소프트웨어 품질 예측 모델을 위한 분류 프레임워크

Taxonomy Framework for Metric-based Software Quality Prediction Models

홍의석
성신여자대학교 IT학부

Euy-Seok Hong(hes@sungshin.ac.kr)

요약

본 논문에서는 위험도라는 품질 인자를 예로 들어 메트릭 기반 소프트웨어 품질 예측 모델들을 네가지 타입으로 분류하는 프레임워크를 제안한다. 모델들은 다음과 같은 두가지 기준에 의해 분류된다: 모델 입력 메트릭 형태, 과거 프로젝트 데이터의 필요 유무. 분류된 타입들은 각각의 특성을 가지며 새롭게 정의된 몇가지 기준들에 의해 타 타입들과 장단점이 비교되었다. 이러한 정성적인 평가를 거쳐 품질 예측 모델을 이용하고자하는 개발 집단은 어떤 품질 예측 모델이 자신들에게 적합한지를 판단할 수 있게 된다. 또한 각 타입에 속하는 위험도 예측 모델들을 구현해 예측 성능을 측정한 선행 연구 데이터를 분석하여 예측 성능에 못지않게 모델이 속한 타입의 특성이 모델 선정의 중요한 관건이 됨을 보였다.

■ 중심어 : | 소프트웨어 품질 | 예측 모델 | 분류 프레임워크 |

Abstract

This paper proposes a framework for classifying metric-based software quality prediction models, especially case of software criticality, into four types. Models are classified along two vectors: input metric forms and the necessity of past project data. Each type has its own characteristics and its strength and weakness are compared with those of other types using newly defined criteria. Through this qualitative evaluation each organization can choose a proper model to suit its environment. My earlier studies of criticality prediction model implemented specific models in each type and evaluated their prediction performances. In this paper I analyze the experimental results and show that the characteristics of a model type is the another key of successful model selection.

■ keyword : | Software quality | Prediction model | Taxonomy framework |

I. 서 론

소프트웨어 산업이 발전함에 따라 소프트웨어 개발 프로세스 개선 및 평가 방법이 주목받고 있다. 소프트웨어 프로세스에서 가장 중요한 것은 주어진 예산과 자원을 이용하여 고품질의 소프트웨어를 제작 및 유지보수 하는 것이다. 이를 위하여 분석이나 설계와 같은 초기 프로세스에서 나중에 구현될 소프트웨어의 품질 인자를 예측하는 예측 모델의 연구가 활발히 진행되고 있다. 왜냐하면 품질에 큰 영향을 미치는 핵심 부분들을

* 이 논문은 2010년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.

접수번호 : #091023-003
접수일자 : 2009년 10월 23일
심사완료일 : 2010년 05월 03일
교신저자 : 홍의석, e-mail : hes@sungshin.ac.kr

초기 프로세스에서 선정하고 이에 맞게 한정된 자원들을 잘 분산 배치함으로써 주어진 시간 내에 고수준의 품질을 보장하는 소프트웨어를 제작할 수 있기 때문이다. 이러한 예측 모델의 필요성은 결과 산물이 수십만 LOC 이상으로 매우 크며 실행 정확성이 요구되는 시스템에 더욱 필요하다.

예측 모델들은 주로 초기 단계의 명세들을 이용한 설계 메트릭들을 사용하며 가장 많이 사용되는 메트릭들은 복잡도에 관련된 메트릭들이다. 예측하고자 하는 품질인자들은 유지보수성이나 위험도 등이었으며 연구들 중 대다수는 위험도에 관한 것들이었다. 설계 개체의 위험도란 개체가 구현되었을 때 갖는 결함경향성을 의미한다[1]. 위험도 예측 연구가 많은 이유는 생산된 소프트웨어에서 결함을 많이 일으킬 부분을 초기 단계에서 미리 찾아 적절한 자원할당을 가능케 함으로써 프로젝트 전체 비용을 낮추고 소프트웨어 품질을 높이는데 매우 유용하게 사용할 수 있기 때문이다. 따라서 본 논문의 소프트웨어 예측 모델은 위험도 예측 모델을 의미한다. 유지보수성과 같은 다른 품질인자들에도 본 연구의 결과와 유사한 프레임워크를 적용할 수 있다.

위험도 예측 모델은 설계 개체를 입력으로 받아 그것이 결함경향 개체인지 아닌지를 판단하는 모델이며, 모델의 입력은 주로 설계 개체들을 정량화 한 메트릭 벡터 형태가 된다. 기존에 제안된 위험도 예측 모델들은 많이 알려진 메트릭 벡터들로 설계 개체들을 정량화 한 후 이들을 위험 그룹과 비위험 그룹으로 분류하는 분류 모델들이 대부분이었다. 이들은 주로 과거 프로젝트에서 얻은 위험도 결과 데이터를 학습하여 현재 프로젝트에 사용하는 훈련 모델들이었으며 훈련 알고리즘으로는 복잡한 통계 기법들이나 인공지능 기법들을 사용하였다. 이들 연구들의 결과를 보면 예측 정확도는 모델에서 사용한 알고리즘에 의해서는 매우 큰 차이를 내진 않는다. 예를 들어, 가장 최근에 사용되었다할 수 있는 SVM(Support Vector Machine)을 이용한 모델의 예측 정확도 실험 결과도 기존 모델들 중 성능이 좋다고 알려진 오류역전파 신경망을 이용한 모델보다 나은 결과를 내지 않았다[2]. 이는 모델에서 사용한 메트릭이나 훈련 데이터 집합 또는 모델의 훈련 파라미터들의 튜닝에 의해 매 실험마다 다른 결과를 낼 수 있음을 의미한다. 이러한 훈련 모델들은 높은 예측 정확도를 보인다는 장점이 있지만 과거 유사한 개발 환경에서 얻은 실제 프로젝트 데이터인 훈련 데이터 집합을 필요로 한다는 큰 문제점이 있다. 대부분의 개발 집단은 이러한 훈련 데이터 집합을 보유하고 있지 않으며[3], 설사 보유하고 있다 하더라도 모델을 적용하려는 현재 프로젝트의 참여 인력, 개발 환경이 과거 프로젝트와 매우 유사하여야 한다. 따라서 예측 정확도에 매우 큰 차이가 없는 한 현재 진행 프로젝트와 개발 집단의 환경에 적합한 예측 모델이 필요하다. 그러려면 예측 모델들을 몇 가지 기준에 의해 정성적으로 평가하여야하고, 이에 앞서 수많은 품질 예측 모델들을 몇가지 형태로 분류하는 연구가 필요하다. 소프트웨어 메트릭 분야에서는 분류에 관한 연구가 있었지만[4], 예측 모델의 분류에 관한 연구는 모델 사용 단계에서 분류트리 모델과 같이 다중 결정 사이클을 사용하는지 여부를 따지는 연구[5] 외에는 없었다. 본 논문에서는 많은 예측 모델들을 분류하기 위해 분류 프레임워크를 정의하고 이를 통하여 이론적으로 의미 있는 네 개의 타입을 정의한다. 모델 평가 기준들을 만들어 모델들을 정성적으로 평가하고 각 타입에 속하는 모델의 예를 들어 간단한 실험 결과를 제시한다. 예측 모델을 사용하고자 하는 집단은 정성적인 평가 결과를 통해 자신들의 환경에 맞는 예측 모델의 타입을 찾아 적당한 모델을 선정할 수 있게 된다. 또한 연구 결과인 모델 분류 용어는 다른 모든 관련 연구들에서 이용함으로써 예측 모델 연구 분야의 일반화된 접근을 가능케 한다는 기대효과가 있다.

2장에서는 메트릭을 이용한 소프트웨어 품질 예측 방법과 기존의 위험도 예측 모델들을 살펴보고 3장에서는 품질 예측 모델을 분류하기 위한 프레임워크의 정의와 그 결과인 네가지 타입을 설명한다. 4장에서는 분류된 모델들을 정성적으로 평가하기 위한 평가 기준과 평가 결과 그리고 각 타입에 속하는 모델들의 예를 들어 간단한 예측성능 결과에 대해 언급하고 5장에는 결론에 대해 기술한다.

II. 관련 연구

1. 소프트웨어 품질 프레임워크

개발 초기 단계에서 개발 후 소프트웨어의 여러 품질 인자들을 예측하는 것은 메트릭들을 개발 프로세스에 통합하는 방법들 중 하나이다. 품질 인자란 라인수, 제어흐름수와 같이 산물 개체로부터 직접 얻을 수 있는 수가 아니라 소프트웨어의 추상적인 품질 특성을 의미한다. 품질 인자에는 유지보수성, 신뢰성, 재사용성, 위험도 등이 있으며 이들은 직접적인 수로 나타낼 수는 없다. 단지 관련된 메트릭들로 이를 추정(estimation)하거나 예측(prediction)할 수 있는 것이다.

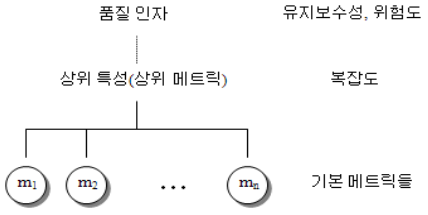

그림 1. 소프트웨어 품질 프레임워크

[그림 1]은 여러 연구자들이 사용한 계층적인 품질 프레임워크 형태이다[6]. 가장 상위의 품질 인자는 가장 하위의 메트릭들 즉 메트릭 벡터인 (m_1, m_2, ..., m_n)으로 정량화되거나 이들 기본메트릭들을 조합하여 정의한 복잡도 메트릭들로 정량화될 수 있다. 이는 객체의 어떠한 품질 특성을 알기 위해 직접 측정할 수 있는 기본 메트릭들을 구하고, 그들의 조합으로 상위 특성을 나타내는 상위 메트릭을 구한 후 그를 이용하여 품질 인자를 추정할 수 있다는 것을 의미한다. 예를 들면 설계 단계에서 한 모듈의 유지보수성 특성을 알아보기 위해 제어 흐름 수, 정보 흐름 수 등의 기본 메트릭들을 구하여 그들의 합으로 상위 수준의 특성인 복잡도 값을 구하고 이를 위험도의 추정치로 사용할 수 있다.

2. 위험도 예측 모델

위험도 예측 모델은 [그림 1]의 품질 프레임워크만으로 설명하기엔 부족하다. 위험도 예측은 각 모듈의 결함수를 추정하는 것보다는 모듈의 위험 여부를 판단하는 것이 중요한 목표이기 때문이다. 따라서 위험도 예측 모델은 [그림 2]와 같이 설계 개체들을 입력으로 받아 위험그룹과 비위험그룹으로 나누는 분류 모델이 대부분이다.

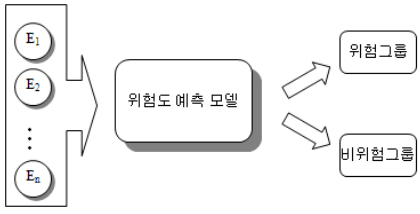

그림 2. 소프트웨어 품질 분류 모델

분류 모델들은 대부분 과거 위험도 데이터를 학습해 현재 프로젝트의 위험도 예측에 사용하는 훈련 모델들이다. 이들은 주로 입력 데이터들을 여러 개의 패턴으로 나누는 패턴 분류 기법들을 사용한다. 사용된 훈련 알고리즘들은 판별분석[7], 로지스틱 회귀분석[8] 등과 같은 통계 기법이나 분류트리[9], 역전파 신경망[10], SVM[2] 등과 같은 인공지능 기법들이다. 이런 모델들의 성능 검증은 모델의 사용방법의 용이성이나 개발환경의 적합도가 아니라 실험을 통한 예측정확도 비교를 통해 이루어졌다. 하지만 이러한 분류 알고리즘들은 데이터의 분포와 프로젝트의 상황, 모델의 튜닝 정도에 따라 다른 성능 결과를 낼 수 있으므로 어떤 모델이 가장 좋다는 합의는 위험도 연구 분야에서 이루어지지 않았다. 최근의 연구들은 [2]와 같이 NASA 오픈 데이터 집합과 같은 수년간 축적된 오픈 데이터 집합을 사용하여 실험을 행하는 경우가 많아지고 있으나 이 역시 위험도와 관련 없는 기본 메트릭들도 모두 입력 메트릭 벡터에 포함된다는 문제점이 있다. [표 1]은 [2]의 실험 결과 중 [2]의 새로운 제안 모델인 SVM과 기존에 많이 알려진 오류역전파 신경망 모델인 MLP의 예측 정확도에 대한 유의성 검정 결과를 요약한 것이다. *Yes*는 두 모델간에 유의미한 결과 차이가 있다는 걸 나타내고 (+)는 SVM이 더 나은 결과를 보인다는 의미이다. 하지만 표에서 보듯이 두 모델이 큰 차이를 보인 6개 부분

에서 (+)는 두 개뿐이므로 새로운 모델인 SVM은 MLP보다 낫다고 할 수 없다. 이는 적어도 훈련 모델이라는 큰 범주 안에서는 새로운 모델의 연구가 큰 의미를 지니지 않는다는 것을 의미한다.

표 1. SVM과 MLP의 유의성 검정 결과 [2]

측정치 / 데이터집합	Accuracy	Precision	Recall
CM1	No(+)	No(−)	Yes(+)
PC1	No(−)	No(−)	No(+)
KC1	Yes(−)	Yes(−)	Yes(+)
KC3	Yes(−)	Yes(−)	No(+)

많지는 않지만 [그림 1]과 같이 프로그램의 위험도를 추정할 수 있는 복잡도 메트릭들을 정의하고 그 타당성을 입증하여 정의한 메트릭들을 바탕으로 시스템의 위험도를 예측하는 연구들도 있었다[11]. 이들 연구들은 복잡도 메트릭이 프로그램의 오류의 분포와 관련이 있음을, 즉 복잡도가 높은 모듈일수록 오류가 발생할 가능성이 높다는 점을 가정하고 있다. 위험도 메트릭 제작은 위험도에 가장 관련이 많은 기본 메트릭을 하나 선정하여 예측에 사용할 수도 있고 위험도와 관련이 있는 기본 메트릭들을 조합하여 하나의 조합 메트릭 형태를 사용할 수도 있다. 후자가 위험도에 관련된 여러 요인들을 고려할 수 있을 것 같지만 여러 메트릭들을 조합함으로써 각 구성 요소들의 특성을 잃어버릴 위험성이 높다[12].

본 연구의 중요한 목적은 이와 같은 많은 예측 모델들을 몇 개의 형태로 분류하여 정성적인 평가를 하는 것이다. [표 1]에서 언급한 것과 같이 같은 타입에 속하는 모델들은 서로 크게 유의미한 예측 성능의 차이를 보이지 않는다는 관찰 하에서는 개발 집단들은 정성적인 평가를 통해 적합한 모델을 선정하여 사용할 수 있을 것이다.

III. 품질 예측 모델의 분류

1. 분류 프레임워크

본 연구에서 제안할 예측 모델 분류 프레임워크는 모델의 입력 형태와 과거 프로젝트 데이터의 필요 여부라는 두가지 독립적인 관점으로 이루어진다.

첫번째 관점은 모델의 입력 형태에 따른 분류이다. 모델의 입력은 기본 메트릭 벡터 형태가 될 수도 있고, 하나의 메트릭 즉 스칼라 메트릭 형태가 될 수도 있다. 스칼라 메트릭은 하나의 기본 메트릭 또는 여러 기본 메트릭들을 조합한 조합 메트릭 형태가 될 수 있다. 즉 [그림 1]에서 스칼라 메트릭은 m_i와 같은 기본 메트릭 또는 $f(m_p, ..., m_q)$ 형태의 조합 메트릭 형태가 가능하다.

메트릭 벡터 입력 모델의 경우 입력 메트릭 벡터를 보고 바로 위험도를 결정할 수는 없다. 그러므로 예측 모델 내부에서는 여러 가지 기법들을 사용하여 이들 입력을 해석하여 결론을 낸다. 이 기법들로 사용되는 것이 앞에서 기술한 여러 인공지능 기법들이나 통계 기법들이 된다. 스칼라 메트릭 입력 모델은 위험도와 관련있는 기본 메트릭을 찾거나 새로운 메트릭을 제작하는 데 초점을 맞추며 자동화된 분류보다는 이들 메트릭 값이 높을수록 위험도가 높아진다는 것을 밝힘으로써 위험도와 같은 품질 인자를 하나의 메트릭값으로 정량화하려는 시도이다. 예를 들자면 스칼라 메트릭 기법 중 기본 메트릭 기법은 LOC나 Cyclomatic Complexity 등과 같은 기본 메트릭값과 위험도와의 연관성을 실험적으로 검증하여, 이들 값이 높을수록 위험도가 높은 개체로 예측한다. 이보다 복잡한 조합 메트릭 기법의 예는 [11]의 연구와 같이 개체의 복잡도를 정의하기 위해 개체의 내부복잡도와 외부복잡도를 기본 메트릭들로 정의하고 이들을 조합하여 하나의 복잡도 메트릭을 만든 후 이를 이용하여 위험도를 예측하는 방법을 들 수 있다.

예측 모델 분류에 사용되는 또 하나의 관점은 모델이 감독형(supervised)인가 비감독형(unsupervised)인가에 대한 것이다. 전자는 모델 제작 시에 과거 프로젝트 데이터를 이용한다. 즉 어떤 입력 값에 대해 알려진 결과(위험도 유무)가 있으며 이 결과는 옳다고 가정하는 것이다. 이를 훈련 데이터 집합이라고 하며 이는 모델을 데이터에 맞게 훈련시키는 데 사용된다. 제작된 모델을 검증하기 위한 데이터 집합을 검증 데이터 집합이

라 한다.

감독형 모델의 문제점은 제작된 모델이 훈련 데이터에 완전히 의존한다는 것이다. 그러므로 과거 프로젝트 데이터로 만들어진 모델을 현재 프로젝트에 적용하려면 과거와 현재 프로젝트를 수행하는 그룹들과 프로젝트들의 성격이 매우 유사해야 한다는 문제점이 있다[3]. 따라서 과거 데이터를 이용하지 않고 현재 진행되고 있는 프로젝트 데이터의 부분을 훈련 데이터 집합으로 하여 만든 모델을 이용하여 반복되는 프로세스에 적용시키려는 노력도 행해지고 있지만, 이 역시 적용이 매우 어려우며 개발 프로세스 상 부자연스럽다. 감독형 모델은 프로젝트의 성격이 변할 때마다 새로이 모델을 만들어야 한다는 부담도 있다. 대부분의 소프트웨어 개발 집단에서 위험도에 관련된 데이터들을 보관하지 않으므로 감독형 모델의 적용은 매우 문제점이 많다고 할 수 있다.

이에 반해 비감독형 모델은 과거 프로젝트 데이터를 필요로 하지 않는다. 과거 지식이 없는 상태에서 현재의 벡터 데이터만으로 개체의 위험도를 알아내는 것은 매우 어렵기 때문에 아직 메트릭 벡터를 입력으로 한 비감독형 모델은 거의 없다. [13]는 Kohonen SOM 신경망을 사용하여 입력 벡터들의 분포를 보고 서로 유사한 입력 부분들을 묶어 클러스터들을 나눈 다음, 나누어진 클러스터들을 사람이 보고 각 클러스터들의 성질(위험도 여부)을 결정하는 모델을 제안하였다. 이 모델은 감독형 모델의 문제점들을 모두 해결하지만 메트릭 값들의 분포에만 결과가 의존한다는 문제점이 있다. 즉 이 모델은 사용하는 복잡도 메트릭이 위험도와 연관성이 매우 깊어야 하며, 메트릭 값들의 분포가 위험도가 높고 낮은 입력 개체에 대해 서로 분명하게 달라야 한다.

그림 3은 독립적인 두가지 분류 관점이 반영된 계층적인 분류 프레임워크이다. 계층 구조를 이루므로 그림에서 하위 모델은 상위 모델 타입들로 구성된다.

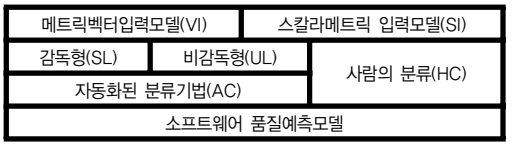

그림 3. 예측 모델 분류 프레임워크

[그림 3]의 가장 윗부분을 해석하면 소프트웨어 품질 예측 모델은 메트릭 벡터 입력 모델과 스칼라 메트릭 입력 모델로 나뉜다. 스칼라 메트릭 입력 모델은 다시 기본 메트릭 입력 모델과 조합 메트릭 입력 모델로 나뉠 수 있지만 이는 그림 상에 나타내지 않았다. 정량화된 결과를 보고 사람이 품질을 분류하는 모델은 훈련 데이터 집합을 필요로 하지 않으므로 비감독형 모델로 볼 수도 있겠지만 비감독형이란 단어가 비감독형 학습 알고리즘에서 사용된 것이므로 감독형과 비감독형 모델은 자동화된 분류기법을 가진 모델로 규정하였으며 사람의 분류 모델을 따로 규정하였다. 따라서 예측 모델은 자동화된 분류 기법 사용 모델과 사람이 품질을 분류하는 모델로 나뉘며 전자는 감독형 학습기법과 비감독형 학습 기법을 사용하는 모델로 나뉜다. 감독형 모델은 메트릭 벡터 입력 모델이고, 비감독형 모델은 메트릭 벡터 입력 모델 또는 스칼라 메트릭 입력 모델이며, 사람이 분류하는 모델은 스칼라 메트릭 모델이다. 스칼라 메트릭 모델도 감독형으로 만들 수 있겠지만 감독형인 경우는 메트릭 벡터 모델의 정확도가 더 높은 것이 명백하므로 이는 프레임워크에서 제외하였다. 또한 메트릭 벡터 입력 모델도 사람의 분류 모델이 될 수 있겠지만 시스템을 이루는 수많은 개체들의 메트릭 벡터들을 보고 사람이 분류하는 것은 사실상 불가능하므로 역시 프레임워크에서 제외하였다.

2. 예측 모델 타입

분류 프레임워크로부터 다음과 같은 네가지 형태의 의미 있는 예측 모델 타입을 얻을 수 있다.

- *VI-SL*(Vector Input – Supervised Learning) 메트릭 벡터 입력 모델이면서 감독형 모델인 경우
- *VI-UL*(Vector Input – Unsupervised Learning) 메트릭 벡터 입력 모델이면서 비감독형 모델인 경우
- *SI-AC*(Scalar Input – Automatic Classification) 스칼라 메트릭 입력 모델이면서 비감독형 모델인 경우
- *SI-HC*(Scalar Input – Human Classification) 스칼라 메트릭 입력 모델이면서 사람이 분류하는 모델인 경우

[그림 3]을 보면 SI-AC 대신 SI-UL이라고 이름 짓는 것이 더 타당하리라 여겨지지만 조합 메트릭을 사용하는 모델에서는 학습(learning)이라는 용어를 사용하지 않으며 단순히 자동 분류를 하는 알고리즘을 사용하는 것에 불과하므로 SI-AC라는 표현을 사용하였다.

정의한 네가지 부류에 속하는 위험도 예측 모델들이 본 연구의 선행연구로 제안되었으며 이들을 각 타입의 비교와 평가에 이용한다. 모델들은 ITU-T의 표준안으로 널리 사용되고 있는 객체지향 실시간 시스템 명세 언어인 SDL(Specification and Description Language)의 설계 명세를 입력으로 사용한다. [그림 4]는 각 부류에 속하는 모델들을 나타낸다. VI-SL 모델로는 오류역전파 신경망을 이용한 BPM, 유전자 알고리즘을 이용한 GAM [14]이 있으며 VI-UL 모델로는 앞에서 기술한 SOM 신경망을 이용한 KSM[13]이 있다. SI-AC와 SI-HC 모델로는 SDL 설계 개체의 내부 복잡도와 외부 복잡도를 가중합과 가중곱 방법으로 혼합한 혼성 복잡도 메트릭을 사용하는 HMM-AC와 HMM-HC가 있다 [15]. [그림 4]에서 TRE_i는 훈련 데이터 집합의 개체를 나타내고 TRM_i와 FP_i는 이 개체를 정량화한 메트릭 벡터와 위험도 유무 결과값을 나타낸다. 이들은 훈련 데이터 집합을 필요로 하는 VI-SL 모델에만 사용된다. E_i와 M_i는 현재 프로젝트의 설계 개체와 정량화된 메트릭 벡터를 나타내며 벡터 입력 모델인 VI 모델에 사용된다. C_i는 E_i의 혼성복잡도를 나타내며 SI 모델인 HMM에 사용된다.

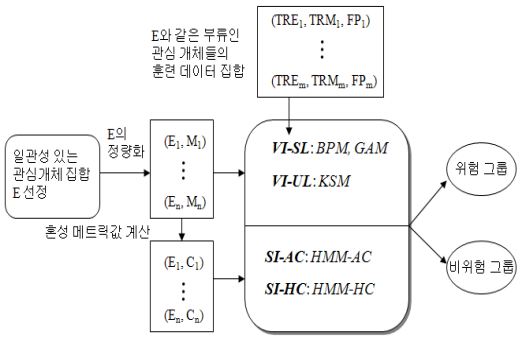

그림 4. 네가지 타입의 예측 모델 예들

IV. 품질 예측 모델의 평가

1. 정성적 평가

네가지 타입에 해당하는 위험도 예측 모델들을 정성적으로 비교하기 위해 모델들을 비교하는 여러 기준들과 각 기준에 대한 모델들의 평가 결과를 [그림 5]에 나타내었다. 결과는 다섯 단계로 표현되며 좋은 평가 결과일수록 밝게 나타내었다. 각 단계는 몇 배가 좋다는 비율적인 의미는 없으며 서수적인 의미만 있다. 본 절에서는 각 기준에 대한 모델 평가의 근거를 기술한다.

모델 / 기준	*HMM-AC*	*HMM-HC*	*BPM*	*KSM*
① 학습 비용				
② 사람의 분석 단계 필요				
③ 조합 메트릭 문제				
④ 결과의 원인 분석 어려움				
⑤ 훈련데이터집합 필요				
⑥ 위험도 정량화 (개체 비교 가능)				
⑦ 적용 데이터 분포에 의존				

그림 5. 예측 모델들의 정성적 평가

①은 모델을 학습시키는데 드는 비용이며 SL 모델인 BPM이 높다. 신경망을 학습시키는데 많은 계산을 수행해야 하는 KSM 역시 UL 모델이지만 높은 비용을 요구한다. ②는 모델 사용 단계에서 사람의 분석 단계가 필요한 것인가에 대한 것이며 이는 UL 모델이 가진 문제점들 중 하나이다. KSM이 클러스터들을 위험 부분과 비위험 부분으로 나누는 분석 단계의 필요성 때문에 가장 높으며 HMM은 스칼라 메트릭 값을 다루므로 상대적으로 비용이 작다. HMM-AC는 자동으로 클러스터링을 수행하나 위험 개체수가 너무 많거나 적게 선정되었을 때 사람의 분석에 의한 분할 결정이 필요하고, HMM-HC는 분할 결정을 사람에만 의존하는 것이다. HMM은 이를 혼합한 형태로 사용하는 것이 바람직하나 모델 비교에서는 이들을 분리 기술하였다. ③은 조

합 메트릭 형태를 사용하는 모델의 문제점이다. HMM은 조합 메트릭인 혼성 복잡도 메트릭을 입력으로 사용하므로 기본 메트릭들의 성질을 잃을 가능성이 있다. 그러나 모의실험 결과 이로부터 생기는 예측 오류는 매우 적었다. 그러므로 표에서 가장 나쁜 평가가 아니라 중간 수준의 평가를 내렸다.

④는 위험도 예측 결과에 대한 원인 분석의 어려움 정도를 나타낸다. 만약 예측 결과가 잘못 되었다면 원인을 분석하여 모델에서 이를 제거해야 하며, 모델이 현재 프로젝트에 사용될 때도 위험 개체에 대한 원인 분석을 하여야만 적당한 자원 할당 등의 조치를 취하기 쉽다. HMM은 혼성 메트릭의 정의와 각 부분을 형성하는 부분들이 명확히 기본 메트릭들로 정의되어있기 때문에 혼성 복잡도가 커지는 이유를 하위 기본 메트릭 값들을 분석해봄으로써 알 수 있다. KSM은 각 클러스터의 성격이 분석 단계에서 규명되면 입력 개체가 어떤 클러스터에 속하는 것에 대한 원인 분석이 가능하다. 즉 KSM은 분석 단계에서 여러 클러스터의 성격 규명에 많은 비용이 드는 대신 예측 결과에 대한 원인 분석의 비용은 매우 적다. 훈련 데이터 집합의 성질에 의존하는 SL 모델은 주로 적용 데이터 분포에만 의존하는 UL 모델보다 원인 분석이 어렵다. BPM은 블랙 박스적인 성격을 가지므로 원인 분석이 거의 불가능하다.

⑤는 당연히 SL 모델인 BPM에만 사용되는 비용이다. ⑥은 위험도 값을 정량화 할 수 있는가에 대한 것이다. 위험도 정량화로 가능한 것은 개체간의 위험도 비교이다. HMM은 위험도를 하나의 혼성 메트릭으로 정량화 하므로 이 부분에서 가장 뛰어나다. 그러나 정의한 혼성 메트릭은 비율 스케일(ratio scale)이 아니기 때문에 “p개체가 q개체보다 위험도가 높다” 등의 서수 스케일(ordinal scale)에 관련된 결정만 내릴 수 있지 몇 배 더 위험하다는 등의 비율에 관한 결정은 내릴 수 없다. KSM은 위험도를 스칼라 값으로 정량화 하지는 않지만 여러 클러스터의 성질을 정함으로써 각 클러스터의 위험도 비교도 어느 정도 가능케 한다.

⑦은 적용 데이터 분포에 주로 의존하는 UL 모델의 단점이다. 적용 데이터의 분포에 따라 벡터를 직접 클러스터링 하는 KSM은 적용 데이터 집합의 분포에만 의존하기 때문에 문제점이 매우 크다. 그러나 HMM 같은 경우는 기본 메트릭들을 의미적으로 잘 정의한 혼성 복잡도 제작 프레임워크에 맞추어 혼성 메트릭으로 바꾸었기 때문에 적용 데이터의 분포에 대한 직접 의존도를 많이 낮추었다. 즉 HMM은 혼성 메트릭 값의 분포에 의존한다. HMM-HC는 사람이 위험 그룹의 크기를 정하는 경우도 있으므로 복잡도 분포에 의해 자동 클러스터링 하는 HMM-AC보다는 분포에 덜 의존한다고 할 수 있다.

[그림 5]에 나타난 정성적인 비교 결과는 HMM이 우수하게 나타난다. 그러나 [그림 5]는 각 기준들의 상대적인 중요도 가중치를 고려하지 않았고, 중요한 모델 평가 기준인 예측 정확도 비교가 없다. 모델의 특성만을 고려한다면 적용 데이터와 유사한 데이터 영역 공간에 속하는 훈련 데이터 집합 결과를 학습한 VI-SL 모델, 즉 BPM이 높은 예측 정확도를 나타낼 것이다.

2. 예측 성능

표 2는 선행 연구인 [13-15]의 예측 성능 실험 결과의 중요 부분을 요약한 것이다. 선행 연구들 모두 같은 데이터 집합을 사용하였으므로 비교가능하며 표에 기술한 결과들은 가장 일반적인 동일한 환경 하에서 모델별로 가장 좋은 결과들을 나타낸 것이다. 훈련2, 검증2는 위험 그룹과 비위험 그룹간의 차이를 크게 해 위험도 판단이 애매한 개체들을 없앤 데이터 집합을 나타낸다. Type I은 비위험 개체수에 대해 비위험 개체를 위험 개체로 선정한 수를 나타내고, Type II는 위험 개체수에 대해 위험 개체를 비위험 개체로 선정한 수를 나타낸다. 후자가 전자보다 품질에 중요한 영향을 미치는 오류이다. BPM과 GAM은 훈련2보다 일반적인 훈련1로 훈련시킨 결과이며, BPM이 검증1에 대해서는 위험개체를 23개를 선정하였으므로 HMM-HC는 위험개체를 23개로 선정한 결과를 나타내었다. 혼성 메트릭 정의에 포함되지 않는 BRS 메트릭값을 고려한 HMM-HC 결과도 나타내었으며 HMM-AC에서는 가중곱, HMM-HC에서는 가중합 형태의 혼성 메트릭이 더 나은 결과를 보였으므로 해당 메트릭 형태를 사용한 결과를 나타내었다. KSM 결과는 클러스터를 15개로 하고

30번 클러스터링을 수행한 결과의 평균값이다.

표 2. 예측 모델들의 예측정확도 예

모델 \ 판별 오류	검증1		검증2	
	Type I	Type II	Type I	Type II
BPM	4/177	4/23	7/177	0/23
GAM	6/177	4/23	2/177	0/23
HMM-AC	24/177	5/23	0/177	5/23
HMM-HC	8/177	8/23	2/177	2/23
HMM-HC(BRS)	5/177	5/23	1/177	1/23
KSM	16.8/177	2.6/23	5.8/177	1.2/23

실제 프로젝트 데이터는 검증2 형태를 보일 가능성이 많으므로 검증2 부분의 Type II 오류가 중요한 성능 비교값이 된다. 결과를 보면 예상한 것과 같이 VI-SL 모델인 BPM, GAM이 좋은 예측 성능을 보인다. 또한 타 모델들도 매우 큰 오류를 내지는 않았으므로 HMM-AC를 제외하고는 다른 모델들도 사용가능하다는 것을 알 수 있다. [표 2]에는 정확히 나타나있지 않지만 실험 결과 전체를 분석해보면 BRS를 고려한 HMM이 BPM의 성능에 근접한 성능을 보였으며, KSM은 전체 적중률 면에서 BRS를 고려하지 않은 HMM-HC와 비슷한 성능을 보였지만 Type II 오류가 BPM과 비슷하게 적었다.

3. 토론

VI-SL 모델이 가장 좋은 예측 정확도를 보인다고 해서 항상 사용할 수 있는 것은 아니다. 하지만 개발 집단이 현재 프로젝트 데이터와 매우 유사한 과거 훈련 데이터를 보유하고 있다면 VI-SL 모델을 사용하는 것이 바람직하다. 대부분의 개발 집단은 훈련 데이터 집합을 보유하고 있지 않다. 이때에는 BRS를 고려한 HMM을 사용할 수 있다. HMM 사용 시 문제점은 위험 그룹의 크기를 결정하기 어렵다는 것이었다. 가중곱 메트릭을 사용하는 HMM-AC가 어느 정도 만족할만한 결과를 냈지만 항상 올바른 결과를 내리라고는 예상되지 않는다. 위험 그룹의 크기를 입력 집합의 약 10%로 하여 HMM-AC와 함께 HMM-HC를 사용하는 것이 바람직하다.

HMM의 부정확성의 원인 중 하나는 조합 메트릭의 문제점이었다. HMM에서 제공하는 혼성 메트릭은 스칼라 형태이기 때문에 결과값과 그에 사용되는 메트릭 벡터를 함께 관찰해야 결과에 대한 이유를 분석할 수 있다. KSM은 이러한 결과의 분석 부분을 위험 클러스터와 비위험 클러스터를 선정하는 부분으로 대체하였다. KSM은 메트릭 벡터들의 분포에 의해 입력 개체들을 여러 패턴들로 나누며 이는 입력 집합 개체들의 여러 성격들을 정확히 구분할 수 있는 기회를 제공한다. 클러스터들의 성질을 분석하면서 위험도뿐만 아니라 다른 품질 인자들에 관한 클러스터들을 결정할 수 있으므로 유지보수성과 같은 다른 품질 인자 예측 모델로의 확장이 용이하다.

위험도 예측 모델로 위험 그룹이 선정되면 위험 그룹을 분석하여 선정 원인을 밝힌 후 그를 최소화 하는 방향으로 재설계를 할 수도 있고, 이유 분석 없이 위험 개체의 구현에 더 많은 자원을 할당할 수도 있다. 물론 이는 극단적인 두 형태의 개발 프로세스이며, 대부분은 이를 적당히 혼합한 프로세스가 사용된다. 프로젝트의 규모와 생산 소프트웨어의 크기가 커질수록 원인 분석 과정이 필요하며, 분석 과정에 참가하는 전문가들도 많이 필요하다. 분석 과정에서는 원인 분석이 쉬운 KSM이나 HMM이 VI-SL 모델보다 분석을 용이하게 한다.

SDL이 사용되는 통신 소프트웨어 개발과 같이 대규모 프로젝트에서는 위험도 예측에 걸리는 시간보다는 예측 정확도가 훨씬 중요하다. 만약 훈련 데이터 집합이 없다면 HMM과 KSM을 함께 사용하여 결과를 비교하여 최종 결론을 내리는 것이 가장 바람직하다. HMM과 KSM은 매우 간단한 알고리즘을 사용하여 개발 부담이 없으며, 모델 속도와 선정 원인 분석에 큰 강점을 지니고 있기 때문이다. HMM과 KSM을 같이 사용하면 SI-HC, SI-AC, VI-UL 타입들의 장점들을 모두 받아들일 수 있으며, 분석을 통해 각 모델의 예측 정확도보다 높은 정확도를 낼 수 있다. 만약 비용 문제나 분석 전문가 부족 등의 문제로 두 모델을 함께 사용할 수 없는 상황이라면 앞의 기술과 같이 BRS를 고려한 HMM을 사용하는 것이 가장 바람직하다.

V. 결 론

대형 소프트웨어 개발이 많아지고 소프트웨어 프로세스에 관심이 많아짐에 따라 위험도 예측 모델과 같은 소프트웨어 품질 예측 모델을 사용하여 전체 개발 비용을 낮추면서 고품질의 소프트웨어를 생산하는데 대한 관심이 높아지고 있다. 많은 관련 연구들이 수행되었지만 새로운 알고리즘을 사용하여 예측 성능을 높이려고 한 것들이 대부분이었다. 본 논문에서는 많은 예측 모델들을 분류하기 위한 프레임워크를 정의하고 이론적으로 가능한 네개의 예측 모델 타입을 정의하였다. 또한 각 타입에 속하는 위험도 예측 모델들을 제시하고 비교기준을 정의하여 정성적인 비교 평가를 실시하였다. 몇가지 관련 연구나 실험들을 통하여 VI-SL 모델과 같이 같은 부류에 속하는 예측 모델들 중 좋은 성능을 갖는다고 알려진 모델들의 예측 정확도에는 유의미하게 큰 차이가 없음을 알 수 있었다. 따라서 모델을 필요로 하는 개발 집단들은 분류된 네가지 형태 정보와 정성적인 평가를 이용하여 자신들의 상황에 맞는 적당한 모델을 선정하여 이용할 수 있다. 위험도 예측 모델의 경우는 훈련 데이터 집합을 보유하고 있는 경우에는 BPM이나 GAM 같은 VI-SL 모델, 보유하고 있지 않은 경우에는 HMM과 KSM을 함께 사용하는 것이 적당하며 이들의 사용도 어려운 경우에는 BRS를 고려한 HMM을 사용하는 것이 적당하다는 결론을 얻었다.

참 고 문 헌

[1] C. Ebert, "Fuzzy classification for software criticality analysis," Expert Systems with Applications, Vol.11, No.3, pp.323-342, 1996.

[2] K. O. Elish and M. O. Elish, "Predicting defect prone software modules using support vector machines," J. Systems Software, Vol.81, No. 5, pp.649-660, 2008

[3] N. Ohlsson and H. Alberg, "Prediction Fault-Prone Software Modules in Telephone Switches," IEEE Trans. Software Eng., Vol.22, No.12, pp.886-894, 1996.

[4] F. B. Abreu and R. Carapuça, "Candidate metrics for object-oriented software within a taxonomy framework," J. Systems Software, Vol. 26, No.1, pp.87-96, 1994.

[5] L. Chen and S. Huang, "Accuracy and efficiency comparisons of single- and multi-cycled software classification models," Information and Software Technology, Vol.51, No.1, pp.173-181, 2009.

[6] J. McCall, P. Richards, and G. Walters, "Factors in Software Quality," three volumes, NTIS AD-A049-014, 015, 055, 1997.

[7] T. M. Khosgoftaar and E. B. Allen, "Early Quality Prediction: A Case Study in Tele communications," IEEE Software, Vol.13, No.1, pp.65-71, 1996.

[8] K. E. Emam, W. Melo, and J. C. Machado, "The prediction of faulty classes using object oriented design metrics," J. Systems Software, Vol.56, No.1, pp.63-75, 2001.

[9] J. Tian, A. Nguyen, C. Allen, and R. Appan, "Experience with identifying and characterizing problem-prone modules in telecommunication software systems," J. Systems Software, Vol.57, No.3, pp.207-215, 2001.

[10] T. M. Khoshgoftaar and D. L. Lanning, "A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase," J. Systems Software, Vol.29, No.1, pp.85-91, 1995.

[11] W. M. Zage and D. M. Zage, "Evaluating Design Metrics on Large-Scale Software," IEEE Software, Vol.10, No.4, pp.75-81, 1993.

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

[13] 홍의석, “훈련데이터집합을 사용하지 않는 소프트웨어 품질예측 모델”, 정보처리학회논문지, 제10-D권 제4호, pp.689-696, 2003.
[14] 홍의석, “GAM: 대형 통신 시스템을 위한 위험도 예측 모델”, 컴퓨터교육학회논문지, 제6권, 제2호, pp.33-40, 2003.
[15] 홍의석, “대형 소프트웨어 시스템의 결함경향성 예측을 위한 혼성 메트릭 모델”, 컴퓨터교육학회논문지, 제8권, 제5호, pp.129-137, 2005.

저 자 소 개

홍 의 석(Euy-Seok Hong) 정회원

- 1992년 : 서울대학교 계산통계학과(학사)
- 1994년 : 서울대학교 계산통계학과(석사)
- 1999년 : 서울대학교 전산과학과(박사)
- 1999년 ~ 2002년 : 안양대학교 디지털미디어학부 교수
- 2002년 ~ 현재 : 성신여자대학교 IT학부 교수

<관심분야> : 소프트웨어공학, 소프트웨어 품질, 웹기반 응용 기술, CAI