Journal of The Korea Society of Computer and Information
Vol. 20, No. 1, January 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.1.197

소프트웨어 모듈 심각도 측정을 위한 메트릭 집합

홍 의 석*

A Metrics Set for Measuring Software Module Severity

Euy-Seok Hong*

요 약

모든 소프트웨어 결함들이 시스템에 같은 정도의 영향을 미치는 것이 아니므로 결함이 미치는 충격의 정도를 나타내는 결함 심각도는 소프트웨어 품질 관련 작업들에 중요한 역할을 하고 있다. 결함 심각도 관련 기존 연구들은 심각도 레벨은 정의하였지만 품질 작업의 기본 단위인 모듈의 심각도에 관한 언급은 거의 없었다. 본 논문에서는 심각도 레벨이 증가함에 따라 심각도 값이 급격히 증가하는 심각도 성질을 이용하여 결함 심각도 메트릭을 지수 함수 형태로 정의한 후, 모듈 내부의 결함 수와 결함 심각도 메트릭에 기반한 새로운 모듈 심각도 메트릭 집합을 정의하였다. 제안 메트릭들의 적용가능성을 보이기 위해 Weyuker 기준들을 이용한 분석적 검증과 NASA 공개 데이터 집합을 이용한 실험적 검증을 수행하였으며, 제안 메트릭들 중 ms는 모듈의 심각도 정량화에, msd는 심각도에 기반한 시스템간의 비교에 매우 유용하게 사용될 수 있다는 것을 보였다.

▸ Keywords : 결함 심각도, 모듈 심각도, 소프트웨어 메트릭, 메트릭 검증

Abstract

Defect severity that is a measure of the impact caused by the defect plays an important role in software quality activities because not all software defects are equal. Earlier studies have concentrated on defining defect severity levels, but there have almost never been trials of measuring module severity. In this paper, first, we define a defect severity metric in the form of an exponential function using the characteristics that defect severity values increase much faster than severity levels. Then we define a new metrics set for software module severity using the number of defects in a module and their defect severity metric values. In order to show the applicability of the proposed metrics, we performed an analytical validation using Weyuker's properties and experimental validation using NASA open data sets. The results show that ms is very useful for measuring the module severity and msd can be used to

•제1저자 : 홍의석
•투고일 : 2014. 9. 15, 심사일 : 2014. 10. 8, 게재확정일 : 2014. 10. 29.
* 성신여자대학교 IT학부(School of Information Technology, Sungshin Women's University)
※이 논문은 2012년도 성신여자대학교 운정 글로벌프로젝트 지원사업 지원에 의하여 연구되었음.

compare different systems in terms of module severity.

▸ Keywords : defect severity, module severity, software metrics, metrics validation

I. 서 론

소프트웨어 결함(defect)은 소프트웨어 요구사항이나 명세를 만족시키지 못하게 만드는 소프트웨어 산물의 문제나 결핍이다[1]. 소프트웨어 개발이 시작되면서부터 오늘날까지 소프트웨어 결함을 줄이려는 노력은 계속되어왔다. 결함을 완전히 방지하는 것은 매우 어려우므로, 개발 집단은 소프트웨어 테스팅 기법들을 통하여 결함들을 찾아내어 없애는 작업을 수행해야한다. 하지만 이러한 결함 해결 방법은 비용이 많이 필요하므로 전체 시스템을 구성하는 각 부분들의 결함경향성을 예측하는 결함 예측 모델이 비용 절감에 도움을 주고 있다[2]. 결함 예측 모델들은 대부분 결함경향성 유무만을 판단하는 분류 모델형태이며 소프트웨어를 정량화한 메트릭들을 입력으로 사용한다. 메트릭 기반 결함 예측 모델에 관한 연구들은 NASA IV&V 메트릭스 데이터 프로그램[1], PROMISE 레포지토리[2]와 같은 공개 데이터 집합의 등장에 따라 2000년대 중반부터 급속히 많아졌다[3, 4].

결함 유무만을 판단하는 예측 모델의 문제점은 결함이 가지는 심각도나 우선도 같은 속성들을 고려하지 않았다는 것이다. 결함 속성들 중 가장 중요하게 여겨지는 것은 심각도이다. 결함 심각도는 소프트웨어 시스템과 사용자들에게 결함이 미치는 충격의 정도를 나타내는 척도이다[5]. 모든 소프트웨어 결함들이 시스템에 같은 정도의 영향을 미치는 것이 아니므로 소프트웨어 결함 심각도는 소프트웨어 품질 분야의 여러 결정에 매우 중요한 역할을 한다. 결함 예측 모델 역시 심각도에 기반한 예측 모델, 즉 고심각 부분과 저심각 부분을 나누어 예측할 수 있는 모델이 단순히 결함 유무만을 예측하는 모델보다 훨씬 효용성이 높다. 어떤 결함이 더 심각하냐는 정보는 소프트웨어 엔지니어나 프로젝트 관리자로 하여금 시스템의 어느 부분에 더 집중하여야하는가에 대한 해답을 줌으로써 보다 적은 비용으로 고효율의 테스팅 작업이나 리팩토링 작업, 자원 할당 작업 등을 가능케 한다[2].

품질 관련 작업들은 대부분 모듈단위로 이루어지므로 심각도 정보가 결함 단위로 제공되는 것보다는 모듈 단위로 제공된다면 대형 시스템의 작업을 하는 경우 매우 유용하게 사용될 수 있다. 하지만 기존 연구들과 개발 집단들에서 정의하고 사용한 심각도 정보는 모두 결함 삼각도였고 모듈 심각도에 대한 연구는 거의 없었다. 따라서 본 논문의 목적은 모듈이 가지고 있는 결함들의 정보를 이용하여 모듈의 심각도를 측정하는 메트릭 집합을 제안하는 것이다. 소프트웨어 모듈이란 시스템을 구성하는 기능 단위를 의미하며, 함수나 프로시져 또는 객체지향 시스템의 메소드가 이에 해당된다. 모듈 심각도는 모듈이 지닌 결함으로 인하여 소프트웨어 시스템과 사용자들에게 미치는 충격의 정도이다. 제안 메트릭들의 실험적 검증에는 NASA IV&V 메트릭스 데이터 집합을 사용하였다.

2장에서는 기존의 심각도 관련 연구들을 살펴보고, 3장에서는 제안하는 모듈 심각도 메트릭들에 대해 설명한다. 4장에서는 제안 메트릭들의 분석적, 실험적 검증 결과를 언급하고, 5장에서는 메트릭들의 여러 응용 부분들을 설명한 후, 6장에서 결론을 기술한다.

II. 관련 연구

1. 결함 심각도

결함을 관리하는 개발 집단이나 연구 집단은 결함 심각도 정보를 사용하여왔다. 결함 심각도는 정수나 실수 형태의 메트릭이 아닌 심각한 단계를 나타내는 서수 스케일(ordinal scale) 형태의 심각도 레벨로 정의되었다. IEEE Std. 1044-2009에서는 심각도 레벨의 예를 명시하였지만 개발 집단들은 기존에 정의하여 사용하던 레벨들을 많이 사용하고 있다. 이들이 정의하여 사용하는 심각도 단계의 범위는 3에서 7 단계까지 다양하지만 중심 부분을 형성하는 단계들의 내용은 비슷하며, 작은 레벨 값을 가질수록 더 심각한 결함임을 뜻한다. 결함 심각도는 사람이 분석을 통해 결정하므로 정확

1) http://mdp.ivv.nasa.gov
2) http://promise.site.uottawa.ca/SERepository

성이나 일반성 등에 문제가 생길 수 있다. 따라서 심각도 평가 시 텍스트 마이닝과 기계학습 기법 등을 이용하여 자동으로 평가하는 것을 도와주는 시스템도 등장하였다[6].

1.1 IEEE Std 1044-2009

결함을 위한 IEEE의 최신 표준안은 IEEE Std 1044-2009이다. 결함을 표현하기 위한 많은 속성들을 정의하였으며 심각도 속성의 예로 다음과 같은 5개의 레벨을 명시하였다.

Blocking - 테스팅이 불가능 또는 적절한 해결책을 진행할 수 없거나 알 수 없음
Critical - 기본적인 수행들이 불가피하게 중단되고, 안전성과 보안이 위태로워짐
Major - 기본적인 수행들이 영향 받지만 진행 가능함
Minor - 기본적이지 않은 수행들이 중단됨
Inconsequential - 수행에 중요한 영향이 없음

1.2 Bugzilla

대표적인 결함 추적 시스템인 Bugzilla[3]를 사용하는 Eclipse 등의 오픈 소스 프로젝트들은 결함들을 다음과 같은 7개의 심각도 레벨로 나누었다.

Blocker - 매우 심각한 결함으로 릴리즈,개발, 테스팅 작업을 막음
Critical - 시스템 고장, 데이터 손실, 심각한 메모리 누수
Major - 중요한 기능의 손실
Normal - 일반적인 문제들, 특별한 상황에서의 몇몇 기능의 상실
Minor - 중요하지 않은 기능의 손실
Trivial - 철자법, 정렬 오류 같은 외형상 문제들
Enhancement - 개선 요구(실제로는 결함이 아니지만 분류에 들어가 있다)

1.3 NASA 데이터 집합

NASA IV&V 메트릭스 데이터 집합을 구성하는 프로젝트들 중 몇몇은 결함 심각도 정보를 포함하고 있다. NASA 제공 문서에는 심각도 레벨에 대한 정의를 기술하고 있진 않지만 레벨 값은 1에서 5의 값을 가지며 1이 가장 심각한 결함을 뜻하고 5가 가장 심각하지 않은 결함을 뜻한다고 명시되어있다. 예를 들면 레벨 1 결함은 해결책이 없는 심각한 기능의 손실을 야기하고, 레벨 5 결함은 시스템에 어떤 중요한 영향도 끼치지 않는 가벼운 결함이라는 의미이다.

2. 모듈 심각도

모듈 심각도를 측정한 연구는 없었지만 심각도에 기반한 클래스의 결함경향성을 예측한 연구는 있었다. [7]은 객체지향 시스템인 NASA의 KC1 프로젝트를 이용해 고심각 및 저심각 결함 예측 모델을 제작하였으며, 사용한 C&K 메트릭들 중 대부분이 심각도에 기반한 클래스 결함경향성과 밀접한 관계가 있음을 보였다. 결함들은 결함 심각도 레벨 값이 1이면 고심각도 결함, 2~5이면 저심각도 결함으로 분류하였다. 고심각 결함 예측 모델은 고심각 결함이 하나 이상인 클래스를 결함경향으로 예측하며, 저심각 결함 예측 모델은 저심각 결함이 하나 이상인 클래스를 결함경향으로 예측하는 모델이었다. 즉, 고심각 결함이 하나 이상인 클래스를 고심각, 저심각 결함이 하나 이상인 클래스를 저심각 클래스로 나누었다. 이 연구는 모듈에 존재하는 결함의 수는 고려하지 않았다.

[2]는 소프트웨어 진화과정에서 소프트웨어 메트릭과 이를 이용한 예측 모델이 다음 릴리즈의 결함 경향 클래스들을 판별해낼 수 있는지 여부에 대한 연구를 Bugzilla를 사용한 Eclipse 프로젝트의 세 개의 릴리즈 데이터를 사용하여 수행하였다. 결함 예측 모델로 이진 결함 분류 모델과 결함 심각도에 기반한 다중 분류 모델을 사용하였으며 모델 구축을 위해 로지스틱 회귀분석을 사용하였다. 다중 분류를 위해 Bugzilla의 7개의 심각도 레벨들 중 중요하지 않은 두 개의 레벨들을 제외하고 나머지 레벨들을 다음과 같이 세 개의 카테고리로 분류하였다: 고영향(Blocker, Critical), 중영향(Major), 저영향(Normal, Minor). 해당 결함을 가진 클래스들을 각각 고심각, 중심각, 저심각 클래스들로 분류하였으며, 두가지 종류의 심각도 결함들을 가진 클래스는 각 분류에 한 번씩 삽입되었다.

심각도를 고려하지 않은 일반적인 이진 분류 결함 예측 모델들은 모듈이나 클래스를 단순히 결함 여부에 따라 결함 모듈과 비결함 모듈로 분류하였다. 예를 들면, NASA 데이터를 사용한 이진 분류 모델의 출력값인 결함 여부(defective) 불리언 변수는 다음과 같이 모듈의 결함 수를 나타내는 error_count 속성을 사용하여 명시적으로 정의하였다[8].

$$defective\ ?= (\text{error\_count} \geq 1) \quad (1)$$

3) http://www.bugzilla.org

III. 모듈 심각도 메트릭 집합

기존의 결함 심각도 레벨을 이용하여 새로운 결함 심각도 메트릭을 정의하고, 이를 기반으로 새로운 모듈 심각도 메트릭들을 정의 및 제안한다.

1. 고려 사항

모듈의 심각도에 영향을 미치는 요인은 모듈이 가지고 있는 결함의 수와 결함들의 심각도들이다. 만약 모듈에 존재하는 결함들이 모두 같은 심각도를 가진다면 결함수가 많은 모듈일수록 더 심각한 모듈이다. 또한 모듈들에 존재하는 결함수가 모두 같다면 더욱 심각한 결함들을 많이 가진 모듈일수록 더 심각하다. 결함 심각도는 레벨값으로 표현되므로 결함 심각도 레벨을 표현하기 위해 다음과 같은 표기를 사용한다.

$defectlevel_i$: 결함 i 의 결함 심각도 레벨. (2)
1~max_level 범위의 값을 갖는다.

제안하는 메트릭들을 설명하기 위해 그림 1의 예제 시스템을 사용한다. 예제 시스템은 5개의 모듈로 이루어지며 결함들은 1~3 범위의 심각도 레벨 값을 갖는다. 레벨 1은 가장 심각한 심각도 레벨을 의미하며, 심각한 결함일수록 짙은 색으로 표현하였다.

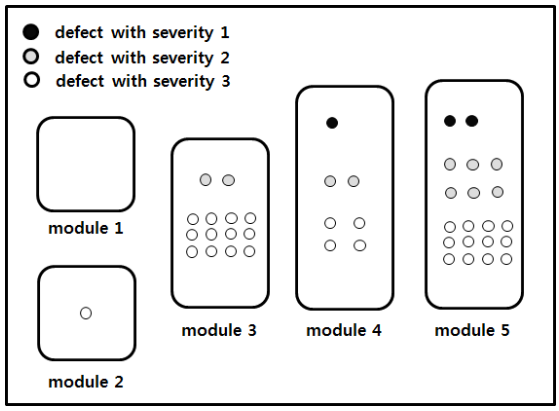

그림 1. 다섯 개 모듈들로 구성된 예제 시스템
Fig. 1. Example system with five modules

예제 시스템은 이진 분류 결함 예측 모델의 문제점을 명확하게 보여준다. module 1만 결함이 없으므로 module 1의 defective는 0이지만, 나머지 모든 모듈들의 defective는 1이다. module 2는 매우 가벼운 결함 1개를 가지지만, module 5는 매우 심각한 결함 2개와 중간 심각한 결함 6개, 가벼운 결함 12개를 지니고 있다. 두 모듈의 defective는 1이므로 식 (1)을 사용하는 이진 분류 모델은 이 두 모듈을 같은 결함경향 모듈로 여기지만 두 모듈이 가진 결함의 특성은 심각도 면에서 엄청난 차이가 있다. 단순한 관찰로 module 5의 모듈 심각도가 가장 심각하고, module 2는 가장 심각하지 않다는 것은 자명하다. 하지만 module 3과 module 4의 비교는 간단하지 않다.

2. 제안 메트릭 집합

2.1 결함 심각도 레벨(Defect Severity Level)

defectleveli 값은 작을수록 i가 더 심각한 결함임을 의미한다. 레벨 값이 클수록 더 심각한 결함임을 나타내기 위해 defectleveli를 이용하여 새로운 심각도 레벨 메트릭 dsli를 아래와 같이 정의하였다. 예를 들면 심각도 레벨이 1~5 범위의 값을 갖는 경우에 가장 심각한 결함 i의 defectleveli는 1이고, dsli는 5이다.

$$dsl_i = max\_level - defectlevel_i + 1 \quad (3)$$

2.2 결함 심각도(Defect Severity)

결함 심각도 레벨을 나타내는 dsl은 서수 스케일 메트릭이다. 즉, 심각도 레벨은 결함 심각도의 순서만을 정해줄 뿐 그 이상의 정보를 주지는 못한다. 심각도 레벨간의 심각도 차이는 선형 관계가 아니며, 심각도 레벨이 커짐에 따라 실제 심각도의 값은 선형 관계보다 훨씬 크게 증가한다. 이 차이의 정도는 소프트웨어 개발 프로젝트와 모듈의 성질, 결함의 미세한 성질 등에 따라 다르다. 즉, 어떤 프로젝트에서 발생한 두 결함의 심각도 레벨이 1과 3일 때, 레벨이 1과 3인 다른 두 결함이 존재한다고 하면 이 결함들의 심각도 차이는 같다고 할 수 없다. 또한 같은 심각도 레벨값을 갖는 결함들이라도 심각도 값은 서로 다를 수 있다. 따라서 심각도 레벨을 이용하여 심각도를 정확히 수식으로 정의할 수는 없다. 하지만 선형 관계보다 빠르고 크게 증가하는 것은 확실하므로 본 논문에서는 이러한 성질을 지닌 대표적인 형태 중 하나인 지수형 관계로 결함 심각도를 정의한다. 결함 심각도 ds_i는 결함 i의 심각도를 수치화한 메트릭이다.

$$ds_i = a^{dsl_i} \qquad s.t.\ a \geq 2 \quad (4)$$

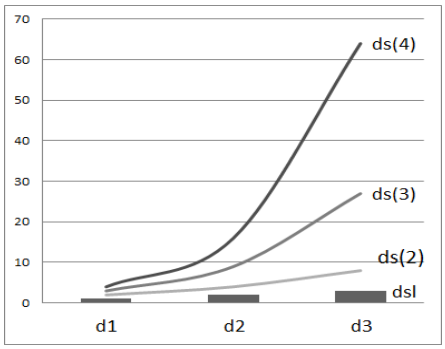

그림 2. dsl을 ds로 변환
Fig. 2. Transformation dsl to ds

그림 2는 dsl이 1, 2, 3인 결함의 ds 그래프를 나타낸 것이다. ds(2), ds(3), ds(4)는 a가 2, 3, 4인 경우를 나타내며 a 값을 증가시킴에 따라 결함 심각도값이 매우 빠른 속도로 증가함을 볼 수 있다.

2.3 모듈 심각도

모듈의 심각도를 나타내기 위한 여러 메트릭들을 정의한다. 모듈 m에 존재하는 결함들의 집합을 D_m이라 표시하였다. $|D_m|$은 m에 존재하는 결함들의 수를 의미한다.

① SSL(Sum of Severity Levels)

모듈에 존재하는 결함들의 심각도 레벨들의 합이다. 하지만 앞의 지적과 같이 심각도 레벨이 선형관계가 아니므로 이들의 단순합이라는 매우 큰 문제점이 존재한다. 또한 모듈의 사이즈가 커지고 결함이 많아지면 커질 가능성이 많아진다는 문제점이 있다. 그림 1의 module 3과 module 4의 ssl은 16과 11이지만, module 4가 가장 심각한 결함이 존재하므로 module 3이 module 4보다 더 심각하다고 보기는 어렵다. 이와 같은 결과는 ssl이 ds가 아니라 dsl을 사용하기 때문이다.

$$ssl_m = \sum_{i \in D_m} dsl_i \tag{5}$$

② MSL(Maximum Severity Level)

모듈에 존재하는 결함들의 심각도 레벨들 중 가장 큰 심각도 레벨이다. 가장 심각한 결함 하나가 그보다 작은 심각도 레벨의 많은 결함들보다 심각한 영향을 미칠 수 있으므로 이는 매우 중요한 척도이다. 하지만 높은 레벨 결함을 갖는 경우에는 중요한 의미를 가지지만 작은 레벨 결함들만을 갖는 모듈에서는 중요도가 떨어지며, 결함의 수를 전혀 고려하지 않은 척도이다. 그림 1의 예를 보면 module 5가 나머지 모듈보다 심각도가 높은 것이 확실하지만 msl_3, msl_4, msl_5는 2, 3, 3으로 세 모듈간의 변별력이 떨어진다. 기존 연구들 중 결함 심각도에 기반해서 모듈의 레벨을 평가한 연구들은 대부분 이 메트릭의 관점을 사용하였다. 예를 들면 [7]은 msl_m이 5인 결함을 고심각 결함, 나머지를 저심각 결함으로 분류한 후 전자를 지니고 있는 모듈은 고심각 클래스, 나머지를 저심각 클래스로 분류하였다. [2]는 msl_m이 7, 6인 결함을 고영향, 5인 결함을 중영향, 4, 3인 결함을 저영향 결함으로 정하고 각각을 지닌 클래스를 고심각, 중심각, 저심각 클래스로 분류하였다.

$$msl_m = \max_{i \in D_m} dsl_i \tag{6}$$

③ MS(Module Severity)

모듈에 존재하는 결함들의 결함 심각도들의 합이다. 레벨들의 단순합 문제와 msl의 문제점들을 해결하지만 역시 사이즈가 큰 모듈일수록 커질 가능성이 있다. 또한 ds를 정의한 식 (4)에서 a의 값을 달리함에 따라 변화가 심하다. 결함 심각도 레벨 수가 작아서 레벨 사이의 심각도 차가 큰 경우에는 a의 값을 크게 하고, 반대의 경우에는 작게 하는 것이 적당하다. a의 값을 표시한 ms를 ms(a)라 하면 그림 1의 예에서 module 3과 4는 ms(2)인 경우는 32, 24로 module 3의 심각도가 높지만 ms(3)인 경우는 54, 57로 module 4의 심각도가 높아진다.

$$ms_m = \sum_{i \in D_m} ds_i \tag{7}$$

④ MSM(Module Severity Mean)

모듈에 존재하는 결함들의 심각도 값들의 평균값으로, 모듈의 사이즈에 영향을 받지 않지만 결함의 수에도 영향을 받지 않는다. 평균값이므로 결함 심각도가 낮은 모듈들이 많아지면 msm은 작아진다. 그림 1의 예에서 module 4의 msm이 module 3과 5의 msm보다 크다. 그 이유는 두 모듈이 dsl이 1인 결함들을 너무 많이(12개) 갖고 있기 때문이다.

$$msm_m = \frac{ms_m}{|D_m|} \tag{8}$$

⑤ MSD(Module Severity Density)

ms_m은 사이즈가 큰 모듈일수록 더 커질 가능성이 많다. 따라서 모듈 m의 100 SLOC 당 모듈 심각도 값을 다음과 같이 정의하였다. 일반적으로 밀도 메트릭에 많이 사용되는 1 KLOC 대신 100 LOC을 사용한 이유는 1 KLOC 당 심각도 값이 너무 크기 때문이다. msd는 모듈의 사이즈에 독립적이면서도 모듈 심각도 정보를 잘 표현하고 있으므로 두 소프트웨어 시스템을 모듈 심각도 측면에서 비교할 때 사용할 수 있다. 비교 시스템에 비해 msd 분포가 명백하게 크다면 msd를 낮추기 위한 작업을 하는 것이 바람직하다.

$$msd_m = \frac{ms_m}{SLOC} \cdot 100 \quad (9)$$

표 1과 그림 3은 그림 1 예제 시스템에 제안 메트릭들을 적용한 결과이다.

표 1. 그림 1 시스템의 모듈 심각도 결과
Table 1. Module severity results of Fig.1 system

m	defect count	defective	msl	ssl	ms(2)	ms(3)	msm(2)	msm(3)
1	0	0	0	0	0	0	0	0
2	1	1	1	1	2	3	2	3
3	14	1	2	16	32	54	2.29	3.86
4	7	1	3	11	24	57	3.43	8.14
5	20	1	3	30	64	144	3.2	7.2

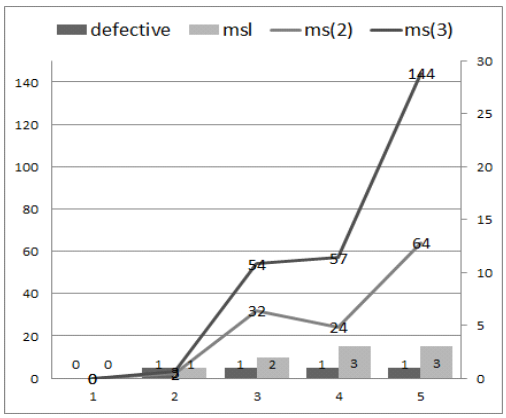

그림 3. 그림 1 시스템의 모듈 심각도 그래프
Fig. 3. Module severity graph of Fig. 1 system

예제 시스템의 각 모듈의 크기, 즉 SLOC는 나타내지 않았고 msd는 구할 수가 없으므로 결과에서 제외되었다. 앞에서의 언급과 같이 ms값으로 module 3과 module 4를 비교한 결과는 식 (4)의 a값에 의존하므로 a값의 올바른 결정은 모듈 심각도 메트릭 집합의 정확도를 결정하는데 매우 중요한 요소이다.

IV. 메트릭 검증

새로운 메트릭을 검증하는 방법에는 분석적 검증 방법과 실험적 검증 방법이 사용된다. 전자는 측정 이론(Measurement theory)[9, 10]이나 검증 기준(Evaluation criteria)[11] 등을 이용하는 것이며, 후자는 실제 소프트웨어 시스템을 입력으로 하여 결과 메트릭들을 구하여 해석해 봄으로써 메트릭들의 유용성을 검증하는 것이다. 제안 메트릭들을 검증하기 위해 검증 기준을 이용한 분석적 검증과 NASA 데이터 집합을 이용한 실험적 검증을 수행하였다.

1. 분석적 검증

대부분의 분석적 검증은 소프트웨어 복잡도 메트릭을 위한 것들이다. 이들 중 대표적인 것은 복잡도 메트릭이 가져야 할 아홉 개의 필요조건들을 제시한 Weyuker의 기준들을 이용한 검증이다[11]. IEEE의 정의에 의하면, 소프트웨어 복잡도란 "시스템이나 콤포넌트의 설계나 구현을 이해하고 검증하기 어려운 정도"라고 정의하고 있다[12]. 즉, 소프트웨어 복잡도 메트릭은 소프트웨어의 구조나 외부 입출력, 내부의 문장과 같은 여러 문법적 요소들을 이용하여 복잡도를 정량화하기 위한 것이므로 결함 수나 결함의 심각도 수준에 기반한 모듈 심각도 메트릭은 복잡도 메트릭이라 보기는 어렵다. 하지만 복잡도와 마찬가지로 심각도도 높을수록 안좋다는 점, 복잡도가 높아질수록 심각도가 높은 결함들이 많아질 것이라는 점은 모듈 복잡도와 모듈 심각도 상에 상당한 연관 관계가 있다는 것을 의미한다. 따라서 기준들의 만족 여부 보다는 정의한 메트릭의 이론적 분석을 위해 Weyuker의 기준에 적용해 볼 필요가 있다.

1.1 Weyuker의 기준

Weyuker가 제시한 복잡도 메트릭이 가져야 할 아홉 개의 필요조건은 다음과 같다. P, Q는 프로그램을 의미하고 |P|, |Q|는 P, Q의 복잡도 값을 의미한다. 모듈 복잡도에 적용 시에는 P, Q가 모듈이고 |P|, |Q|는 모듈 심각도에 해당한다.

1) (∃P)(∃Q) (|P|≠|Q|)
2) c를 0보다 크거나 같은 실수라 하면 복잡도가 c가 되는 유한개의 프로그램이 있다.
3) |P|=|Q|인 서로 다른 P, Q가 존재 가능하다.
4) (∃P)(∃Q) (P≡Q[4] & |P|≠|Q|)
5) (∀P)(∀Q) (|P|≤|P;Q|[5] & |Q|≤|P;Q|)
6) a: (∃P)(∃Q)(∃R) (|P|=|Q| & |P;R|≠|Q;R|)
   b: (∃P)(∃Q)(∃R) (|P|=|Q| & |R;P|≠|R;Q|)
7) (∃P)(∃Q) (Q는 P의 문장들을 재순열함 & |P|≠|Q|)
8) 만약 P가 Q의 바뀐 이름이면 |P| = |Q|
9) (∃P)(∃Q) (|P|+|Q| < |P;Q|)

1.2 검증 결과

조건 1)은 서로 다른 모듈 P, Q가 다른 심각도 값을 가질 수 있으므로 당연히 만족한다. 조건 2)는 일정한 심각도를 갖는 모듈이 유한개임을 의미한다. 문제는 이 조건에서 고려되는 모듈이 문법적으로 가능한 무한개의 모듈 집합이라는 것이다. Weyuker는 이런 의도로 조건2를 만들었으나, [13]은 유한개의 모듈 집합만을 고려 대상으로 하여 조건2를 검증 조건에서 제외하였다. 심각도 메트릭은 조건2를 만족하지 않는다. 왜냐하면 같은 결함 결과값을 갖는 모듈에 문법적으로 문제가 없는 문장들을 단계적으로 붙여 무한개의 모듈들을 만들 수 있기 때문이다. 조건 3)은 서로 다른 모듈들이 같은 심각도 값을 가질 수 있으므로 당연히 만족한다. 두 모듈 P, Q가 같은 입출력을 가지고 같은 기능을 수행하지만 모듈 내부의 구현은 다른 형태가 될 수 있으므로 다른 결함들을 가질 수 있기 때문에 조건 4)도 만족한다. P의 문장을 의미 있게 재순열하여 Q를 구성하였을 때 논리적인 결함에 영향을 줄 수 있으므로 조건 7)을 만족한다. 모듈 이름의 수정이 내부 결함에 영향을 미치지 않으므로 조건 8)은 당연히 만족한다.

Weyuker의 조건들 중 모듈 심각도 메트릭에 적용하기 모호한 것들은 P;Q가 들어간 조건들이다. Weyuker는 P;Q를 프로그램 부분들의 문법적 연결이라 기술하였지만 특별한 정의를 내리지 않았다. P;Q가 단순히 문법적으로 P와 Q를 순서적으로 결합하였고 P, Q, P;Q에 논리적 결함이 없다면 P;Q의 결함 집합은 P와 Q의 결함 집합들의 합집합이 될 것이므로 |P|+|Q| = |P;Q|가 성립될 것이다. 하지만 P와 Q에 논리적인 결함이 존재하고 P의 출력이 Q의 입력이 되는 P;Q의 경우에는 P;Q에 P와 Q에 없던 새로운 논리적인 결함이 발생할 수도 있다. 또한 매우 드문 경우지만 P와 Q를 결합 과정에서 P 또는 Q의 논리적 결함이 해결될 수도 있다. 예를 들면, P가 입력 변수를 3만큼 증가 시켜 출력(리턴)하는 모듈인데 2만큼만 증가시키는 결함이 있고, Q는 입력 변수를 1만큼 증가 시켜 출력하는 모듈인데 2만큼 증가시키는 결함이 있을 때, P와 Q의 결합인 P;Q는 입력 변수를 3과 1의 합인 4만큼 증가시킨다고 기대되며 실제로는 2와 2를 합하여 정확한 값인 4를 출력한다. 이 예는 P와 Q에 각각 결함이 1개씩 존재하지만, P;Q에는 결함이 없어진 예이다.

이와 같은 분석의 결과로 모듈 심각도는 조건 6)과 조건 9)는 만족하지만 단조성(monotonicity)을 나타내는 조건 5)는 만족하지 않는다. 왜냐하면 두 모듈의 결합이 논리적 결함을 늘릴 수도 있지만 줄일 수도 있기 때문이다. 단, 문법적 결함만을 고려한다면 조건 5)를 만족한다.

표 2. Weyuker 기준에 대한 모듈 심각도 만족 결과
Table 2. Conformance of metrics to Weyuker's properties

조건	1	2	3	4	5	6	7	8	9
만족여부	○	×	○	○	×	○	○	○	○

결과적으로 모듈 심각도 메트릭은 조건 2와 조건 5를 제외한 기준들은 모두 만족하였다. 만족시키지 못한 조건 2는 오늘날 가장 많이 사용되는 복잡도 메트릭들 중 하나인 Cyclomatic Complexity도 만족하지 못하는데, [10]은 이로 인하여 메트릭의 타당성이 문제되는 것은 아니라고 주장하였다.

2. 실험적 검증

NASA 데이터 집합은 NASA에서 제공한 원본과 이들 중 일부를 정제하여 PROMISE에 넣은 두 가지 버전이 존재한다. 후자는 심각도 정보를 갖고 있지 않으므로, 본 연구에서는 원본 데이터를 사용한다. NASA 데이터 집합을 구성하는 13개 프로젝트들 중에서 심각도 값이 없거나 모두 0인 경우를 배제하고 얻어진 프로젝트들은 PC4, KC1, KC3, KC4, JM1이다. 이들 중 KC4는 다른 프로젝트들에 비해 결함의 심각도, 우선도를 제외한 다른 속성 메트릭 값들이 많이 부족하였으므로 제외하였고, 나머지 프로젝트들에 대해 모호성 분석을 실시하였다. [14]는 NASA 데이터 집합에 심각한 모호성이 존재함을 밝혀내었다. 이와 같은 분석 결과는 각 프로젝트의 결함 정보 파일들인 product_module_metrics와 defect_product_relations가 갖고 있는 결함 관련 정보가 서로 틀리다는데 기인한다. [14]의 분석 실험 결과 JM1과

4) P와 Q가 같은 기능을 갖는다. P와 Q는 같은 입력들에 의해 멈추고, 입력 각각에 대해 같은 출력을 갖는다.
5) P와 Q를 문법적으로 합한(순서를 생각하여 연결한) 프로그램

PC4가 가장 모호성이 적은 프로젝트였으므로, 실험적 검증을 위한 프로젝트로 JM1과 PC4를 사용한다. 모듈 심각도 값들을 구하기 위한 결함 정보와 심각도 정보들은 product_module_metrics와 defect_product_relations 파일들을 합성하여 구했다. 모듈 심각도 메트릭들 중 가장 의미 있는 것은 ms이며, 프로젝트들의 심각도를 전체적으로 비교하는 것은 msd를 사용하였다.

2.1 JM1

JM1은 C 언어로 구현된 실시간 시스템 프로젝트이다. 시스템은 315 KLOC, 10,878 모듈로 구성되었으며 그 중에서 결함을 가지는 모듈은 2,102개가 있다. 결함은 3,161개 존재하며 결함 심각도 레벨 값은 1~5 범위의 값을 갖는다.

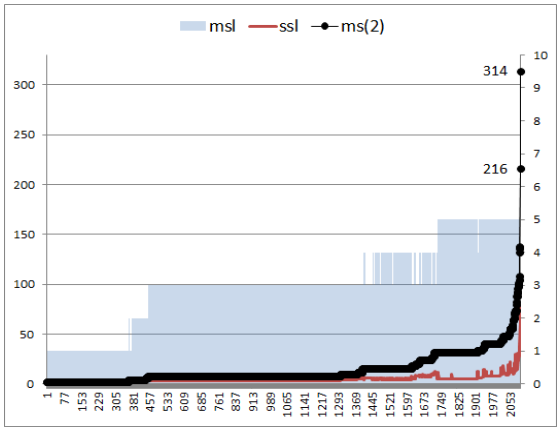

그림 4. JM1의 모듈 심각도 결과 그래프
Fig. 4. Module Severity graph for JM1

그림 4는 JM1의 모듈 심각도를 ms(2)를 기준으로 정렬한 그래프이다. 그래프의 판독을 위해 결함이 없는 모듈들은 제외하였으며, msl은 오른쪽 축을 사용하는 막대 그래프로 ms(2)와 ssl은 왼쪽 축을 사용하는 꺾은선 그래프로 나타내었다. msl 그래프에 빈틈이 보이는 걸로 보아 반드시 ms(2)가 클수록 msl이 커지지는 않지만 대부분 모듈에서 이러한 성질은 지켜지는 것을 볼 수 있다. JM1은 가장 심각한 결함들이 소수인 일반적인 시스템과 다르게 가장 심각한 결함의 보유를 의미하는 msl이 5인 모듈들이 결함 보유 모듈의 16% 이상인 343개나 되므로 사실상 msl로 고심각 모듈들의 차이를 평가하는 것은 불가능하다. 하지만 ms(2)는 314인 것 1개, 216인 것 1개, 약 130 정도가 되는 것들 3개로 나열되므로 더 심각한 모듈들을 정확히 선정할 수 있다.

2.2 PC4

PC4는 C 언어로 구현된 지구 궤도 위성 비행 시스템 프로젝트이다. 시스템은 36 KLOC, 1,458 모듈로 구성되었으며 그 중에서 결함을 가지는 모듈은 178개가 있다. 결함은 370개가 존재하며 결함 심각도 레벨 값은 1~5가 아니라 1~3 범위의 값을 갖는다. 이는 defectlevel 값이므로 dsl은 3~5 범위의 값을 갖게 되며, msl 역시 3~5 범위의 값을 갖는다. 그림 5는 그림 4와 같은 기준으로 PC4를 표현한 그래프이다. msl이 5인 고심각 모듈들이 58개로 결함 모듈의 32%가 넘으므로 이들을 심각도 기준으로 변별하는 것은 불가능하지만 ms(2)를 사용하면 440인 것 1개, 약 230인 것 2개, 152인 것 1개, 90대인 것 3개 등으로 명확히 심각도 순으로 정렬 가능하다.

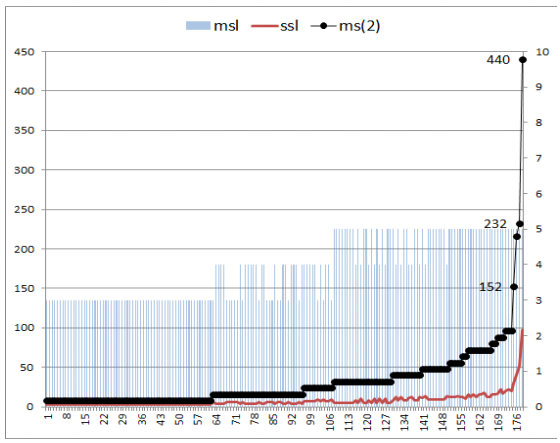

그림 5. PC4의 모듈 심각도 결과 그래프
Fig. 5. Module Severity graph for PC4

V. 메트릭 응용

제안 메트릭들 중 모듈 심각도를 나타내는 대표적인 메트릭은 ms이다. 그림 4와 그림 5는 ms를 이용하여 각 모듈들을 심각도에 따라 비교할 수 있다는 것을 보여준다. 모듈 심각도 메트릭을 이용할 수 있는 또 다른 중요한 응용 분야는 서로 다른 소프트웨어 시스템 간의 비교이다. 전체적으로 심각도가 높은 시스템은 재공학 등의 기법을 써서 심각도를 낮추는 작업을 하여야한다. 두 시스템을 비교하기 위해서 가장 적당한 메트릭은 모듈이나 시스템의 사이즈에 영향을 받지 않는 msd이다.

표 3. JM1과 PC4의 msd 분포
Table 3. msd for JM1 and PC4

프로젝트	msd 평균	msd 표준편차
JM1	8.809	38.202
PC4	14.162	58.969

표 3은 JM1과 PC4의 msd 통계치들을 비교한 것이다. 결과는 JM1이 PC4보다 덜 심각한 시스템이라는 결과를 보인다. 하지만 표준편차가 매우 크므로 두 프로젝트의 정확한 비교를 위해서는 msd 그래프 비교가 필요하다. 두 프로젝트의 모듈 수가 10,878과 1,458로 틀리므로 두 프로젝트의 msd 값들을 오름차순으로 정렬한 후 JM1 데이터를 샘플링하여 1,458개로 만든 후 비교하였으나 msd가 0인 부분이 너무 많아서 그래프의 비교가 불가능하였다. 따라서 결함을 가지고 있는 모듈들만을 같은 방법으로 비교하였다. JM1의 2,102개의 결함 보유 모듈들을 msd 기준으로 정렬한 후 처음부터 12번째 모듈들을 선택하는 방법으로 175개를 선택하였다. PC4의 가장 큰 msd값은 1066으로 PC4의 분포와 두 번째로 큰 값이 640인 점을 고려해볼 때 이상점으로 판단되어 비교에서 제외하였다. JM1의 msd가 가장 큰 부분의 24개 모듈에서 간격을 6으로 하여 4개의 모듈들을 선택함으로써 175개에 2개를 더한 177개의 모듈 샘플을 만들었다. 결과적으로 JM1은 샘플링된 175개에 msd가 큰 부분에서 2개를 더 샘플링해서 177개의 샘플을 만들었고 PC4는 178개에서 이상점 1개를 제외하여 177개를 만들었다. 그림 6은 그 결과를 나타낸 것으로 결함 모듈들을 비교하면 PC4가 JM1보다 거의 모든 부분에서 msd가 크다. 이는 PC4의 dsl이 JM1보다 큰 3부터 시작하는 영향도 있지만 msd가 매우 큰 고심각 결함의 고밀집 모듈도 PC4가 더 크다는 것은 PC4가 JM1에 비해서 심각도가 큰 프로젝트라는 것을 의미한다.

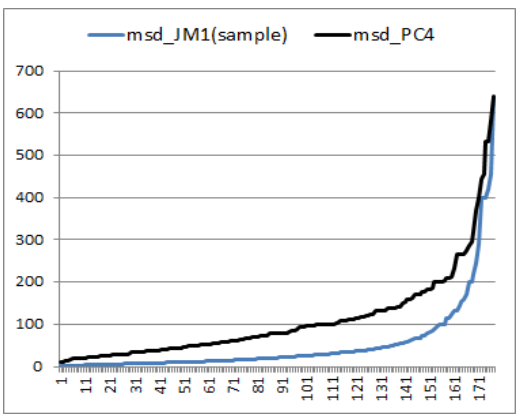

그림 6. JM1과 PC4의 심각도에 기반한 비교 결과
Fig. 6. Severity comparison between JM1 and PC4

Ⅵ. 결 론

모듈의 결함 보유 여부만으로 결함에 관련된 모듈의 성질을 나타내기에는 너무 부족하다. 서로 다른 모듈들의 결함 관련 성질은 모듈이 가지고 있는 결함의 수와 결함의 심각도에 따라 매우 큰 차이가 있다. 결함 심각도의 중요성을 언급한 기존 연구들은 결함의 심각성에 따른 심각도 레벨은 정의하고 있지만 모듈의 심각도를 정량화하려는 시도는 거의 없었다. 소프트웨어 품질 관련 작업은 주로 모듈 또는 시스템 단위로 이루어지므로 모듈의 심각도를 정확히 측정하는 것은 적절한 자원 할당을 통한 성공적인 소프트웨어 프로젝트 관리에 매우 중요한 역할을 한다. 따라서 본 논문에서는 모듈 내부의 결함 수와 결함의 심각도를 고려한 모듈 심각도 메트릭 집합을 정의하였다.

제안 메트릭들의 유용성을 검증하기 위해 Weyuker의 기준들을 이용한 분석적 검증과 NASA 데이터 집합을 사용한 실험적 검증을 수행하였다. 분석적 검증 결과, 메트릭의 타당성에 큰 영향을 미치지 않는 것으로 알려진 두 조건들 외의 모든 조건들을 만족시킴으로써 제안 메트릭 집합이 이론적으로 타당한 형태라는 것을 보였다. JM1과 PC4 프로젝트를 이용한 실험적 검증 결과, 제안 메트릭들 중 ms는 모듈의 심각도를 수치화함으로써 단순한 심각도 레벨 정보가 갖는 변별력 문제를 해결하였고, msd는 서로 다른 시스템 간에 모듈 심각도에 기반한 비교를 가능케 하였다. 따라서 본 논문의 제안 메트릭들은 심각도에 기반한 모듈 간의 비교뿐만 아니라 소프트웨어 시스템 간의 비교에도 매우 유용하게 사용될 수 있다.

참고문헌

[1] IEEE Standard Classification for Software Anomalies, IEEE Std. 1044-2009.

[2] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post release software evolution process," Journal of Systems and Software, Vol.81, No.11, pp. 1868-1882, 2008.

[3] C. Catal, "Software fault prediction:A literature review and current trends," Expert Systems

with Applications, Vol.38, No.4, pp.4626-4636, 2011.

[4] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," IEEE Trans. Software Eng., Vol.38, No.6, pp. 1276-1304, 2012.

[5] D. E. Harter, C. F. Kemerer, and S. A. Slaughter, "Does Software Process Improvement Reduce the Severity of Defects? A Longitudinal Field Study," IEEE Trans. Software Eng., Vol.38, No.4, pp. 810-827, 2012.

[6] T. Menzies and A. Marcus, "Automated Severity Assessment of Software Defect Reports," Proc. of ICSM 2008, pp.346-355.

[7] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," IEEE Trans. Software Eng, Vol.32, No.10, pp.771-789, 2006.

[8] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Reflections on the NASA MDP data sets," IET Software, Vol.6, No.6, pp. 549-558, 2012.

[9] N. Fenton, "Software Measurement: A Necessary Scientific Basis," IEEE Trans. Software Eng., Vol.20, No.3, pp.199-206, 1994.

[10] H. Zuse, Software Complexity Measures and Methods, Walter de Gruyter, 1991.

[11] E. J. Weyuker, "Evaluating Software Complexity Measures," IEEE Trans. Software Eng., Vol.14, No.9, pp.1357-1365, 1988.

[12] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610.12

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

[14] E. Hong, "Ambiguity Analysis of Defectiveness in NASA MDP data sets," Journal of the Korea Society of IT Services, Vol.12, No.2, pp.361-371, 2013.

저 자 소 개

홍 의 석

1992: 서울대학교
계산통계학과 전산과학전공 학사.
1994: 서울대학교
계산통계학과 전산과학전공 석사.
1999: 서울대학교
계산통계학과 전산과학전공 박사
현 재: 성신여자대학교 IT학부 교수
관심분야: 소프트웨어 품질 예측,
소프트웨어 메트릭, MSR 등
Email : hes@sungshin.ac.kr