韓國IT서비스學會誌
第12卷 第2號
2013年 6月, pp.361-371
http://dx.doi.org/10.9716/KITS.2013.12.2.361

NASA MDP 데이터 집합의 결함도 모호성 분석*

홍 의 석**

Ambiguity Analysis of Defectiveness in NASA MDP Data Sets*

Euyseok Hong**

■ Abstract ■

Public domain defect data sets, such as NASA data sets which are available from the NASA MDP and PROMISE repositories, make it possible to compare the results of different defect prediction models by using the same data sets. This means that repeatable and general prediction models can be built. However, some recent studies have raised questions about the quality of two versions of NASA data set, and made new cleaned data sets by applying their data cleaning processes. We find that there are two ways in the NASA MDP versions to determine the defectiveness of a module, 0 or 1, and the two results are different in some cases. This serious problem, to our knowledge, has not been addressed in previous studies. To handle this ambiguity problem, we define two kinds of module defectiveness and two conditions that can be used to determine the ambiguous cases. We meticulously analyze 5 projects among the 13 NASA projects by using our ambiguity analysis method. The results show that JM1 and PC4 are the best projects with few ambiguous cases.

Keyword : Public Data Sets, NASA MDP Data Sets, Defectiveness, Ambiguity Analysis

논문투고일 : 2013년 01월 27일 논문수정완료일 : 2013년 05월 29일 논문게재확정일 : 2013년 06월 06일

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2010-0021902).

** 성신여자대학교 IT학부 교수

1. 서 론

소프트웨어 품질 예측은 설계 산물을 사용하여 개발 산물인 소프트웨어 코드의 품질 인자들을 예측하거나, 초기 코드를 사용하여 테스팅 단계의 품질 인자들을 예측하는 작업이다. 기존의 연구들을 보면 예측하려는 품질 인자는 대부분 소프트웨어 결함(software defect)에 관련된 것들이었으며 정확한 결함 수 또는 결함 경향이 많은 모듈 순서를 예측하는 것보다는 결함 발생 여부를 예측하는 것들이 대부분이었다. 이와 같은 연구 경향은 모듈 결함경향 순서 모델이나 정확한 결함수를 예측하는 모델이 더 유용하기는 하지만[7], 정확한 모델을 제작하기 어렵다는 점과 결함경향성 판단 만으로도 충분히 개발 프로세스에 유용한 정보로 활용할 수 있다는 점에 기인한다. 따라서 본 연구에서 언급하는 결함 예측 모델은 메트릭 벡터들로 입력 모듈들을 정량화한 후 이들을 결함 모듈과 비결함 모듈로 분류하는 분류 모델(classification model)을 의미한다. 결함 예측 모델은 구현시 결함이 많이 생길 부분들을 미리 찾아 적절한 자원 할당을 가능케 함으로써 소프트웨어 개발 전체 비용을 낮추고 구현된 소프트웨어의 품질을 높여준다[6].

예측 모델들은 크게 훈련 데이터를 사용하여 학습한 감독형 모델과 훈련 데이터가 없는 경우 사용할 수 있는 비감독형 모델로 나뉜다. 기존에 제안된 모델들의 대부분은 감독형 모델들이며, 사용된 훈련 알고리즘들은 판별분석, 선형 회귀분석, 로지스틱 회귀분석 등과 같은 통계 기법이나 분류트리, 랜덤포리스트, 역전파 신경망, CBR, SVM 등과 같은 인공지능 기법들이다[2, 3, 6]. 비감독형 모델은 데이터의 분포를 이용하는 클러스터링 알고리즘들이 주로 사용되고 예측 성능도 떨어지기 때문에 매우 극소수의 연구들이 존재하며[1, 13], 충분치 않은 훈련 데이터가 존재할 때 사용할 수 있는 세미감독형 모델들도 제안되었다[10].

결함 예측 모델들에 대한 필요성은 널리 인식되었지만 2000년대 초까지는 연구들이 많이 진행되지 않았다. 하지만 2000년대 중반으로 오면서 매년 10개에 가까운 연구가 발표되었고 2005년 이후에는 그 수가 급격히 늘어나고 있다[3, 6]. 이 같은 현상의 주된 이유는 공개 데이터 집합의 등장 때문이다. 2000년대 초에 NASA IV&V 메트릭스 데이터 프로그램(MDP)[1]의 데이터가 공개되고 2005년에 소프트웨어 예측 모델에 관련된 공개 데이터 집합들을 관리하는 PROMISE 레포지토리[2] 운영이 시작됨으로써 이들을 사용한 연구들이 매우 많아진 것이다. 가장 많이 사용된 공개 데이터 집합은 NASA MDP 데이터 집합으로, 원본 데이터와 이들 중 일부를 정제하여 PROMISE에 넣은 두가지 버전이 존재한다. Hall et al.[6]이 분석한 2000년부터 2010년까지 발표된 결함 예측 관련 논문들은 208편이었으며, 그 중 62편이 NASA 데이터를 사용한 연구였다. 비공개 데이터를 사용한 많은 연구들의 문제점은 이 데이터를 다른 연구들에서 재사용하기 어렵기 때문에 다른 연구의 결과들과 비교 및 개선작업이 어렵다는 것이다[2]. 수많은 모델들 중 어떤 것을 택하느냐도 어려운 문제이기 때문에 여러 모델들의 선택부터 결함 예측, 평가를 지원하는 결함 예측 프레임워크에 대한 연구들도 등장하였다[8, 12].

많은 연구들이 모델의 알고리즘에 치중하였지만 결과에 매우 큰 영향을 끼치는 것은 데이터를 어떻게 전처리하여 정제하는가이다[8, 11]. 따라서 대표적인 공개 데이터인 NASA 데이터도 어떻게 전처리 하냐에 따라 모델의 결과는 달라진다. NASA 데이터의 두 버전도 속성과 인스턴스들의 수가 일치하지 않으며 인스턴스 순서도 일치하지 않는다. 이는 NASA 데이터를 사용한 연구들도 결과에 대한 비교가 어렵다는 것을 의미한다[11]. 따라서 최근에는 NASA 데이터 집합의 문제점들을 지적하고 정확한 정제 방법을 제공하는 연구들이 등장하였으며[5, 11], 이들은 모델의 입력에 해당되는 여

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

러 메트릭 속성들을 정제하는데 초점을 맞추었다.

NASA 데이터를 사용하여 이진 분류 결함 예측 모델을 제안한 연구들은 일반적으로 모델 출력에 해당하는 결함 여부값(*defective*)을 다음과 같이 error_count 속성을 사용하여 정의하였다[5]. 이들 중 명시적으로 (1)과 같은 수식을 제시한 연구들도 다수 존재한다[4, 9, 8].

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

본 연구의 중요한 목적들 중 하나는 NASA 데이터 집합을 이용하여 모듈의 결함 여부를 결정하는 방법에 모호성이 존재하는 것을 밝히는 것이며, 그 결과로 NASA 데이터 집합에서 식 (1)과 다른 방법이 있음을 알아내었고, 그 방법의 결과값이 식 (1)의 결과와 일치하지 않는다는 문제점을 발견하였다. 이 결함도 정의에 대한 모호성은 데이터 집합 자체의 문제이므로 모호한 레코드들을 모두 삭제하거나 모호성이 가장 적은 데이터 집합들을 모델 연구에 사용할 수 있다. 프로젝트 당 상당히 많은 레코드들이 모호성이 존재하였으므로 본 연구는 후자의 방법을 택하여 가장 모호성이 적은 프로젝트들을 실험을 통해 제시한다. 이를 위해 NASA 데이터 집합의 13개 프로젝트들[3] 중에서 결함 관련 속성들이 명확하게 정의된 5개의 프로젝트들을 선정하였으며, 두 가지 형태의 결함도 해석을 이용하여 모호성 분석 방법을 정의하였다. 제안한 모호성 분석 방법을 이용해 5개의 프로젝트들에 대해 모호성 분석 실험을 행하여 모호성이 가장 낮은 프로젝트 두 개를 선정하였다.

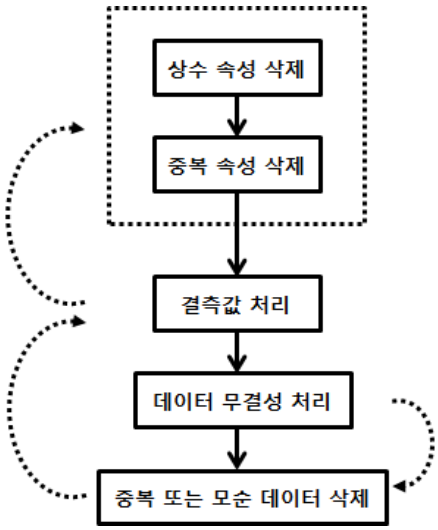
제 2장에서는 NASA 데이터 집합에 대한 최근 데이터 정제 연구들과 결함 속성에 관한 정제 연구들을 살펴보고 제 3장에서는 NASA 데이터 집합의 결함도 관련 속성들에 모호성이 존재함을 실례를 통해 밝히고 이를 분석할 수 있는 모호성 분석 방법을 정의한다. 제 4장에서는 NASA 데이터 집합에 행한 모호성 분석 실험과 그 결과로 얻은 가장 모호성이 적은 프로젝트들에 대해 설명하고 제 5장에는 결론에 대해 기술한다.

3) 13개 프로젝트들은 <표 1>에 나타내었다.

2. 관련 연구

2.1 NASA 데이터 정제에 관한 연구

데이터 정제(data cleaning) 작업이란 데이터를 사용하기 전에 데이터에 존재하는 여러 문제점들을 찾아 해결하여 문법적으로나 의미적으로 완전한 데이터 집합을 만드는 과정을 의미한다.



[그림 1] NASA 데이터 정제 프로세스들

품질 예측 모델을 제안하고 제안 모델에 대한 성능 실험을 하기 위하여 모델 구축 전에 행해진 간단한 데이터 정제 작업이 아니라 NASA 데이터의 문제점을 지적하고 이를 해결하기 위한 표준화된 정제 기법을 제안한 연구는 많지 않다. 대표적인 두 연구는 [11]과 [5]이며, 두 연구의 결과는 유사한 점이 많다. 두 연구의 가장 큰 차이점은 제안한 정제 작업의 순서가 다르다는 것이며 이를 [그림 1]에 나타내었다. 그림은 두 연구의 데이터 정제 프로세스를 개략적으로 나타낸 것으로 실선은 [5], 점선은

[11]의 정제 작업 순서를 나타낸다. [그림 1]의 각 정제 단계 작업은 [5]의 용어를 사용하였다.

다음은 그림에서 나타낸 각 단계들의 간략한 설명이다.

단계 1) 상수 속성 삭제(removal of constant attributes) : 속성들 중 모두 같은 값을 갖는 속성들은 새로운 지식을 제공하지 않으므로 삭제한다.

단계 2) 중복 속성 삭제(removal of repeated attributes) : 같은 의미를 갖는 두 개 이상의 속성들은 하나를 남기고 삭제한다. KC4의 number_of_lines와 loc_total은 중복 속성의 예이다.[4)]

단계 3) 결측값 처리(handling of missing values) : 결측값은 해당 케이스를 삭제할 수도 있고 적절한 값으로 채워 넣을 수도 있다. 일반적으로 error_count 등이 결측값이면 0으로 채워 넣는다.

단계 4) 데이터 무결성 처리(enforce data integrity) : 이 작업은 상충되는(conflicting) 속성값 처리와 타당하지 않은(implausible) 속성값 처리로 나눌 수 있다. 전자는 몇 가지 속성값들의 일정한 관계가 성립할 때 그 규칙을 지켜야한다는 것이다. 예를 들면 halstead_length 속성값은 num_operators 값과 num_operands 값의 합이 돼야 한다. 후자의 예로는 loc 값이 1.1이 될 수 없다는 것을 들 수 있다.

단계 5) 중복 또는 모순 데이터 처리(removal of repeated and inconsistent instances) : 중복 케이스는 입출력 값이 모두 같은 케이스이고 모순 케이스는 입력은 같고 출력이 다른 것이다. 이들은 같은 하나의 케이스만 남기고 삭제한다.

4) 본 논문에서 NASA 데이터 집합의 메트릭 표현은 소문자를 사용하였다. LOC_TOTAL, ERROR_COUNT는 loc_total, error_count로 나타내었다.

두 연구들 중 [11]은 데이터 정제 순서의 중요성을 강조하였다. 즉, 정제 순서를 명확하게 정해놓지 않으면 순서가 다른 정제 작업은 매우 다른 결과를 가져올 수 있다는 것이다. 정제 순서는 삭제하지 않으면 문제가 발생하는 심각한 데이터 처리(단계 4)부터 시작하고 하고, 문제를 발생시키지는 않지만 결함 예측에 도움이 되지 않는 데이터를 처리하는 순서로 할 것을 제안하였다. 이와 같은 순서로 작업해야 상충 데이터 해결에 도움이 되고, 정제 작업을 통해 적은 데이터 손실을 가져올 수 있다고 주장하였다. 즉, [그림 1]에서 [11]의 정제 순서는 점선을 따라 단계 4 → 단계 5 → 단계 3 → 단계 1, 2이다.

또 다른 연구인 [5]가 가장 강조한 것은 중복된 데이터가 모델의 훈련 데이터 집합(training data set)과 훈련된 모델을 검증하는 검증 데이터 집합(test data set)에 들어가게 되면 모델의 성능이 급격히 상승돼 왜곡된 결과가 나올 수 있다는 것이다. 이를 해결하기 위해 중복 케이스 삭제 외에도 중복된 케이스들에 서로 다른 속성값을 추가하는 기법을 제안하였다.

2.2 결함도 관련 속성 정제에 관한 연구

NASA 데이터 집합에서 결함도란 해당 모듈이 결함 모듈인지, 아닌지를 나타내는 것이다. 대부분의 결함 관련 예측 모델들은 입력 모듈이 결함 경향 모듈인지 아닌지를 판단하는 이진 분류 모델 형태이므로 모델 제작이나 검증에 사용되는 데이터 집합은 이 결함 여부를 나타내는 속성, 즉 결함도(defectiveness)를 가지고 있다. 결함도 속성의 값은 1(true)과 0(false)이 된다. 결함도 속성은 NASA 데이터 집합에서 제공되고 있지는 않으며 결함 여부에 관련된 속성들을 사용하여 연구자가 정의하여 데이터 집합에 넣어야 한다.

NASA 데이터 집합을 사용한 많은 연구들은 수식 (1)과 같이 모듈의 결함도가 1이란 것을 모듈이 에러를 하나 이상 가지고 있다는 것으로 해석했다

[5]. 모듈의 에러 수 값은 error_count 속성이 가지고 있다. error_count 외에도 결함 여부에 대한 속성들은 여러 파일에 걸쳐 존재한다. 1,000 loc당 에러 수를 나타내는 error_density, 결함을 나타내는 식별자인 defect_id, 결함의 심각도와 우선도를 나타내는 severity와 priority 등이 있다.

기존의 NASA 데이터 정제 연구들이나 결함 예측 연구들에서 모델의 출력인 결함도의 정의에 관련된 속성들의 무결성이나 모순 여부를 언급한 연구는 찾아볼 수 없다. 본 연구의 목적은 기존 연구들이 사용한 error_count 속성 값에 기반한 결함도 정의에 문제가 있음을 밝히고 이를 해결하는 것이다.

3. 모호성 분석 방법

3.1 데이터 집합 선정

결함 예측 모델 연구에 사용되는 NASA 데이터 집합은 두 가지 버전이 존재하며, 이는 NASA MDP 원본 데이터 집합과 PROMISE의 NASA 데이터 집합이다. PROMISE 데이터 집합은 이미 정제 과정을 거친 데이터로 한 프로젝트가 여러 개의 테이블(파일)로 구성된 것이 아니라 하나의 테이블로 구성되어있다. 여기에는 defect_id, 결함 심각도(severity), 결함 우선도(priority)와 같은 모듈의 여러 결함 정보들과 각 모듈이 가지고 있는 결함들을 나타내는 모듈-결함간 정보들이 생략되어있으므로 이들의 분석을 목적으로 하는 본 연구에는 적합하지 않다. 따라서 본 연구는 NASA 원본 데이터 집합을 사용하였다.

NASA MDP 데이터 집합은 13개의 프로젝트들로 이루어져있다. 각 프로젝트의 도메인이나 구현 언어, 크기, 참여 인력 등은 매우 다르므로 결함도에 관련된 속성들이 부족하거나 속성이 존재한다 하더라도 값이 존재하지 않는 프로젝트들이 존재한다. 따라서 이 속성들이 명확하게 나타나있는 프로젝트들을 연구 대상으로 축약하였다. 데이터 집합 선정시 모듈 관련 속성보다는 결함 관련 속성들에 초점을 맞추었다. 그 이유는 모듈 결함도에 대한 두 가지 정의들 중 하나는 기존 연구들이 사용하는 모듈 관련 속성인 error_count를 사용하지만, 본 연구에서 새롭게 정의하는 방법은 결함 관련 속성과 모듈의 연관 관계를 이용하기 때문이다. 따라서 프로젝트들 중 결함 부분이 명확하게 잘 정의된 프로젝트들을 선정하여 실험할 필요가 있다. NASA 데이터 집합에서 결함의 성격을 규정하는 속성들은 심각도, 우선도, close_reason, Fix_Hour 등 여러 개가 있다. 하지만 심각도, 우선도를 제외하고는 프로젝트별로 속성의 이름이 다르거나 속성이 존재하여도 결측값들이 너무 많다는 문제가 있다. 따라서 결함이 명확하게 정의된 프로젝트를 선정하는 기준으로 심각도와 우선도 속성값들의 유무를 사용하였다.

결함 심각도는 해당 결함이 전체 시스템에 미치는 충격의 정도를 의미하며, 결함 우선도는 결함을 빨리 처리해야하는 우선 정도를 의미한다. 결함 심각도 및 우선도는 여러 회사나 개발 집단에서 많이 사용하는 용어이지만 값의 범위와 각 값의 의미에 대한 일반적인 표준 정의는 없다. 주로 사용되는 심각도, 우선도는 1~3값 또는 1~5값을 가지며 1이 가장 심각한 또는 우선적인 값이다. NASA 데이터는 심각도 값으로 1~5, 우선도 값으로 1~3을 갖는다.

본 연구에서는 심각도와 우선도 메트릭 값이 명확히 나타난 프로젝트들을 결함도 관련 속성들이 명확한 프로젝트로 규정하였다. <표 1>은 13개 프로젝트들이 가지고 있는 심각도, 우선도 정보들을 분석한 것이다.

두 속성값이 없거나 모두 0인 경우, 한 가지만 있는 경우를 배제하였다. 그 결과 얻어진 연구 대상 프로젝트들은 PC4, KC1, KC3, KC4, JM1이다. KC1, KC3, KC4같이 우선도 값이 부족한 경우는 대상으로 선정하였으며, 이들 중 KC4는 연구 대상으로 선정하였지만 다른 프로젝트들에 비해 심각도, 우선도를 제외한 다른 속성 메트릭 값들이 많이 부족하였다.

〈표 1〉 심각도, 우선도 정보 분석

프로젝트	심각도, 우선도 정보
CM1	priority만 있음
PC1	priority만 있음
PC2	severity, priority 모두 0
PC3	severity, priority 모두 0
PC4	severity, priority 값 있음
PC5	없음
KC1	severity, priority 값 있음 priority 값 부족
KC3	severity, priority 값 있음 priority 값 3개 없음
KC4	severity, priority 값 있음 priority 값 부족
MC1	priority만 있음
MC2	없음
JM1	severity, priority 값 있음
MW1	severity, priority 모두 0

3.2 결함도 정의의 모호성

앞에 언급하였다시피 NASA 데이터 집합에 속하는 프로젝트들의 성격은 각각 다르다. 따라서 예측 모델의 품질 예측 단위가 객체지향언어를 사용한 프로젝트에서는 클래스나 메소드가 될 수도 있고 일반 언어를 사용한 시스템에서는 함수나 모듈, 서브루틴이 될 수 있다. NASA 문서에서는 가장 하위의 기능 단위들(함수, 모듈, 서브루틴, 메소드)을 총칭해서 모듈(module)이라 정의 하였으며 대부분의 기존 모델 연구들은 이를 품질 예측 단위로 사용하였기에 본 연구의 기본 단위도 모듈로 하였다.

모듈의 결함 여부에 관련된 속성들은 몇 개의 파일에 분산되어있다. NASA 데이터 집합을 분석해보면 연구 대상으로 선정한 5개 프로젝트 뿐만 아니라 13개 프로젝트 모두가 구성원인 파일들의 형태와 이름이 매우 유사하다. 프로젝트를 구성하는 파일들 중 결함 정보를 갖는 파일들은 다음 세 개의 파일이다. *projectname*은 각 프로젝트의 이름을 의미한다.

- *projectname*_product_module_metrics(PMM)
- *projectname*_defect_product_relations(DPR)
- *projectname*_static_defect_data(SDD)

PMM은 모듈에 관련된 여러 속성들로 구성되어 있으며 결함 예측 모델의 입력들은 대부분 이들 중에서 수집된다. 속성들로는 모듈을 구분하는 식별자인 module_id, loc_total 등으로 구성된 loc관련 메트릭들, cyclomeatic_complexity로 대표되는 McCabe 기반 메트릭들, halstead_length, halstead_content 등과 같은 Halstead 기반 메트릭들, 모듈의 결함 여부와 관련된 error_count, error_density 등이 있다. DPR은 모듈 각각에 연관된 결함들 정보들이 들어있으며 module_id, defect_id, severity, priority로 구성 되어있다. SDD에는 결함이 발생하고 그를 해결할 때까지 변하지 않는 정적인 결함 정보들이 들어있으며 defect_id, severity, priority 속성과 함께 결함을 해결하는데 든 시간을 나타내는 fix_hours, 결함 해결을 위해 변경되거나 추가된 loc를 나타내는 sloc_count 등이 있다.

이들 세 개의 파일의 속성들로부터 한 모듈의 결함 여부를 정의해보면 두 가지 방법이 가능하다. 대부분의 기존 연구들이 사용한 첫 번째 방법은 PMM에 있는 모듈의 error_count 속성을 이용하는 것이다. NASA 데이터 집합을 설명한 NASA 문서에는 error_count를 “the number of defects associated with a module”이라 정의하였다. 따라서 모듈의 error_count는 그 모듈에서 발생한 결함의 수이다. 따라서 모듈은 error_count가 1보다 크거나 같으면 결함 모듈이고 0이면 결함 모듈이 아니라고 정의할 수 있다. 다음 식 (2)는 이를 표현한 것이다. (m, error_count)은 PMM에 존재하는 모듈 m의 케이스의 부분 케이스이다.

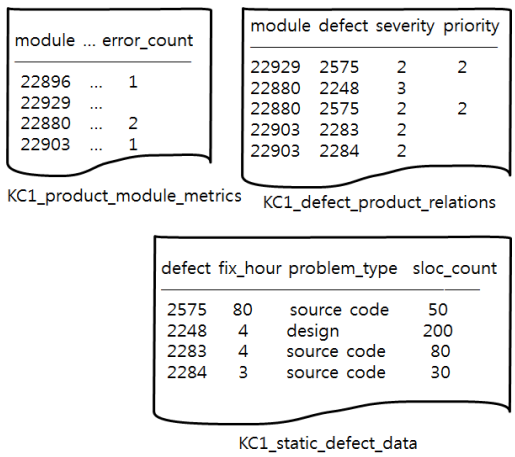
$$defective1(m) = \begin{cases} 1, if\ \text{error\_count} \geq 1 \ _{s.t.\,(m,\text{error\_count})\in \text{PMM}} \\ 0, otherwise \end{cases} \quad (2)$$

모듈의 결함도에 대한 또 다른 정의 방법은 DPR 파일 정보를 이용하는 것이다. 모듈에 결함이 존재하면 DPR 파일에 해당 모듈과 결함의 쌍이 케이스로 존재한다. 따라서 모듈 m의 결함수는 DPR에

나타난 m의 케이스 수와 같다. 즉, 모듈은 DPR에서 자신에 연관된 defect_id가 하나 이상 존재하면 결함 모듈이고 존재하지 않으면 결함 모듈이 아니라 정의할 수 있다. 식 (3)은 이를 표현한 것이다. (m, defect_id)는 DPR에 존재하는 모듈 m과 결함 defect_id로 구성된 케이스의 부분 케이스이다.

$$defective2(m) = \begin{cases} 1, if\ \text{defect\_id}\ exists\ _{s.t.(m,\text{defect\_id})\in \text{DPR}} \\ 0, otherwise \end{cases} \quad (3)$$

앞에서 기술한 것과 같이 많은 기존 연구들이 식 (2)를 사용하여 결함도를 정의하였다. 이는 모듈의 결함도 예측에 관한 정보들이 모두 PMM에 있으므로 DPR 정보를 분석할 필요가 없었기 때문이다. 하지만 극소수의 결함 심각도에 기반한 예측 모델 연구에서는 식 (3)을 사용하였다[14]. 왜냐하면 모듈의 심각도 정보는 PMM과 DPR을 합성하여야 얻어질 수 있기 때문이다. 식 (2)와 (3)은 같은 개념에 대한 다른 접근들이기 때문에 당연히 같은 결과를 보여야한다. 하지만 NASA 데이터를 분석한 결과, 같지 않은 경우가 존재하였다. 즉, 모듈의 결함도에 대해 NASA 데이터 집합은 모호성을 가지고 있다.



[그림 2] 모듈의 결함도 모호성의 예

[그림 2]는 KC1 프로젝트 실제 데이터를 그림으로 옮긴 것으로 이 모호성의 실례를 보여준다. 모듈 22896은 PMM에서 error_count는 1이지만 DPR에서는 케이스가 없다. 즉, *defective1*(22896)은 1이지만 *defective2*(22896)은 0으로 같지 않다. PMM에서 22929는 error_count가 없지만 DPR에서는 22929는 결함 2575를 갖고 있다. SDD를 보면 결함 2575는 80 시간을 들여 50 라인이나 수정한 명백히 존재하는 소스 코드 결함이다. KC1의 PMM에서 error_count는 1부터 존재하므로 결측값은 0으로 해석된다. 따라서 *defective1*(22929)은 0이지만 *defective2*(22929)은 1이므로 같지 않다. 모듈 22880의 경우는 DPR에서도 2개의 결함이 존재하므로 PMM의 error_count가 2라는 결과와 일치한다. 하지만 모듈 22903은 error_count는 1이지만 DPR의 결함 정보는 2개 존재한다.

3.3 모호성 분석 기준

모듈 m이 결함도 정의에 대해 모호성이 존재하지 않으려면 *defective1*(m)와 *defective2*(m)이 같아야 한다. 따라서 다음과 같은 경우에 모듈 m은 결함도에 대한 모호성이 존재하게 된다.

$$defective1(m) \neq defective2(m) \quad (4)$$

이 조건을 풀어서 기술하면 다음과 같은 두 가지 경우가 된다. 여기서 식 (2), (3)의 결과인 1과 0은 true와 false로 해석한다.

$$\begin{aligned} &\neg defective1(m) \wedge defective2(m) \\ &defective1(m) \wedge \neg defective2(m) \end{aligned} \quad (5)$$

식 (5)에서 앞에 있는 조건은 PMM에서 m의 error_count는 0으로 결함이 없는 것 같지만 DPR에서는 m의 defect_id가 존재하는 경우이며, 뒤에 있는 조건은 PMM에서 error_count는 1 이상으로 결함이 있는 것 같지만 DPR에서는 m에 대한 de-

fect_id가 존재하지 않는 경우이다.

4. 모호성 분석 실험

4.1 실험

제 3장에서 정의한 모호성 분석 기준을 사용하여 5개의 연구 대상 프로젝트들을 분석하였다. 모호성 분석의 목적은 여러 다른 특성을 가진 프로젝트들을 결함도 모호성 기준으로 평가하여 가장 모호성이 적은 프로젝트들을 선정하는 것이다. 모호성이 많은 프로젝트 데이터를 사용한 결함도 관련 예측 모델들의 실험 결과는 신뢰성이 떨어지므로 모호성 분석을 통과한 프로젝트를 사용하는 것은 모델 성능 평가의 전처리 과정에서 매우 중요한 부분을 차지한다.

<표 2>는 5개 프로젝트에 대한 특성과 결함도 모호성 관련 분석 결과를 나타낸다. 'err_cnt > 0'은 각 프로젝트의 PMM 파일에서 error_count > 0을 만족하는 모듈수를 뜻한다. 'defect_yes'는 DPR 파일에서 defect를 가지는(defect_id가 존재하는) 모듈수를 뜻하며 'defect_no'는 그 반대의 의미이다. 'err_cnt > 0 and defect_no'는 error_count > 0인데 defect는 없는, 결함도 정의가 모호한 모듈수를 의미하며 모호성을 판단하는 기준인 식 (5)의 후자 조건을 만족하는 모듈의 수를 의미한다. 또 다른 모호한 모듈들을 나타내는 'err_cnt = 0 and defect_yes'는 error_count는 0인데 defect는 있는 모듈수를 나타내며 식 (5)의 전자 조건의 경우를 나타낸다. 이들 네 개 값의 관계를 수식으로 정리하면 다음과 같다.

$$'\mathrm{err\_cnt} > 0' - '\mathrm{err\_cnt} > 0 \text{ and } \mathrm{defect\_no}' = '\mathrm{defect\_yes}' - '\mathrm{err\_cnt} = 0 \text{ and } \mathrm{defect\_yes}' \quad (6)$$

식 (6)을 해석하면 결함 모듈의 수를 구하는 두 가지 방법에 대해, 각 방법이 포함한 모호한 모듈들의 수를 뺀 값은 같다는 의미이다. 'defect_num'은 해당 프로젝트를 구성하는 전체 모듈들이 가지는 defect 수이며 'err_cnt_sum'은 전체 모듈들의 error_count의 합이다. 이 두 값도 프로젝트의 전체 결함수를 나타내므로 같아야하지만 5개 프로젝트 모두 다르다. 'defect_percent'와 'err_percent'는 전체 모듈 중 이들이 차지하는 비율을 의미한다.

4.2 실험 결과 분석

5개 프로젝트에 대해 앞에서 정의한 식 (5)의 모호성 분석 기준을 적용하여 실험한 결과 NASA 데이터 집합은 매우 심각한 문제점이 있음을 발견

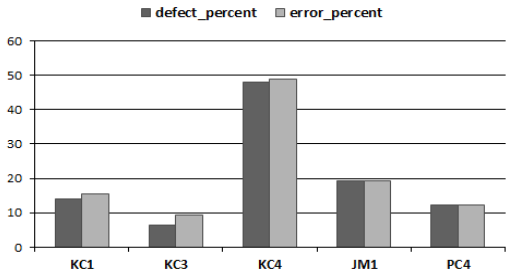
〈표 2〉 NASA 데이터 집합의 결함도 모호성 분석 결과

프로젝트 특성	KC1	KC3	KC4	JM1	PC4
구현 언어	C++	Java	Perl	C	C
모듈수	2,107	458	125	10,878	1,458
err_cnt > 0	325	43	61	2102	178
defect_yes	293	29	60	2102	178
err_cnt_sum	525	101	253	3179	367
defect_num	669	62	248	3161	370
err_percent	15.4	9.4	48.8	19.3	12.2
defect_percent	13.9	6.3	48	19.3	12.2
err_cnt > 0 and defect_no	39	15	4	0	0
err_cnt = 0 and defect_yes	7	1	3	0	0

하였다. 이는 같은 프로젝트 내에서 결함 정보를 가지고 있는 파일인 PMM과 DPR이 갖고 있는 결함도 관련 정보가 서로 틀리다는데 기인 한다.

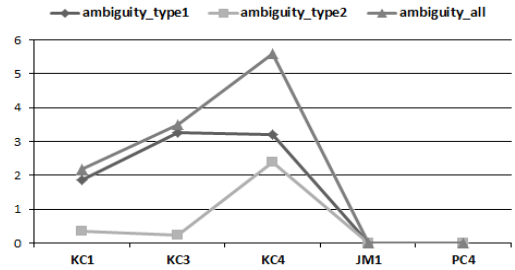
KC1은 2,107개의 모듈로 구성된 C++로 구현된 프로젝트이다. KC1_PMM을 분석한 결과는 error_count > 0인 모듈수는 325개이며, KC1에 존재하는 전체 에러수(즉, 결함수)는 525개이다. 하지만 KC1_DPR을 분석해 보면 결함을 가진 모듈수는 293개이며 전체 결함수는 669개이다. 결국 KC1 데이터는 error_count가 0보다 크지만 defect 정보가 없는 모듈이 39개 존재하고, defect 정보가 존재하지만 error_count가 0인 모듈이 7개 존재한다는 모순이 있다. 식 (6)을 이용하여 이 관계를 정리하면 325-39 = 293-7이 된다. 또한 전체 결함 수도 144개(669-525)라는 차이가 난다.

<표 2>의 여러 특성들을 들어 각 프로젝트들을 결함도 모호성 기준에서 평가하자면 먼저 'defect_num'과 'err_cnt_sum'이 같아야 하고 전체 모듈수 당 이들의 비율인 'defect_percent'와 'err_percent'가 같아야 한다. 5개의 프로젝트 중 이를 만족하는 프로젝트는 없지만 차이가 매우 적은 프로젝트는 KC4, JM1, PC4이다. [그림 3]은 두 형태의 결함 비율을 나타낸다.



[그림 3] 프로젝트 별 두 형태 결함 비율

가장 중요한 모호성 분석 기준인 식 (5)의 두 조건의 결과인 'err_cnt > 0 and defect_no'과 'err_cnt = 0 and defect_yes'는 모호성이 없으려면 모두 0이 되어야 한다. 이를 만족하는 프로젝트는 JM1과 PC4이다. [그림 4]에서는 두 기준들 중 첫 번째 기준이 나타내는 모듈수의 비율을 ambiguity_type1, 두 번째 기준이 나타내는 모듈수 비율을 ambiguity_type2, 총 모호성을 나타내는 두 모호성 비율의 합을 ambiguity_all이라 표현하였다. [그림 4]에서도 JM1, PC4에서 세 가지 값 모두 0이 됨을 알 수 있다.



[그림 4] 두 가지 모호성 타입 분석 결과

5. 결 론

소프트웨어 공학 분야에서도 PROMISE와 같은 공개 데이터 집합들이 생겨나면서 결함 예측 모델에 대한 연구들도 급속도로 증가하였다. 그들 중 가장 많은 연구들에 사용된 데이터 집합은 NASA MDP 데이터 집합이다. 본 논문에서는 NASA 데이터 집합의 결함도 관련 속성들에 모호성이 존재하여 결함도 출력값을 정의 할 때 심각한 문제가 발생한다는 점을 발견하였다.

최근까지 수행된 많은 예측 모델 연구들은 결함 예측 모델의 출력값인 결함 여부를 알아내는 방법으로 product_module_metrics 파일의 속성인 error_count가 0보다 크다는 조건을 사용하였다. 하지만 결함 정보들을 가지고 있는 또 다른 파일인 defect_product_relations을 분석한 결과는 기존 결과들과 차이가 나는 부분이 있었다. 본 논문은 이러한 모호성의 분석 기준을 정의하고 이를 이용하여 13개 프로젝트들 중 결함 정보가 명확하게 정의된 5개의 프로젝트들을 선정하여 모호성 분석 실험을 수행하였다. 그 결과 JM1과 PC4가 가장 모호성이 적은 프로젝트임을 발견하였다.

참 고 문 헌

[1] 홍의석, “훈련데이터 집합을 사용하지 않는 소프트웨어 품질예측 모델”, 「정보처리학회논문지」, 제10-D권, 제4호(2003), pp.689-696.

[2] Catal, C. and B. Diri, “A systematic review of software fault prediction studies”, *Expert Systems with Applications*, Vol.36, No.4(2009), pp.7346-7354.

[3] Catal, C., “Software fault prediction : A literature review and current trends”, *Expert Systems with Applications*, Vol.38, No.4(2011), pp.4626-4636.

[4] Elish, K. O. and M. O. Elish, “Predicting defect prone software modules using support vector machines”, *J. Systems Software*, Vol. 81, No.5(2008), pp.649-660.

[5] Gray, D., D. Bowes, N. Davey, Y. Sun, and B. Christianson, “Reflections on the NASA MDP data sets”, *IET Software*, Vol.6, No.6 (2012), pp.549-558.

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

[7] Khoshgoftaar, T. M. and E. B. Allen, “Ordering fault-prone software modules”, *Software Quality Journal*, Vol.11, No.1(2003), pp. 19-37.

[8] Lessmann, S., B. Baesens, C. Mues, and S. Pietsch, “Benchmarking Classification Models for Software Defect Prediction : A Proposed Framework and Novel Findings”, *IEEE Trans. Software Engineering*, Vol.34, No.4(2008), pp.485-496.

[9] Menzies, T., J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors”, *IEEE Trans. Software Engineering*, Vol.33, No.1(2007), pp.2-13.

[10] Seliya, N. and T. M. Khoshgoftaar, “Software quality analysis of unlabeled program modules with semisupervised clustering”, *IEEE Trans. Systems, Man and Cybernetics*, Vol.37, No.2(2007), pp.201-211.

[11] Shepperd, M., Q. Song, Z. Sun, and C. Mair, “Data Quality : Some Comments on the NASA Software Defect Data Sets”, http://nasa-softwaredefectdatasets.wikispaces.com/file/view/NASA+defect+data+sets+paper.pdf.

[12] Song, Q., Z. Jia, M. Shepperd, S. Ying, and J. Liu, “A General Software Defect-Proneness Prediction Framework”, *IEEE Trans. Software Engineering*, Vol.37, No.3(2011), pp.356-370.

[13] Zhong, S., T. M. Khoshgoftaar, and N. Seliya, “Analyzing Software Measurement Data with Clustering Techniques”, *IEEE Intelligent Systems*, Vol.19, No.2(2004), pp.20-27.

[14] Zhou, Y. and H. Leung, “Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults”, *IEEE Trans. Software Engineering*, Vol.32, No.10(2006), pp.771-789.

◈ 저 자 소 개 ◈

홍 의 석 (hes@sungshin.ac.kr)

서울대학교 계산통계학과 전산과학전공에서 학사, 석사 학위를 취득하였으며 서울대학교 전산과학과에서 박사학위를 취득하였다. 안양대학교 디지털미디어학부 교수로 근무하였으며, 현재는 성신여자대학교 자연과학대학 IT학부 교수로 재직 중이다. 연구 관심분야는 소프트웨어공학 분야 중 소프트웨어 품질, 웹기반 응용 기술 등이다.