스들을 사용하여 프로그램의 클래스와 모듈 단위의 복잡도 등을 측정하고, 그래픽 분석할수 있는 테스팅 지원도구를 개발하였다.

2. 소프트웨어 메트릭스

2.1 소프트웨어 메트릭스의 정의

소프트웨어 메트릭스(Software Metrics)는 컴퓨터 소프트웨어에 대한 넓은 영역의 측정을 말한다. 측정은 소프트웨어를 개선시키려는 의도로 소프트웨어 프로세스에 적용된다. 이러한 소프트웨어 메트릭스들은 소프트웨어 프로젝트 전반적인 부분에 걸쳐서 사용되어진다.

2.2 구조적 소프트웨어 매트릭스

2.2.1 McCebe의 Cyclomatic Complexity

소스 코드의 복잡도를 측정하는 방법에는 여러 가지 방법들이 있는데 가장 보편적인 방법이 McCabe가 제안한 메트릭스이다. 이 방법은 프로그램의 논리 흐름을 표현한 제어 흐름 그래프(Control Flow Graph)를 기초로 한다. McCabe의 복잡도는 Cyclomatic 수를 기본으로 하는데 이는 소스 코드에서의 조건문의 수나 플로우 그래프에서의 노드(node)와 에지(edge)의 수를 통한 계산 그리고 플로우 그래프에서의 영역 체크를 통하여 계산되어진다. McCabe는 이러한 Cyclomatic 수를 기본으로 하여 여

韓國領 독도 Dokdo is Korean Territory

러 가지 메트릭스들을 제안하였다.

2.3 객체지향적 소프트웨어 메트릭스

2.3.1 CK 메트릭스

Chidamber와 Kermerer는 클래스당 가중치를 부여한 메소드(WMC), 상속 트리의 깊이(DIT), 자식의 수(NOC), 객체간의 결합도(CBO), 클래스에 대한 응답(RFC), 메소드의 응집도 결핍(LCOM)을 척도로 제안하였다

2.3.2 Lorenz와 Kidd 제안 메트릭스

Lorenz와 Kidd는 클래스의 크기(CS), 부 클래스에 의해 추가된 연산의 수(NOA), 특수화 지수(SI)를 척도로 제안하였다.

2.3.3 McCabe의 5가지 범주

McCabe는 객체지향적 메트릭스를 5가지 범주로 나누어서 범주별 메트릭스 적용 방법을 제시했는데 이는 캡슐화(Encapsulation), 다형성(Polymorphism), 상속성(Inheritance), 품질(Quality), 입도(Granularity)의 5가지이다. McCabe는 기존의 객체지향적 메트릭스와 Cyclomatic Complexity를 변형한 방법으로 5가지 범주에 각각의 여러 가지 메트릭스들을 적용시켰다[1].

3. 자바 프로그램의 그래픽 분석 방법

3.1 그래픽 분석 개요

프로그램의 분석방법은 여러 가지 방법들이 존재하는데, 본 장에서는 이러한 시스템의 동적인 부분과 테스트 케이스 선정을 위해 기존의 도구들에서 많이 사용되어지고 연구되어지는 그래픽 분석 방법에 대하여 설명한다.

3.2 클래스 다이어 그램

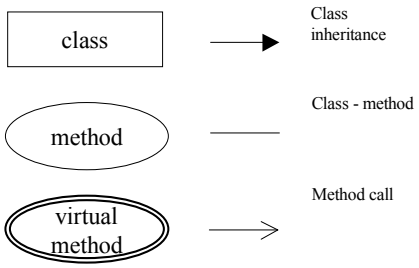
클래스 다이어그램은 클래스와 클래스내에서 선언된 메서드, 어트리뷰트등을 표시하고, 클래스간의 일반화(Generalization), 연관(Association), 의존(Dependency)등의 관계들을 표시할 수 있기 때문에 시스템의 정적인 구조를 잘 표현할 수 있다.

3.3 시퀀스 다이어그램

시퀀스 다이어그램은 시간에 진행에 따른 메세지 순서를 강조한다. 시퀀스 다이어그램은 객체, 시간, 메시지, 객체 생명선 등으로 표시되며, 시간에 따른 객체간의 메시지 전달과 객체들의 생성과 소멸을 시간 순으로 표시하기 때문에 시스템의 동적인 부분을 표시하는데 많이 사용되어진다.

3.4 CMRG 표기법

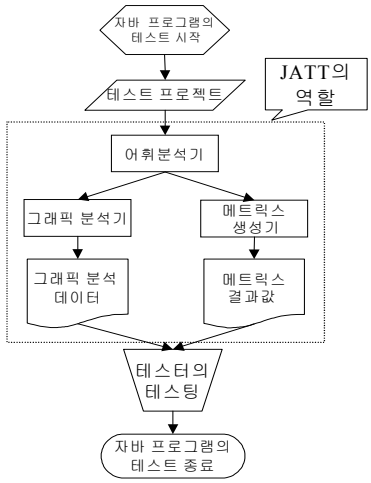
CMRG 표기법은 클래스와 메서드, Virtual 키워드로 선언된 메서드들을 상속과 호출 그리고 클래스와 메서드의 연결관계를 표현해 줌으로써 시스템의 정적인 구조와 각 메서드들의 호출관계를 한눈에 알아볼 수 있으며,[15][17][18][19] CMRG 표기법은 다음의 [그림 3-1]과 같다.



[그림 3-1] CMRG 표기법

4. 자바 테스팅 도구(JATT)의 기능 설계

4.1 JATT(Java Testing Tool)의 구조



[그림4-1] JATT의 구조

[그림4-1]은 JATT의 구조를 보여주고 JATT가 전체적인 프로그램의 테스팅에서의 역할을 보여준다. JATT는 테스트 프로젝트를 입력으로하여 그래픽 분석기를 통한 그래픽 분석 데이터와 메트릭스 생성기를 통한 메트릭스 결과값을 테스터에게 제공하고 테스터는 이러한 결과 자료를 바탕으로 자바 프로그램의 테스팅을 수행한다[18][19].

4.2 자바 프로그램에 적용할 소프트웨어 메트릭스

기존의 객체지향 메트릭스들 중에서 자바 언어에 적용할 수 있는 메트릭스들을 추출하고, 필요에 따라서는 기존의 메트릭스들을 자바 언어에 알맞게 또는 측정하기 편리한 형태로 변환하기도 한다. 또 자바 언어에만 존재하는 특성을 고려하여 새로운 메트릭스를 제안하고 각각의 적용될 메트릭스의 범위를 정의한다.

4.2.1 기존의 메트릭스

시스템에 메트릭스를 적용하는데는 아무리 시스템이 객체지향적으로 잘 작성되어져 있다 할지라도 시스템에 있어서 구조적인 부분은 반듯이 존재하기 마련이다. 따라서 본 논문에서는 구조적, 객체지향적 소프트웨어 메트릭스들을 적절하게 선별하여 적용하도록 한다.

韓國領 독도 Dokdo is Korean Territory

우선 시스템의 객체지향적인 특성을 고려한 메트릭스들은 McCabe가 정의한 5가지 범주에 의해서 나누고, 각각의 범주마다 적용될 메트릭스들을 다음의 [표4-2]에 나타내었다[18][19].

4.2.2 생성되거나 변형된 메트릭스

기존에 사용되어지던 객체지향적 소프트웨어 메트릭스중에 객체지향 프로그래밍 언어인 C++을 기본으로 하고 있는 메트릭스들은 측정 방법을 자바에 직접 적용할 수 없는 경우가 종종 있고, C++에는 없는 개념에 대한 부분을 측정하는 메트릭스는 거의 존재하고 있지 않은 실정이다. 따라서 자바 언어의 특징과 상이한 부분은 수정되어져야 하며, 자바에만 존재하는 부분에 대한 메트릭스의 생성이 필요하다.

다음에 수정되거나 새로이 생성된 메트릭스들을 정의하였다.

가. pctinner(Percent of Inner class)

자바 언어에는 C++과는 달리 클래스 내부에 또다른 클래스를 선언할 수 있는 Inner Class라는 것이 존재하는데, 이러한 Inner Class는 생성과 소멸이 일반 클래스에 선언된 메서드와 어트리뷰트들과 같기 때문에 자주 사용하지 않고 간단한 클래스를 사용하는 경우에 사용하면 편리하다. 그러나 자주 사용할 경우에는 클래스 내부에 클래스가 많아지고 클래스의 품질이 낮아지기 때문에 유지보수성이 떨어지고 재사용성도 떨어지게된다.[16]

·측정방법

- pctinner = ((Inner Class의수) / (전체 클레스의수))*100

나. overloadcnt(Overloading Count)

자바 언어에서는 세 가지 방법의 다형성을 사용하여 코드를 단순화 시킬 수 있는 방법을 제공한다. 첫 번째는 상속성을 사용하는 방법이고, 두 번째는 오버로딩을 사용하는 방법, 세 번째는 인터페이싱을 사용하는 방법이 있는데[20], overloadcnt는 이러한 다형성의 특징중에 오버로드 특징만을 체크하는 메트릭스를 본 논문에서는 기존의 메트릭스와는 달리 추측이 아닌 직접적인 오버로드된 메서드의 수를 계산한다.

·측정방법

- 클래스내에서 같은 이름을 갖는 메서드의 수

다. interfacecnt(Interfacing Count)

자바 언어의 세 가지 다형성 특징중에 하나인 인터페이싱의 수를 카운트한다. 인터페이싱 같은 자바의 특징은 시스템의 복잡도를 줄이는데 있어서 큰 효과를 가져올 수 있는데 이러한 인터페이싱에 정확한 수를 측정함으로써 시스템의 설계 품질 측정을 할 수 있도록 지원한다.

•측정방법

- 시스템에서 정의된 interface의 수

라. abscnt(Abstract Class Count)

자바 언어의 특징중에 하나가 바로 추상화 형이 있다는 점인데, 이러한 추상 클래스는 클래스에 대한 캡슐화 정도를 높일 수 있고 약간은 다형성과도 관련이 있을 수 있다. 그러나 이러한 추상 클래스에 대한 메트릭스는 존재하고 있지 않기 때문에 본 논문에서는 이러한 추상클래스의 수를 측정하여 시스템의 캡슐화 정도를 측정하는데 사용하고자 한다.

•측정방법

- 시스템에서 abstract형으로 선언된 클래스의 수

마. LCOM(Lack of Cohesion in Method)

CK 메트릭스에서 사용하고 있는 LCOM 측정방법은 계산의 결과값이 타당성이 없다는 것이 증명되었기 때문에 다른 방법의 계산법이 필요하게 되었는데 본 논문에서는 Henderson과 Seller가 정의한 식을 사용하였다[5][6][16].

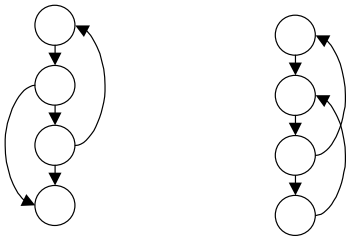
•측정방법

$$LCOM = \frac{\frac{\text{전체변수사용수}}{\text{변수수}} - \text{메서드수}}{1 - \text{메서드수}}$$

(메서드수가 1이면 LCOM값은 0으로 계산한다.)

바. Essential Complexity(ev(G))

자바에서는 goto 키워드가 존재 하지만 이를 사용할 수 없기 때문에 goto문을 사용한 비구조화 루틴은 사용할 수 없다. 따라서 프로그램의 비구조화 루틴을 여기에서 재정의 하는데 이는 반복문에서의 continue문과 break문을 사용한 경우를 측정한다. 자바 언어에 맞게 재정의된 비구조화 루틴을 [그림 4-2]에 보였다.



루프문에서의 continue문과 break문의 사용　　루프문에서의 Label이 있는 continue문과 break문의 사용

[그림4-2] 자바 언어에 맞게 재정의된 비구조화 루틴

4.2.3 메트릭스의 적용범위

韓國領 독도 Dokdo is Korean Territory

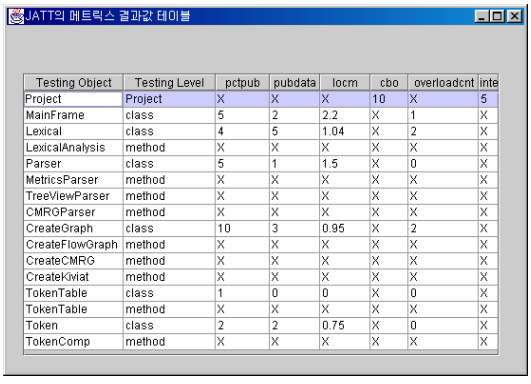
메트릭스범주	메트릭스이름	적용범위		
		시스템	클래스	메서드
Encapsulation	pctpub		O	
	pubdata		O	
	abscnt	O		
	LCOM		O	
	cbo	O		
Polymorphism	overloadcnt		O	
	interfacecnt	O		
	rfc		O	
Inheritance	dept		O	
	rootcnt	O		
	noc		O	
Quality	pctinner	O		
	wmc		O	
	maximum v(G)	O		
	maximum ev(G)	O		
Granularity	v(G), ev(G), iv(G)의 최대값		O	
	v(G), ev(G), iv(G)의 최소값		O	
	v(G), ev(G), iv(G)의 중간값		O	
	v(G), ev(G), iv(G)의 합		O	
Complexity	v(G)			O
	ev(G)			O
	iv(G)			O

[표4-1] 메트릭스의 적용범위

소프트웨어 메트릭스들은 저마다 적용될 수 있는 범위가 있는데 왜냐하면 이들은 메트릭스마다의 측정되는 특징이 저마다 다르기 때문에 나타난다. 이러한 메트릭스의 특징을 고려하여 다음에 메트릭스마다의 적용범위를 [표4-1]에 나타내었다[18][19].

4.3 메트릭스 생성기

메트릭스 생성기는 어휘분석기를 통하여 나온 토큰 테이블을 참조하여 각각의 메트릭스들을 측정할 수 있는 파서에 의하여 메트릭스 값을 산출하여 주는데 결과값은 테이블로 표현되어지고 결과 화면은 [그림4-3]와 같다.



Testing Object	Testing Level	pctpub	pubdata	locm	cbo	overloadcnt	inte
Project	Project	X	X	X	10	X	5
MainFrame	class	5	2	2.2	X	1	X
Lexical	class	4	5	1.04	X	2	X
LexicalAnalysis	method	X	X	X	X	X	X
Parser	class	5	1	1.5	X	0	X
MetricsParser	method	X	X	X	X	X	X
TreeViewParser	method	X	X	X	X	X	X
CMRGParser	method	X	X	X	X	X	X
CreateGraph	class	10	3	0.95	X	2	X
CreateFlowGraph	method	X	X	X	X	X	X
CreateCMRG	method	X	X	X	X	X	X
CreateKiviat	method	X	X	X	X	X	X
TokenTable	class	1	0	0	X	0	X
TokenTable	method	X	X	X	X	X	X
Token	class	2	2	0.75	X	0	X
TokenComp	method	X	X	X	X	X	X

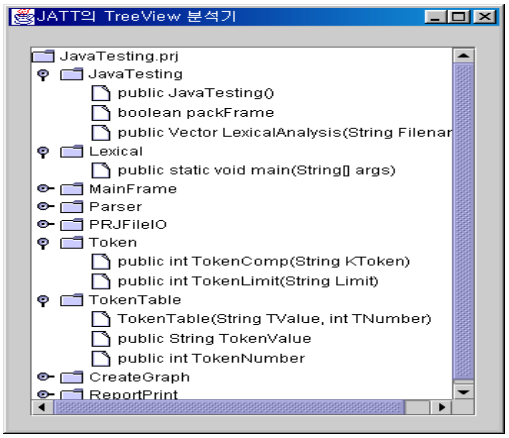
[그림4-3] 메트릭스의 테이블 표현

4.4 그래픽 분석기

시스템의 그래픽 구조 분석 도구는 어휘분석기를 통한 토큰 테이블을 입력으로하여 구조 분석을 위한 파서를 통하여 시스템의 그래픽 구조를 보여주는데 본 논문에서 작성한 도구에서는 TreeView와 CMRG 표기법과 Flow Graph로 표현을 해 준다. 다음의 4.4.1, 4.4.2, 4.4.3에 각각을 나타내었다.

4.4.1 TreeView

TreeView는 테스트 프로젝트의 전체적인 구조와 프로젝트에 속해있는 각 소스 파일에 존재하는 클래스와 클래스에 정의된 어트리뷰트와 메서드를 추출하여 트리 형태로 표현해 준다. 따라서 테스터는 프로젝트의 전체적인 시스템의 구조를 한눈에 알아볼 수 있으며, 이 TreeView를 통하여 각각의 클래스와 메서드들을 선택할 수 있게 한다[18][19].

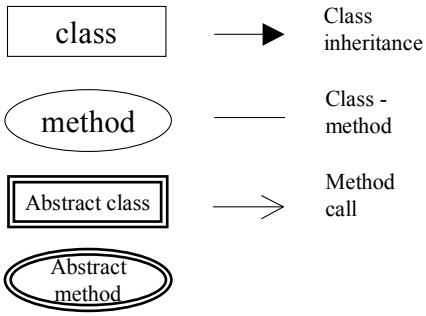


[그림4-4] TreeView를 통한 그래픽 분석 예

4.4.2 CMRG 표현

CMRG는 상속과 클래스-메서드의 관계, 그리고 호출관계에 대한 부분을 표현해 준다. 따라서 테스터에서 클래스 단위의 테스트 케이스 생성을 용이하게 하는 장점이 있다. 그러나 CMRG에서는 C++의 특성만을 강조하기 때문에 자바 언어의 특성에 맞게 수정하였다. [18][19].

수정된 CMRG 표기법은 다음의 [그림4-5]과 같다.



[그림4-5] 수정된 CMRG 표기법

5. 자바 테스팅 도구의 실행

JATT는 플랫폼에 독립적으로 실행이 가능하기 위해서 Java 언어를 기본으로 사용하고 있으며, 플랫폼에 독립적으로 실행시에 각 플랫폼에 알맞는 사용자 인터페이스를 지원하기 위하여 Swing(JFC)를 사용하고 있다.

5.1 구현환경

韓國領 독도 Dokdo is Korean Territory

•구현환경은 Intel Pentium II 400 Processor의 CPU 칩에 128MB의 RAM의 메모리를 사용하는 시스템에서 개발하였으며, 운영체제로는 Microsoft Windows 98을 사용하였다. JATT의 개발에 있어서 사용자 인터페이스는 JBuilder 3.0을 사용하여 디자인하였고, 그 외의 컴파일은 JDK1.2에서 컴파일 시켰다.

5.2 구현 결과

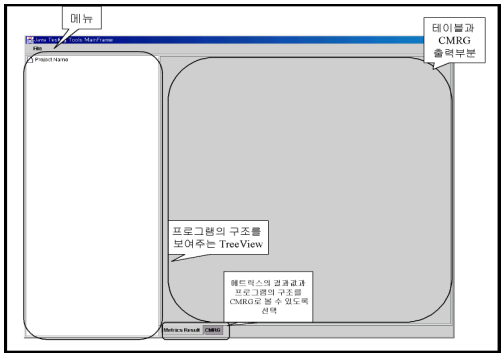
그래픽 구조 분석과 메트릭스 생성도구는 기본적으로 3가지 부분으로 구분할 수 있다.

첫 번째 부분은 프로젝트를 입력으로 하는 자바 파일들을 읽어들여 토큰을 나누어주는 어휘분석기이다. 어휘분석기는 프로젝트명과 파일명 그리고 각 파일에 존재하는 토큰들의 Token Value와 Token Number 값을 주로한다.

두 번째 부분은 어휘분석기에서 입력으로 하는 프로젝트를 입력으로 하여 메트릭스를 생성해주는 메트릭스 생성기이다. 메트릭스 생성기는 어휘분석기를 통한 토큰의 결과를 가지고 본 논문에서 사용하고자 하는 메트릭스의 값을 계산해 나간다. 메트릭스의 결과 값은 테이블 형식으로 화면에 출력하여 보여준다.

세 번째 부분은 토큰들을 분류하여 시스템의 구조를 그래픽컬하게 표현해주는 그래픽 분석기이다. 그래픽 분석기는 시스템 전체의 구조를 TreeView와 CMRG 표기법을 사용하여 시스템의 구조를 보여주고 클래스와 메서드들의 호출관계를 표시하여 클래스 테스트 케이스를 만드는데 도움을 준다.

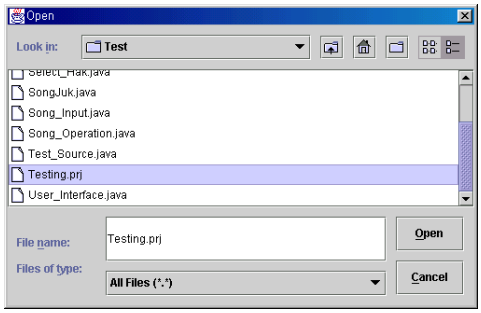
5.4 사용예



[그림5-1] JATT의 실행화면

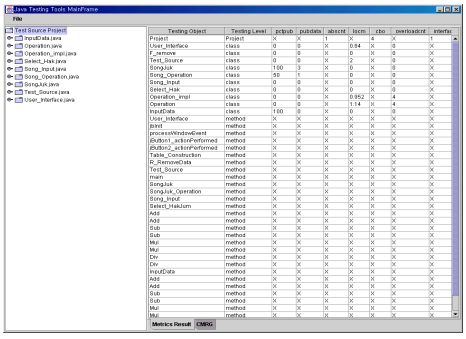
테스트 프로젝트를 설정하기 위해서 테스트 프로젝트 파일을 생성하게 되는데 이는 각 파일의 디렉토리명과 프로젝트 이름으로 구성된다. 이는 확장자를 prj로 표기하도록 한다. 테스트 프로젝트 파일은 다음과 같다.

테스트 프로젝트 파일을 생성하였으면 파일 메뉴에서 프로젝트 파일 열기를 선택하면 프로젝트 파일을 열 수 있는 파일 다이얼로그박스가 나타나게된다.



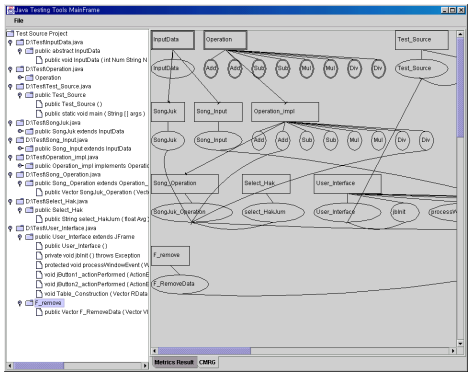
[그림5-2] 테스트 프로젝트를 선택하는 파일 다이얼로그박스

여기에서 테스트 프로젝트 파일(Testing.prj)을 선택하면 테스팅이 이루어지고 전체화면에 테스트 프로젝트의 구조를 좌측의 TreeView에서 보여주고 우측의 Table에서 메트릭스 결과값을 보여주게된다.



[그림5-3] 테스팅이 이루어진 JATT

그리고 우측 아래의 Tab 컨트롤의 버튼을 사용하여 CMRG를 선택하여 볼 수 있도록 한다.



[그림5-4] 테스트 프로젝트의 CMRG표현 결과

6. 결 론

프로그램의 테스팅은 개발 환경이 나아지고 프로그래밍 언어가 발전할수록 점점 더 복잡해지고 어려워져 가고 있다. 이러한 환경 속에서 보다 나은 테스팅 환경을 테스터에게 제공하는 것은 소프트웨어 공학의 소프트웨어 테스팅 부분에 있어서 많은 발전을 줄 수 있다. 하지만 기존의 도구들은 단순히 메트릭스를 적용하여 결과값을 산출해내거나 역공학을 통하여 시스템의 구조를 클래스 다이어그램으로 표현해주는 일들을 통시에 만족시켜주는 도구들은 없었다. 그리고 기존에 사용하던 클래스 다이어그램

韓國領
독도
Dokdo is Korean Territory

같은 경우에는 시스템의 설계에는 많은 도움이 되었지만 테스트시에는 많은 미흡한 점들이 있었고, 테스트에 좋은 CMRG 표기법으로 시스템을 분석해주는 도구들은 존재하지 않았다. 특히 객체지향적 소프트웨어 메트릭스와 구조적 소프트웨어 메트릭스를 동시에 적용하지 않았기 때문에 시스템과 클래스의 품질만을 측정하거나 시스템과 모듈의 품질만을 측정할 수 있었기 때문에 두 가지의 메트릭스의 동시 적용 또한 필요하였다.

본 논문에서 구현한 JATT(JAva Testing Tool)는 이러한 문제점들을 해결하기 위하여 기존에 존재하는 많은 구조적, 객체지향적 메트릭스들중에서 자바 언어에 적합한 메트릭스들을 추출하고, 이들을 적용시켰다. 자바 소스코드를 입력으로하여 소스코드에 대한 메트릭스 결과값을 전체적으로 볼 수 있도록 테이블로 표현하였다. 또한 시스템의 분석을 용이하게 할 수 있게 하기 위해서 TreeView 표현법을 사용하여 클래스와 클래스내에 정의된 메서드와 어트리뷰트들을 public, private, protected로 분류하여 나타내었으며, 시스템의 전체적인 구조를 쉽게 파악하기 위하여 CMRG 표기법을 사용하여 클래스와 메서드, 그리고 abstract 형으로 선언된 메서드와 클래스들의 관계를 상속, 호출, 클래스-메서드 관계를 테스터에게 보여주었다.

JATT는 시스템의 정적 분석과 품질을 평가하는데 있어서는 많은 장점들을 나타내고 있지만 실질적인 테스트 케이스 생성에 관한 부분은 고려되어있지 않다. 그리고 테스팅의 많은 부담을 덜어주기 위해서는 이러한 소스코드상에서의 역공학을 통한 테스팅 이외에 프로젝트 전체를 진행해 나가면서 각 단계마다 테스트를 해줄 수 있는 테스팅 자동화에 대한 연구가 좀 더 필요하다.

7. 참 고 문 헌

[1] T. J. McCabe. Structured Testing Using the McCabe Toolset, Third Edition, T. J. McCabe & Associates Inc, 1996.

[2] Carma McClure, The Three Rs of Software Automation, Prentice Hall,1992.

[3] S. R. Chidamber, C. F. Kemerer, "Towareds Metrics Suite for Object Oriented Design," in A. Paepcke,(ed.) Proc. Conference on Object-Oriented Programming : systems, Languages and Applications (OOPSLA'91), Oct 1991. Published in SIGPLAN Notices, Vol. 26 No. 11, pp. 197-211, 1991

[4] S. R. Chidamber, C. F. Kemerer, "A Metrics Suite for Object Oriented Desing," IEEE Trans. on Software Eng., Vol 20, No. 6, pp. 476-493, 1994

[5] B. Henderson-Sellers, Object-Oriented metri cs measurements of complexity, Prentice Hall, 1996.

[6] B. Henderson-Sellers, "Identifying Internal and External Characteristics of Classes likely to be useful and Structural Complexity Metrics," Proceedings of Internal Conference Object-oriented Information Systems. Dec, 1994.

[7] R. V. Binder, "Design for Testability in OOS," Comm. of ACM, Vol. 37, No. 9, sep, 1994.

[8] L. Briand, Sandro Morasca, "Property-Based Software Engineering Mesaurement," IEEE Trans. On Software Eng., Vol. 22, No. 2, Jan. 1996.

[9] L. Briand, J. Daly, V. Porter, J. Wust, "A Comprehensive Empirical Vallidation of Product Measures for Object-Oriented System," Technical Report ISERN-98-07, 1998.

[10] Roger S. Pressman, Software Engineering-A Practitioner's Approach, Fourth Edition, McGraw -Hill, 1998.

[11] Korson T., Mcgregor J. D., "Understanding Object-Oriented : A Unifying Paradigm," Communi cation ACM, Vol. 33, pp. 41-60, 1990.

[12] Boch, Rumbaugh, Jacobson, UML User's Guide, Addison Wesley, 1999

[13] 윤청, 성공적인 소프트웨어 개발 방법론 하권, 생능출판사, 1998.

[14] 최은만, 소프트웨어공학론, 회중당, 1999.

[15] 노미나 최병주, "상태기반 뮤테이션 테스트 기준을 적용한 클래스 테스트 프로세스," 정보과학회 논문지(B), 제2권, 제8호, 1998.

[16] 김재웅, 유철중, 장옥배, "자바 언어에 적용한 객체지향 설계 척도의 인자 분석," 제1회 한국소프트웨어공학 학술발표집, 제26권 2호, 1호, 1999.

[17] 김상영, 정지환, 김재웅, 황선명, "클래스 지향적 메트릭스에 의한 소프트웨어 테스팅 절차," 제4회 S/W품질관리 심포지움 논문집, pp 415-419, 1999.

[18] 정지환, 황선명, "자바 프로그램의 그래픽 구조 분석과 메트릭스 생성 도구의 설계," 한국정보과학회 춘계 학술발표 논문집, 제1권, pp 516-518, 1999.

[19] 김상영, 정지환, 황선명, "그래픽 구조 분석에 의한 자바 테스팅 도구의 설계 및 구현," 제2회 산·학·연 S/W공학기술학술대회 논문집, pp 197-201, 1999

[20] 이용운외,이것이Java1.2, 삼각형프레스, 1999

韓國領 독도 Dokdo is Korean Territory