자바 프로그램의 그래픽 구조 분석과 메트릭스 생성 도구의 설계

정지환* 황선명**
대전대학교 컴퓨터공학과

A Design of Graphic Structured Analysis and Metrics Tool for Java Program

Ji-Hwan Jung* Sun-Myung Hwang**
Dept. Computer Engineering, Taejon University

요 약

고속으로 발전하는 컴퓨터 분야에 있어서 소프트웨어는 좀더 복잡해지고 대형화 되어갔다 이에 따른 소프트웨어의 테스팅은 소프트웨어 재사용성이나 유지보수, 오류 검출을 하기위한 하나의 수단으로 사용되어져기고 있으며, 보편화 되어가고 있다 그리고 소프트웨어 개발 방법이 구조적 프로그래밍 기법에서 객체지향 프로그래밍 기법으로 변화 할수록 이에 따른 소프트웨어 테스팅 역시 구조적 프로그래밍 기법에서 사용하던 테스팅 방법들을 객체지향적 개념에 맞게 바꾸어 나가는 연구들이 많이 이루어지고 있다.

본 논문에서는 이러한 객체지향 테스팅 기법에서 사용하는 메트릭스들을 선정하여 이를 자바 언어로 작성된 프로그램에 적용하고 그 결과물들을 그래픽컬한 표현으로 나타내어 편리한 테스트 환경을 지원하는 도구를 설계 및 구현하였다.

1. 서 론

객체지향 프로그래밍의 방법론이 발전하는 속도에 비례하여 테스트 기술의 발전도 필요하게되었다. 설계 방법이 아무리 좋고 프로그래밍을 완벽하게 하더라도 오류가 있을 수 있기 때문이다 또한 객체지향 방법이 동원되는 복잡한 시스템은 결과를 예측하기 어려운 경우가 많다.[3] 구조적 프로그래밍 방법으로 작성된 프로그램은 모듈(함수)단위의 테스트를 중심으로 하여 모듈에 대한 어휘분석기를 통한 토큰들을 사용한 메트릭스의 결과값을 얻어내는 반면에 객체지향 프로그래밍 방법으로 작성된 프로그램은 모듈(메서드) 레벨에서의 메트릭스 적용과 클래스 레벨에서의 메트릭스 적용을 할 수 있다 따라서 주 단위가 클래스이며 객체지향 개념인 상속성, 캡슐화, 다형성등에 대한 고려를 하지 않을 수 없기 때문에 객체지향적인 접근 방법이 필요하다

본 논문에서 적용할 자바언어는 최근 각광을 받고 있는 객체지향성을 가장 잘 반영한 언어이며, 수정이 용이하고, 적은 비용으로도 유지보수가 뛰어나고, 재사용성과 확장성이 뛰어나다는 평가를 하고 있다 그렇기 때문에 많은 분야에서 사용되어진다 그러므로 자바 프로그램에 객체지향 소프트웨어 메트릭스를 적용한 테스팅 도구가 필요한 것이 현실이다.[7]

따라서 본 논문에서는 자바로 작성된 소스코드를 입력으로 소스코드를 분석하고, 구조적 소프트웨어 메트릭스와 객체지향 소프트웨어 메트릭스들을 적절히 사용하여 프로그램의 클래스와 모듈 단위의 복잡도 등을 측정하고, 이 결과를 그래픽컬하게 표현할 수 있는 메트릭스 적용 도구를 설계하고 구현하도록 한다

* 대전대학교 컴퓨터공학과 석사과정
** 대전대학교 컴퓨터공학과 교수

2. 소프트웨어 메트릭스

객체지향 테스트는 클래스 중심으로 이루어진다. 따라서 객체지향 개념을 고려하여야 하는데 이들은 소프트웨어의 재사용과 유지보수에 상당히 많은 영향을 미치며 시스템마다 이들에 대한 평가를 달리할 수 있다 [5] 이러한 소프트웨어의 평가는 소프트웨어 메트릭스들에 의해서 이루어지는데 크게 구조적 소프트웨어 메트릭스와 객체지향 소프트웨어 메트릭스로 나누어진다.[2][3]

2.1 구조적 소프트웨어 메트릭스

아무리 객체지향 프로그래밍 방법으로 소프트웨어를 개발한다 하더라도 하나의 모듈(메서드)내에서는 구조적인 구문들을 갖기 마련이다 따라서 메서드에 적용하는 메트릭스들은 구조적 소프트웨어 메트릭스를 그대로 사용하기로 하며 McCabe 의 메트릭스를 사용한다.

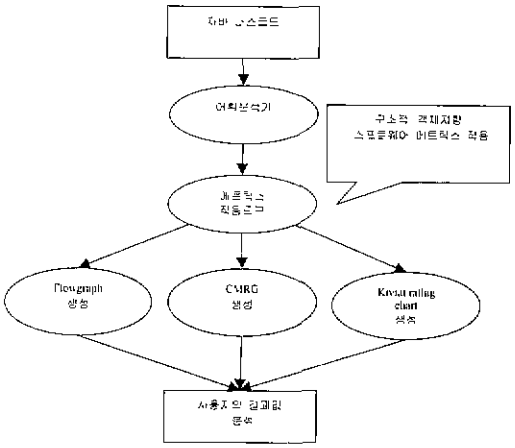
- ✓ McCabe 의 메트릭스 이 메트릭스는 소스코드 상의 조건문의 수와 모듈간의 호출정보를 이용하여 그에 대한 복잡도를 측정할 수 있다 [1]

2.2 객체지향 소프트웨어 메트릭스

객체지향 테스트 메트릭스는 주로 클래스의 품질, 재사용성, 유지보수성등을 평가하는데 사용되어진다 따라서 상속성, 캡슐화, 다형성등을 주로 평가하게 되어진다 본 논문에서 설계하고 구현할 메트릭스 적용 도구는 McCabe 가 강의한 클래스의 캡슐화, 다형성, 상속성, 품질, 밀도의 5 가지 측면에서 메트릭스를 적용하도록 한다 [1]

3. 자바 프로그램을 위한 메트릭스 적용 도구

메트릭스 적용도구는 기본적으로 어휘분석기와 메트릭스 적용도구 그리고 결과값을 표현하기 위한 그래프 생성기 들로 구성되어진다



<그림 1> 지비 프로그램을 위한 메트릭스 적용 도구의 구조

3 1 어휘분석기

메트릭스를 적용하기 위해 소스코드의 어휘 분석은 반듯이 필요하다 여기에서 어휘분석기는 약간의 파서 기능도 갖고 있다 이러한 어휘분석기의 역할은 소스코드를 입력 받아 토큰을 나누고 각 토큰이 의미하는 바가 무엇인지(키워드, 사용자정의어, 연산자, 구분지, 리터럴 등) 가려낸다. 특히 키워드 중에서 조건문, 반복문등을 구별하여 카운트하고, 사용지 정의어에서는 클래스 명과, 메서드 명들을 분류하고 각 메서드들에서 호출되는 메서드들을 분류한다.

3 2 메트릭스 적용

메트릭스는 앞에 설명한 메트릭스들을 어휘분석기에서 얻어낸 정보를 가지고 메트릭스들을 적용한다 사용되어지는 메트릭스들은 다음과 같다

3 2 1 메서드에 적용될 메트릭스[1]

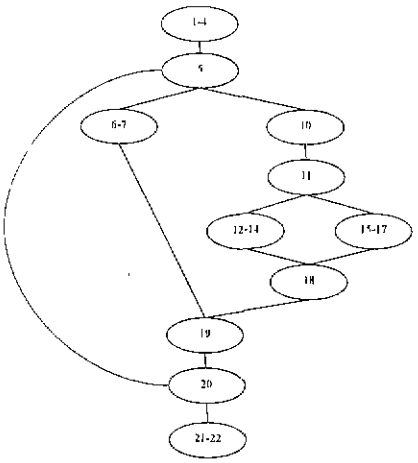
- ✓ Cyclomatic complexity : v(G)e – n + 2(e . edge, n : node)
- ✓ Essential complexity ev(G) flowgraph 상에서 구조적인 루틴을 제외한 flowgraph 의 v(G)값
- ✓ Module design complexity iv(G) flowgraph 상에서 메서드 호출과 관련된 부분을 제외한 flowgraph 의 v(G)값
- ✓ Design complexity : S_0 메서드 호출 관계에서 허위 메서드들의 iv(G)값의 누계(중복 없음)
- ✓ Integration complexity $S_1 = S_0 - n + 1$

메서드에 적용되는 메트릭스들은 flowgraph 를 기본으로 한다 [1] 다음에 flowgraph 생성 예를 나타내었다.

3 2 2 클래스에 적용될 메트릭스

클래스에 적용될 메트릭스들은 앞에서 언급한 바와 같이 5 가지로 분류했다. 이것은 McCabe 가 정의한 것이며, 자바 특성상 적용할 수 없는 메트릭스들은 적용하지 않았고, 자바에 특성에 맞게 메트릭스들을 수정 또는 추가 하였다

```
1   CarStart()
2   {
3   boolean engineState = false;  carStart = false
4   int gasLevel = 0
5   while(carStart != true){
6       if(engineState == true){
7         System.out.println("엔진이 작동 중입니다.");
8         carStart = true
9       }
10      else{
11        if(gasLevel >= 10){
12          System.out.println("자동 점화를 시동합니다.");
13          engineState = true
14        }
15        else{
16          System.out.println("가스를 넣고 있습니다.");
17          gasLevel = 10
18        }
19      }
20    System.out.println("차가 출발합니다.");
21  }
```



<그림 2> Flowgraph 의 생성 예

- ✓ Encapsulation
  - pctpub 클래스내의 public 과 protected 데이터의 퍼센트[1]
  - pubdata . 클래스내에서 엑세스된 public 과 protected 데이터의 수[1]
  - abscnt : 클래스중 abstact 형으로 선언된 클래스의 수(N)
  - locm 응집도의 결핍정도[3][7]

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

  - cbo 상속관계에 없는 클래스 사이에 익세스된 멤버 데이터들의 수[1]
- ✓ Polymorphism
  - overloadcnt : 하나의 클래스에 같은 이름의 메서드의 수(N)
  - interfacecnt 서로다른 클래스에서 같은 이름의 메서드의 수(N)
  - rfc : 한 클래스에서 호출하는 메서드들의 수[1]
- ✓ Inheritance[1]
  - depth : 클래스의 깊이
  - rootcnt 루트 클래스의 수
  - noc 자식 클래스의 수
- ✓ Quality
  - pctinner : 클래스중 inner 클래스의 퍼센트[7]

- wmc class에서 정의된 메서드의 수[1]
- maximum v(G)

$$\frac{\text{하나의 클래스내에 메서드들중 max}(v(G))}{\text{클래스의갯수}}$$

- maximum ev(G)[1]

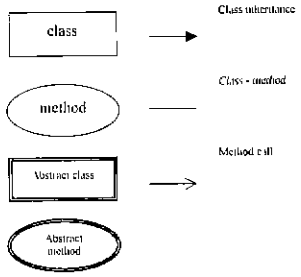
$$\frac{\text{하나의 클래스내에 메서드들중 max}(ev(G))}{\text{클래스의갯수}}$$

Granularity[1]

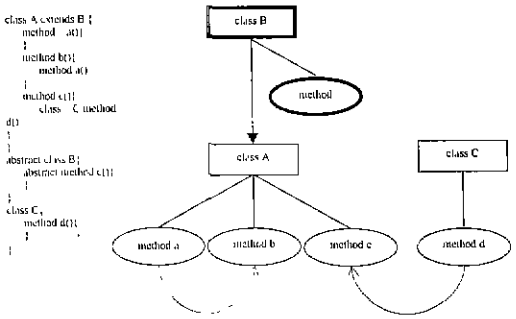
- 하나의 클래내에 메서드중에 v(G), ev(G), iv(G)의 최대값
- 하나의 클래스내에 메서드중에 v(G), ev(G), iv(G)의 중간값
- 하나의 클래스내에 메서드중에 v(G), ev(G), iv(G)의 최소값
- 하나의 클래스내에 메서드중에 v(G), ev(G), iv(G)들의 합

3.3 그래픽칼한 메트릭스 표현

테스트 도구에서 아무리 산출된 결과값이 믿을 수 있다고 원치라고 결과값이 보기좋게 출력되지 않으면 사용자가 판단을 내리기 어렵다 따라서 메트릭스 적용 결과를 그래픽컬하게 표한하는 방법 또한 중요하다. 여기에서는 전체적인 클래스들의 디자인들을 표시하는데는 CMRG(Class Method Relation Graph) 표기법을 사용하여 표현하였다.[6]



<그림 3> CMRG 표기법[4]

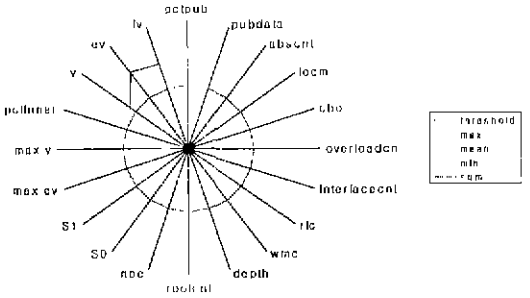


<그림 4> CMRG 표기법의 예

원래의 CMRG 표기법에서는 virtual로 선언된 메서드도 표현하지만 자바에서는 virtual 키워드가 없고 대신에 abstract 형이 있는데 이는 클래스에도 적용되기 때문에 기존의 CMRG 표기법을 변형하였다.

그리고 메트릭스들의 결과값을 표현하는데는 kiviat rating chart를 사용해 표현하도록 한다 Kiviat rating chart에서 각 메트릭스들의 임계값을 표시해주고 각 메트릭스들의 값들을 표현해줌으로써 사용자가 쉽게 시스템을 평가할 수 있게 알 수 있다

또한 메트릭스들 중에 최대, 최소, 중간값들은 색만 변경하여 그려줌으로써 chart가 약간 간소화 될 수 있다 아래에 Kiviat rating chart를 표시하였다.[1]



<그림 4> Kiviat rating chart

4. 결 론

소프트웨어 메트릭스의 목적은 생산품의 품질의 같 이해하고, 효율성을 평가하고 작업의 품질을 향상시키기 위함이다.[2] 지금까지 소프트웨어에 메트릭스 적용에 대한 연구는 너무나도 많이 이루어져 왔다 또한 많은 테스트 도구들에서도 이러한 내용들을 지원하고 있다 그러나 이러한 도구들이 대부분 특정 언어(C++)의 특성만을 지원하고 있다.[7]

본 논문에서는 자바에 알맞은 메트릭스들을 선정하고, C++과 다른 자바의 특성을 고려하여 abstract 형과 overload에 관련된 메트릭스를 생성하고 inner 클래스를 위한 메트릭스를 적용하였다 또한 CMRG 표기법을 사용하여 클래스와 메서드들의 구조를 쉽게 파악하게 하였고, 메트릭스 결과값을 Kiviat rating chart를 사용하였다 이러한 그래픽컬한 표현은 사용자가 시스템을 평가하는데 있어서 좀 더 편리한 환경을 제공할 수 있다

참 고 문 헌

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

[2] Mark Lorenz and Jeff Kidd, "Object-Oriented Software Metrics", Prentice Hall, 1994

[3] B. Henderson-Sellers, "Object-oriented metrics measures of complexity", Prentice Hall, 1996.

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

[5] L Briand, Sandro Morasca, "Property-Based Softeare Engnnering Measurement," IEEE Trans On Software Eng., Vol 22, No 2, Jan. 1996

[6] 노미나, 최병주, "상태기반 듀테이션 테스트 기준을 적용한 클래스 테스트 프로세스", 정보과학회 논문지(B) 제 25 권 제 8 호, 1998

[7] 김재웅, 유철중, 정옥배, "자바 언어에 적용한 객체지향 설계 척도의 인자 분석", 제 1 회 한국 소프트웨어공학 학술발표논문집 제 1 권 1 호, 1999