대형 소프트웨어 시스템의 결함경향성 예측을 위한 혼성 메트릭 모델

홍의석†

요 약

설계 명세를 이용하여 결함경향성이 많은 부분을 예측하는 위험도 예측 모델은 대형 통신 시스템 같이 결과 산물이 매우 큰 시스템의 개발비용을 낮추는데 중요한 역할을 하고 있다. 복잡도 메트릭에 기반한 많은 위험도 예측 모델들이 제안되었지만 그들 대부분은 모델 훈련을 위한 훈련 데이터 집합을 필요로 하고, 설계 개체들을 위험 그룹과 비위험 그룹으로 나누는 기능만 지닌 분류 모델들이었다. 본 논문에서는 두가지 형태의 검증된 혼성 메트릭들을 사용하는 새로운 예측 모델 HMM을 제안한다. HMM의 장점은 설계 개체의 위험도를 정량화함으로써 모델 훈련을 위한 훈련 데이터 집합이 필요 없다는 것과 개체 간에 위험도 비교가 가능하다는 것이다. HMM의 유용성을 보이기 위해 여러 내부 특성들과 예측 정확도 비교를 통해 잘 알려진 예측 모델인 역전파 신경망 모델(BPM)과 HMM을 비교하였다.

키워드 : 결함경향성, 위험도, 혼성 메트릭, 예측 모델

Hybrid metrics model to predict fault-proneness of large software systems

Euy-Seok Hong†

ABSTRACT

Criticality prediction models that identify fault-prone spots using system design specifications play an important role in reducing development costs of large systems such as telecommunication systems. Many criticality prediction models using complexity metrics have been suggested. But most of them need training data set for model training. And they are classification models that can only classify design entities into fault-prone group and non fault-prone group. To solve this problem, this paper builds a new prediction model, HMM, using two styled hybrid metrics. HMM has strong point that it does not need training data and it enables comparison between design entities by criticality. HMM is implemented and compared with a well-known prediction model, BackPropagation neural network Model(BPM), considering internal characteristics and accuracy of prediction.

Keywords : fault-proneness, criticality, hybrid metrics, prediction model

1. 서 론

대형 소프트웨어 제작이 증가함에 따라 설계 메트릭들로 설계 단계 산물들을 정량화 하여 최종 개발물의 품질 인자들을 예측하는 품질 예측 모델들의 중요성이 매우 높아졌다. 왜냐하면 개발 초기 단계의 문제점이 개발 후반부 산물의 품질에 매우 심각한 영향을 미치고 대부분의 소프트웨어 결점들이 매우 적은 수의 모듈들로부터

†정 회 원: 성신여자대학교 컴퓨터정보학부 교수
논문접수: 2005년 2월 4일, 심사완료: 2005년 4월 4일
* 이 논문은 2003년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음

발생한다고 알려져 있기 때문이다[1]. 이러한 설계 정량화와 예측 모델의 필요성은 결과 산물이 매우 크고 실행 정확성이 요구되는 대형 통신 소프트웨어 같은 실시간 시스템 설계에 더욱 필요하다.

설계 단계 산물들의 복잡도 메트릭들을 이용하여 최종 개발물의 품질 인자들을 예측하는 예측 모델들에 대한 연구들은 최근에도 많이 행해지고 있다. 한 설계 개체의 위험도란 개체가 구현되었을 때 갖는 결함경향성을 의미하는 품질 인자이다. 기존에 제안된 위험도 예측 모델들은 기본 메트릭들로 구성된 메트릭 벡터들로 설계 개체들을 정량화 한 후 이들을 위험 그룹과 비위험 그룹으로 분류하는 분류 모델들이 대부분이었다. 하지만 이들은 단순히 설계 개체의 위험 여부만 판단할 뿐 여러 설계 개체들의 위험도를 서로 비교할 수 없으며, 대부분 위험도 결과의 원인을 분석하기 어려운 블랙박스적인 모델들이었다. 또 다른 중요한 문제점은 이들 모델들은 모델 학습을 위해 과거 유사한 개발 환경에서 얻은 실제 프로젝트 데이터인 훈련 데이터 집합을 필요로 한다는 점이다. 대부분의 개발 집단은 이러한 훈련 데이터 집합을 보유하고 있지 않으며, 설사 과거 데이터를 보유하고 있다 하더라도 현재 진행 중인 프로젝트의 참여 인력, 개발 환경, 설계 정량화 메트릭 집합이 과거 프로젝트와 매우 유사하여야만 한다는 문제점이 있다[2]. 이러한 문제점은 한 프로젝트를 위해 개발된 모델을 다른 프로젝트에 사용할 수 없다는 편협성의 문제도 동반한다. 그러므로 대부분의 개발 집단은 기존의 예측 모델들을 현재 프로젝트에 직접 적용할 수가 없다.

본 논문은 위험도를 정량화하는 설계 메트릭을 정의하여 기존 모델들의 문제점들을 해결하는 예측 모델 HMM(Hybrid Metrics Model)을 제안한다. 위험도 같은 소프트웨어 품질 인자를 하나의 스칼라 메트릭으로 정량화하는 것은 위험도가 갖는 부분 성질을 조합하기가 어렵다는 점에서 위험하지만 매우 많은 유용성을 제공한다. HMM은 훈련 데이터 집합이 필요 없으며, 위험도 값을 정량화함으로써 한 설계 개체가 위험 그룹에 속하는지 여부 판단뿐만 아니라 두 개체 중 어느 것이 더 위험한가에 대한 비교를 가능하게 해준다. 또한 프로젝트 환경, 종류 등에 상관없이 어느 프로젝트에나 적용할 수 있는 일반화된 모델이다. 예측 정확도 면에서는 훈련 데이터 집합을 사용하는 모델보다 뛰어날 수는 없지만 이런 많은 유용성을 가진 HMM이 기존 방법들보다 예측 정확도 면에서 얼마나 뒤떨어지는가를 알아보는 것도 본 논문의 중요한 목적 중 하나이다.

위험도 예측 모델은 수천, 수만 LOC(Line Of Code) 정도의 시스템보다는 수십만 LOC 이상의 대형 시스템을 개발하는 경우에 유용하다. 수십 개의 설계 개체로 구성된 시스템의 경우에는 설계자의 의미적인 판단으로 위험도가 높은 개체들을 선정할 수 있지만 수백개 이상의 설계 개체로 구성된 시스템의 경우에는 위험 부분을 찾는 자동화된 방법이 필요하기 때문이다. 그러므로 위험도 예측 모델에 관한 연구는 주로 대형 통신 시스템 등을 개발하는 개발 집단을 중심으로 행해져왔다. 제안 모델의 입력 대상은 ITU-T의 표준안으로 사용되고 있는 객체지향 실시간 시스템 명세 언어인 SDL(Specification and Description Language)로 작성한 설계 명세이며 이를 정량화하는 기본 메트릭들은 [3]에서 제안한 SDL 메트릭 집합을 사용한다. 기본 메트릭들을 이용하여 혼성 메트릭을 정의하는 기법은 [4]의 혼성 복잡도 메트릭 제작 프레임워크를 사용한다. 이 메트릭 집합의 이론적 타당성 및 유용성은 공리적 검증 방법과 차원 분석 과정을 통해 검증되었다[5].

2장에서는 기존에 제안된 위험도 예측 모델들을 간략히 살펴보고 3장에서는 새로운 예측 모델의 구조와 사용법을 설명한다. 4장에서는 모의실험을 통하여 역전파 신경망 모델과 제안 모델의 예측 정확도를 비교하고 5장에는 결론과 향후 연구 과제에 대해 기술한다.

2. 관련 연구

시스템 개발 초기 단계에서 위험도를 예측하기 위한 관련 연구들은 크게 두가지로 구분할 수 있다. 첫 번째는 과거 유사 프로젝트의 위험도 자료들 즉 훈련 데이터 집합에 기반한 모델을 만들어 현재 수행중인 프로젝트의 위험도 예측에 적

용하는 것이다. 이들은 주로 입력 데이터들을 여러 개의 패턴으로 나누는 패턴 분류 기법들을 사용하며 예로는 판별분석[6], 분류트리[1], 역전파 신경망[7], 회귀분석[8], 회귀트리[9], CBR(Case Based Reasoning)[10], 유전자 알고리즘[11] 등이 사용되었다. 이런 모델들의 성능은 데이터의 분포와 프로젝트의 상황에 따라 다른 결과를 내므로 어떤 것이 가장 좋다고 할 수 없지만 예측 정확도 측면에서만 본다면 역전파 신경망 모델과 CBR 모델이 비교적 좋은 성능을 보이는 것으로 알려져 있다. 대부분의 위험도 예측 모델에 대한 연구들은 이러한 형태를 취하지만 앞의 기술과 같이 분류 모델이라는 점, 결과에 대한 원인 분석의 어려움, 훈련 데이터 집합이 필요하다는 문제점들을 가지고 있다.

두 번째는 프로그램의 위험도를 예측할 수 있는 메트릭들을 정의하고 그 타당성을 입증하여 정의한 메트릭들을 바탕으로 시스템의 위험도를 예측하는 것이다. 즉 이는 단순히 어떤 설계 개체가 위험한가 아닌가의 여부 결정이 아니라 실제 위험도 값을 정량화 한다. 품질 인자를 관련 메트릭 값으로 추정하듯이 여기서 정량화란 메트릭 값이 직접적인 위험도 값이라는 의미가 아니라 위험도와 연관이 많은 수치값이라는 의미이다. 이런 형태의 모델은 훈련 데이터 집합이 필요 없다는 것과 모델 학습 과정이 없으므로 처리 속도가 빠르다는 장점을 갖는다. 그러나 기존 데이터들을 통한 경험적인 지식이 아니라 하나의 복잡도 메트릭 값에 의존한다는 단점이 있다.

스칼라 메트릭 값으로 위험도를 예측하는 연구로는 Zage의 연구[12], Agresti의 연구[13]와 데이터 바인딩 기법[14] 등이 있다. 이들 중 Zage의 연구를 제외하고는 개체 복잡도의 일부분만의 관점을 갖는 부분 복잡도 형태의 메트릭으로 위험도를 정량화 하였다. Zage는 프로그램 모듈의 복잡도를 외부 복잡도와 내부 복잡도로 나누고 이들의 합을 모듈의 복잡도로 정의하였으나 이는 각 부분 복잡도의 가중치를 고려하지 않은 조합 메트릭이라는 문제점이 있다. 또한 Zage는 계산된 복잡도를 가지고 극단 이상점을 제외한 복잡도의 평균보다 1 표준 편차가 큰 곳을 경계로 하여 결함 경향 여부를 판단하였으므로 위험 집합의 클러스터링에 사람의 결정을 배제하였다.

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 기본 메트릭들을 조합하는 것은 매우 주의를 요한다. 여러 메트릭들을 조합함으로써 각 구성 요소들의 특성을 잃어버릴 가능성이 있기 때문이다[15]. 그러므로 본 논문에서는 SDL 설계 개체들의 위험도를 정량화하기 위해 [4]에서 제안한 혼성 복잡도 메트릭 제작 프레임워크를 사용한다.

두 가지 방향의 관련 연구들 모두 소프트웨어의 복잡도 메트릭이 프로그램의 오류의 분포와 관련이 있음을, 즉 복잡도가 높은 모듈일수록 오류가 발생할 가능성이 높다는 점을 가정하고 있으며 본 논문 역시 이와 같은 가정을 기본 전제로 한다.

3. 제안 모델

제안 모델 구조는 SDL 설계 명세의 관심 개체를 정량화하는 혼성 메트릭 정의 부분과 여러 개의 개체 정량화 값들로부터 위험 그룹을 결정하는 클러스터링 부분으로 나뉜다. SDL을 이용한 설계는 구조 설계 단계와 상세 설계 단계로 나뉘며 정량화가 적절한 개체 형태는 구조 설계 단계의 블록과 상세 설계 단계의 프로세스이다. 두 형태 모두 같은 방법으로 정량화가 가능하므로 본 논문에서는 전자의 경우만 기술한다.

3.1. 혼성 메트릭 형태

한 설계 개체 E의 복잡도는 E와 외부 개체들과의 상호작용에서 발생하는 외부 복잡도와 E의 내부 구조 및 정보량에 의해 발생하는 내부 복잡도에 의해 영향 받는다. [4]는 시스템 내에 존재하는 기능 개체 E의 외부 복잡도, 내부 복잡도와 이를 혼합한 두가지 형태의 혼성 복잡도 메트릭을 제작할 수 있는 혼성 복잡도 메트릭 제작 프

레임워크를 제안하였다. 제작된 혼성 복잡도 메트릭의 이론적 타당성 및 유용성은 공리적 검증 방법과 차원 분석 과정을 통하여 검증 되었으므로[5], 본 논문은 이 프레임워크를 이용한다.

E를 시스템 S를 구성하는 임의의 기능 개체라 할 때 내부 복잡도, 외부 복잡도, 혼성 복잡도는 다음과 같이 정의된다.

I_i, O_i: 같은 종류의(i번째의) 입출력 정보 집합 ($1 \le i \le n$)
VP_i : i번째 관점에 의한 E 내부의 정보 집합 ($1 \le i \le m$)
$IC(E)$: E의 내부 복잡도

$$IC(E) = f\left(\sum |VP_i|\right), \quad f : Z+ \rightarrow R+ \text{인 증가함수} \quad (1)$$

$Z+ = \{x \mid x \in Z, x \ge 0\}$ $R+ = \{x \mid x \in R, x \ge 0\}$[2]
$EC(E)$: E의 외부 복잡도

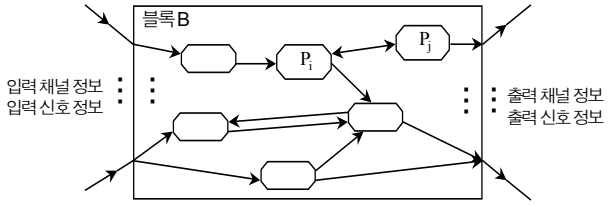
$$EC(E) = g\left(\sum((|I_i|+c)(|O_i|+c))\right), \quad (2)$$

$g : Z+ \rightarrow R+$ 인 증가함수, $c \in Z+$
$C(E)$: E의 혼성 복잡도(C1, C2는 가중합, 가중곱 형태)

$$C1(E) = w_f IC(E) + w_e EC(E), \ (w_i < w_e) \quad (3)$$

$$C2(E) = IC(E) \cdot EC(E)^2 \quad (4)$$

SDL 메트릭들은 SDL 설계 단계와 각 단계의 산물들을 모델화한 구조화된 SDL 모델[3]로부터 정의되며, FBID(Flat BID)는 구조 설계 단계에서 시스템을 가장 자세한 단위들로 분할하였을 때 전체 시스템을 나타내는 BID(Block Interaction Diagram)이다[3].



<그림 1> 구조 설계 단계의 블록

구조 설계 단계는 시스템의 블록 구조와 끝단 블록들의 BID가 결정되는 단계이므로 블록간의 상호 작용은 채널과 신호에 의해서만 이루어지고 각 프로세스간에 주고받는 데이터는 정의되지 않는다. 구조 설계 단계에서 프로세스와 프로세스 외부와의 상호 작용은 라우트들과 신호들에 의해 정의할 수 있으나 프로세스의 내부에 대한 정보는 정의되지 않는다. <그림 1>은 블록 B를 나타내는 사각형 심볼 안에 그의 상세 정보인 FBID를 나타낸 것이다.

블록 B는 구조 설계 단계에서 기능 개체 E의 예이며, 앞에서 기술한 혼성 복잡도 메트릭 제작 프레임워크에 의한 블록 B의 정량화 결과는 다음과 같다.

VP_1 = {x | x는 B의 FBID를 구성하는 프로세스 }
VP_2 = {x | x는 B의 FBID를 구성하는 라우트 }
VP_3 = {x | x는 B의 FBID를 구성하는 신호 }
I_1 = {x | x는 B의 입력채널} O_1 = {x | x는 B의 출력채널}
I_2 = {x | x는 B의 입력신호} O_2 = {x | x는 B의 출력신호}
$IC(B) = \sum |VP_i| = BP + BS + BR$, $f(x) = x$
(BP, BS, BR: number of Processes/Routes/Signals of a Block)
$OC(B) = \sum(|I_i| \cdot |O_i|) = \sqrt{IBC \times OBC + IBS \times OBS}$,
$g(x) = \sqrt{x}$ (IBC,OBC,IBS,OBS: number of Input/Output Channels/Signals of a Block)
$C1(B) = w_f IC(B) + w_e EC(B)$, $(w_i, w_e) = (0.2, 0.8)$
$C2(B) = IC(B) \cdot EC(B)^2$

g(x)로 제곱근 함수를 사용한 이유는 차원 분석의 검증 조건[5]을 만족시키기 위해서이며 내부 복잡도와 외부 복잡도 가중치 값은 [4]의 실험 결과에 의한 것이다. B의 C1, C2라는 표현은 각각 B의 가중합 형태의 혼성 메트릭과 가중곱 형태의 혼성 메트릭을 나타낸다.

3.2. 클러스터링 형태

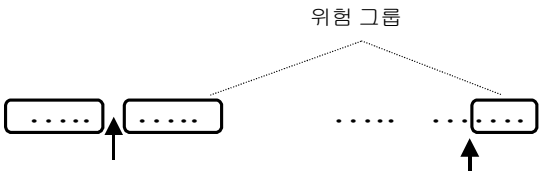
개체들을 정량화한 후 정량화 값들을 기준으로 하여 위험 그룹의 경계 즉 분할을 정하는 방법이 필요하다. 이를 위하여 사람이 적당한 양의 개체들을 위험 그룹으로 선정할 수도 있고 메트릭 값들의 분포에 따라 자동 클러스터링 방법을 사용할 수도 있다. 이 두가지 형태의 방법에 따라 HMM을 HMM-HC(HMM-Human Clustering)와 HMM-AC(HMM-Automatic Clustering) 모델로 나누었다.

HC 형태의 의미는 사람이 일정 개체 집합을 위험 그룹으로 선정하는 모델로, 복잡도가 큰 일정한 양(예를 들면 전체의 10%)의 개체 집합을 위험 집합으로 선정할 수도 있고 복잡도 값으로 정렬된 개체들 중 복잡도 값이 급격히 커지는 부

2) Z: 모든 정수의 집합, R: 모든 실수의 집합
3) 본 논문에서 SDL 메트릭 정의에 관련된 용어들과 배경 이론에 대한 자세한 설명은 [3]을 참조하기 바란다.

분을 경계로 하여 위험 그룹을 선정할 수도 있다. 복잡도 분포를 보고 사람이 위험 그룹을 판단하려면 HMM-HC를 구현한 CASE 도구에서 분석을 위한 그래프나 표 등의 도구를 제공하는 것이 바람직하다. [16]은 몇 개의 통신 시스템을 분석한 결과 위험 집합의 크기는 4-6%인 결론을 얻었다. 하지만 본 논문에서 실험 시 HMM-HC에서 사용한 위험 집합의 크기는 실험의 용이성과 Type II 오류를 줄이는 모델이 적당하다는 관점에서 10% 정도로 하였다.

<그림 2>는 HMM-AC와 HMM-HC를 적용한 경우의 클러스터링 결과를 나타낸 것이다. HMM-HC는 복잡도 분포를 고려않고 사람이 일정한 위험 그룹의 크기를 결정한 경우이다. 점들은 각 개체의 복잡도 값을 의미하며 점 사이의 간격은 복잡도 값의 차이를 나타내고 복잡도는 오름차순으로 나열되었다고 가정한다. 화살표는 결정된 분할이며, 이는 두 모델의 차이점을 잘 나타낸다. 만약 그림과 같이 HMM-AC로 결정한 분할이 전체 개체들의 반 정도를 위험 그룹으로 선정한다면 그림의 HMM-HC의 분할 결정과 같이 복잡도 값이 큰 상위 몇 개의 개체를 위험 그룹으로 선정하는 것이 더 적절할 것이다.



(a) HMM-AC의 분할 결정 (b) HMM-HC의 분할 결정

<그림 2> HMM-AC 대 HMM-HC

HMM-AC에서는 자동 클러스터링 방법을 사용하며, 클러스터링을 수행할 목표값들은 개체들의 혼성 복잡도 값들이다. 알고리즘은 K-means 클러스터링 알고리즘 형태를 사용하며, 목표 클러스터의 수는 두개이다. 본 알고리즘의 목적은 설계 개체들을 복잡도에 따른 오름차순으로 정렬하였을 때 많은 변화를 일으키며 커지는 부분을 찾아 그보다 큰 복잡도를 갖는 집합을 위험 집합으로 선정하는 것이다. 복잡도 값들을 ce, 각 클러스터를 CL_i, 각 클러스터의 평균을 m_i라 할 때 식 (5)를 사용하는 전통적인 알고리즘은 두 클러스터의 복잡도 값이 크게 차이가 나지 않는 분포에서는 적절한 분할을 찾지 못하는 경우가 있었다. 즉 정렬된 복잡도 리스트에서 약간의 복잡도 값 증가로 생긴 분할이 존재한다면 결과 분할 위치는 중간으로 모이는 문제점이 있다. 이를 방지하기 위해 식 (6)를 에러식으로 하여 이를 최소로 하는 분할을 찾는 클러스터링을 사용하였다.

$$\sum_{i=1}^{2} \sum_{ce \in CL_i} | ce - m_i |^2 \qquad (5)$$

$$\sum_{i=1}^{2} \frac{\sum_{ce \in CL_i} | ce - m_i |^2}{| CL_i |} \qquad (6)$$

HMM을 CASE 도구에 구현한다면 HMM-AC의 결과를 사람에게 보여주고 이를 토대로 HMM-HC를 이용한 사람의 위험 그룹 결정이 바람직하다.

4. 모의 실험

4.1. 제작 데이터 집합

제안 모델의 유용성을 검증하기 위해 훈련 데이터 집합을 사용하는 기존 분류 모델 중 우수하다고 알려진 BPM과 예측 정확도를 비교하는 실험을 행하였다. 설계 개체 유형으로는 시스템 분석 단계에서의 블록을 사용하였다. 블록을 정량화하는 메트릭 벡터는 (BRS, EBS, EBC, BP, BS, BR)이며[3] 이는 BPM의 입력이다. EBS는 IBS와 OBS의 합이며, EBC는 IBC와 OBC의 합이므로 BPM은 HMM에 사용된 입력보다 BRS 메트릭 정보를 더 사용한다. 선행 연구 [11]과 유사한 제약 조건을 가진 데이터 집합을 제작하였으며 각 블록의 위험도는 SDL을 이용한 통신 소프트웨어 시스템의 설계 경험이 있는 두 명의 소프트웨어 공학자에 의해 결정되었다.

훈련 데이터 집합은 BPM의 학습에 필요하며 훈련 데이터의 형태는 (BRS, EBS, EBC, BP, BS, BR, FP)로 적용 데이터에 FP가 첨가된 형태이다. FP는 BPM의 출력값으로 1은 위험 개체를 0은 비위험 개체임을 나타낸다. 500개의 블록들

의 데이터를 생성한 후 FP 값을 결정하여 이 중 300개를 선정해 훈련 데이터 집합으로 하고 나머지 200개를 검증 데이터 집합으로 하였다. 이를 훈련데이터집합1, 검증데이터집합1이라 하며 이를 총칭해 데이터집합1이라 한다.

데이터집합1은 일반적인 FP를 결정하는 판단 기준의 경계 부분에 많은 유사한 블록들이 존재하였다. 그러므로 위험 블록과 비위험 블록을 나누는데 많은 판단의 어려움이 있었고 서로 유사한 입력 메트릭 값 분포를 이루는 블록들이 근소한 차이에 의해 서로 다른 그룹에 속하게 되었다. 따라서 위험 그룹과 비위험 그룹간의 차이를 데이터집합1보다 크게 하여 훈련데이터집합2, 검증데이터집합2를 갖는 데이터집합2를 제작하였다. 이는 데이터집합1에서 판단이 애매했던 블록들의 입력 값들을 제약 조건을 만족하는 범위 내에서 고쳐 애매성을 없앤 것이다. BPM 훈련에 사용되는 여러 인자들은 훈련 데이터 자체의 Type I, Type II 오류를 측정하여 좋은 성능을 보이는 인자들을 선택하였으며 학습률을 0.05, 운동량 변화율을 0.2로 하였다.

데이터집합1, 데이터집합2 외에 혼성 메트릭 형태의 상관성 실험과 HMM의 타당성 실험에 사용할 데이터 집합을 제작하였다. 이들은 블록수가 10, 50, 100, 300, 500, 1000, 3000개가 되도록 7개의 파일로 제작하였다.

4.2. 혼성 메트릭 형태의 상관성 실험

HMM 사용 시 고려할 점 중 하나는 두 혼성 메트릭 형태인 가중합과 가중곱 중 어느 형태를 사용하는 것이 효율적이냐에 관한 것이다. 본 실험에서는 두 형태의 상관성을 판단하기 위해 두 형태의 메트릭을 사용하는 HMM-HC의 결과를 비교하였으며 <표 1>은 그 결과이다.

<표 1> 혼성 복잡도 메트릭 상관성 비교

입력 집합	10	50	100	300	500	1000	3000
위험 블록수	1	5	10	30	50	100	300
일치 블록수	1	4	8	25	43	82	246
상관 계수	0.859	0.859	0.852	0.861	0.866	0.866	0.866

<표 1>의 위험 블록수는 가중합과 가중곱 메트릭 즉 C1과 C2를 사용한 HMM-HC가 선정한 위험 블록수이며 이는 입력수의 10%로 고정하였다. 일치 블록수는 두 방법에 의해 선정된 위험 블록 중 일치하는 것들의 수이다. 상관 계수는 입력 집합에 속하는 각 블록들의 C1과 C2 값들의 상관 계수이다. 실험 결과는 선정 블록들의 80%를 조금 넘는 수가 일치하였으며 상관 계수는 0.86 정도가 되었다. 두 메트릭 형태가 매우 상관성이 높다고 할 수는 없지만 어느 정도의 상관성이 있다.

두 메트릭 형태가 선정한 위험 블록들을 비교한 결과 매우 높은 복잡도 값을 갖는 블록들은 거의 일치함을 발견하였다. 상위 5%에 속하는 메트릭들은 90% 정도 일치하였다. 이는 HMM-HC 사용 시 위험 그룹의 크기를 작게 하는 경우에는 두 메트릭 형태 중 어느 것을 사용해도 비슷한 결과를 낸다는 의미이다.

4.3. HMM의 타당성 실험

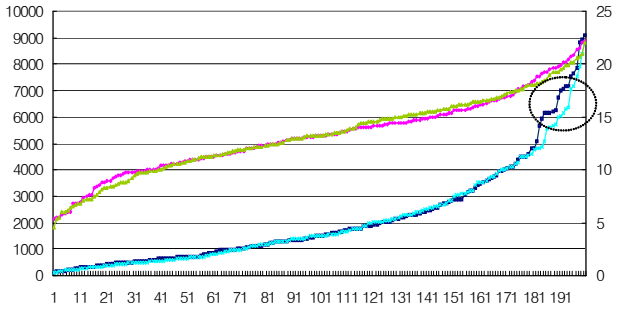
HMM이 모의실험에서 사용하는 데이터 집합에 대해 의미 있는 출력을 내는 지에 대한 타당성 실험을 하였다. 모델의 타당성은 모델이 선정하는 위험 그룹의 크기에 초점을 맞추었다. 타당성 실험 데이터 집합은 전체 데이터 생성에 같은 제약 조건을 둔 균등 분포에 기초하므로 입력 블록수가 증가하면 선정하는 위험 블록수가 증가해야 한다. 또한 선정되는 위험 그룹의 크기는 적절해야 한다.

HMM-HC는 사람이 위험 그룹의 크기를 결정하므로 타당성에 문제가 없다. 따라서 HMM-AC가 적절한 결과를 내는가가 중요하다. <표 2>는 타당성 실험 데이터 집합과 두개의 검증 데이터 집합을 입력으로 하여 C1, C2를 사용하는 HMM-AC에 의해 선정된 위험 블록수 결과이다. C2 결과가 C1의 결과보다는 작지만 너무 많은 위험 블록들을 선정한다.

<표 2> HMM-AC 타당성 실험 결과

입력 집합	10	50	100	300	500	1000	3000	검증1	검증2
C1	6	26	49	142	232	467	1432	95	92
C2	4	12	22	68	118	211	1432	50	41

HMM-AC 결과를 보다 자세히 분석해보기 위해 두개의 검증 데이터 집합의 C1과 C2의 그래프를 <그림 3>에 나타내었다. 이 그림은 검증데이터집합1과 검증데이터집합2의 C1과 C2값들을 오름차순으로 정렬한 것이다. 좌측의 y축은 C2값을 우측의 y축은 C1값을 나타낸다. 위의 두개의 그래프는 두 검증 데이터 집합에 대한 C1의 결과이며 밑의 그래프 중 복잡도 큰 블록들에서 급격히 상승하는 그래프가 검증데이터집합2에 대한 C2 결과이고 다른 것이 검증데이터집합1에 대한 C2 결과이다.



<그림 3> 검증 데이터 집합의 복잡도 분포

그림을 보면 C2가 C1보다 전체적으로 가파른 증가를 보이며 위험 그룹이 존재하는 우측 끝부분에서도 C2가 급격히 증가한다. 이는 C2가 가중곱의 형태이기 때문에 혼성 메트릭을 형성하는 기본 메트릭값의 증가에 대해 증가 속도가 빠른 이유, 즉 변별력이 뛰어나기 때문이다. 그러므로 혼성 복잡도의 증가 분포 그래프를 분석하여 위험 그룹을 선정하는 HMM-HC는 가중곱 메트릭 분포를 이용하는 것이 바람직하다.

위험 그룹과 비위험 그룹과의 값의 차이를 상대적으로 명확하게 만든 검증데이터집합2에 대한 C2 그래프가 다른 것보다 증가 부분이 가파르다. 갑자기 증가하는 부분은 x축의 180 부분과 190 부분 두 곳에서 발견된다. HMM-HC는 이 부분을 그래프를 보고 사람이 선정하는 것이므로 180 부분을 선정할 것이고 HMM-AC의 목표는 이 부분을 정확하게 알고리즘에 의해 자동으로 선정하는 것이다. 그러나 <표 2>를 보면 HMM-AC는 41개의 위험 블록을 선정하므로 이 부분을 찾아내지 못한다.

이 문제점의 이유를 분석한 결과 클러스터링 알고리즘에서 식 (5)를 쓴 것이 주된 이유였다. 즉 입력 데이터의 범위가 매우 넓고 데이터가 골고루 분포되어 있어 (5)를 사용한 알고리즘의 결과 분할 위치가 정렬 데이터의 중간 부분으로 몰리는 성질 때문이었다. 따라서 작은 데이터 증가에도 보다 민감하게 반응할 수 있는 식 (6)을 사용하는 클러스터링을 수행하였으며 <표 3>은 그 결과이다. 수정된 HMM-AC는 C1을 사용하였을 때의 결과는 기존의 HMM-AC 결과와 유사하다. 이는 <그림 3>에서 보듯이 실험 데이터에 대해 C1이 급격한 변화를 보이는 곳이 없기 때문이다. 그러나 C2를 사용하였을 때는 기존 결과보다 매우 향상된 결과를 나타내었다. <표 3>의 결과를 보면 타당성 실험 데이터 집합에 대해 1000개의 입력 집합에 대해서는 불안정한 결과를 내지만 다른 입력 집합에 대해서는 약 15% 정도의 비교적 안정된 결과를 냄을 알 수 있다. HMM-AC의 관심 입력 집합인 검증데이터집합2에 대해 18개의 위험 블록을 선정함으로 수정된 HMM-AC가 <그림 3>의 180 부분의 가파른 상승 부분을 찾아냈음을 알 수 있다.

<표 3> 수정된 HMM-AC 타당성 실험 결과

입력 집합	10	50	100	300	500	1000	3000	검증1	검증2
C1(가중합)	6	27	49	122	215	422	1378	88	114
C2(가중곱)	3	7	10	56	94	84	465	42	18

실험 결과를 보면 수정된 HMM-AC가 기존 HMM-AC보다 타당성이 있으므로 본 논문의 HMM-AC는 수정된 HMM-AC를 의미한다. 결과적으로 HMM의 타당성 실험에서는 HMM-HC와 함께 가중곱 메트릭을 사용하는 HMM-AC가 타당하다는 결론을 얻었다.

4.4. 모델의 예측 정확도 비교

예측 정확도 비교를 위해 Type I, Type II 오류 정보를 사용하였다.

Type I 오류정보: 비위험 개체를 위험 개체로 선정한 수 / 비위험 개체수

Type II 오류정보: 위험 개체를 비위험 개체로 선정한 수 / 위험 개체수

Type II 오류가 시스템의 완성 시기를 늦추고 품질을 저하시키는 원인이 되므로 Type II 오류를 낮추는 것이 Type I 오류를 낮추는 것보다 중요하다.

<표 4>는 BPM, HMM의 예측 결과들이다. HMM-AC는 C2 사용 HMM-AC 결과이다. HMM-HC의 FP는 위험 블록 집합을 의미하며 |FP|는 사람이 지정한 위험 그룹의 크기이다. 검증 데이터 집합 두개는 모두 실제로 23개의 위험 블록들을 가진다. 표에는 두개의 훈련 데이터 집합과 두개의 검증 데이터 집합에 대해 네개의 결과를 나타내었지만 훈련데이터집합2로 훈련시킨 모델에 검증데이터집합1을 적용시키는 것은 의미가 없다. 왜냐하면 위험 그룹과 비위험 그룹의 차이가 명확한 데이터로 훈련된 모델이 차이가 매우 애매한 입력 집합에 대해 정확한 판단을 내리기 어렵기 때문이다. 표에서도 이 경우에 BPM은 매우 많은 오류를 냄을 볼 수 있다.

<표 4> BPM, HMM 선정 오류 결과

모델 \ 판별 오류		검증1		검증2	
		Type I	Type II	Type I	Type II
BPM	훈련1	4/177	4/23	7/177	0/23
	훈련2	6/177	11/23	0/177	0/23
HMM-AC		24/177	5/23	0/177	5/23
HMM-HC (|FP|=23)	C1	8/177	8/23	2/177	2/23
	C2	9/177	9/23	3/177	3/23
HMM-HC (|FP|=30)	C1	13/177	6/23	7/177	0/23
	C2	16/177	9/23	8/177	1/23
HMM-HC (|FP|=23,BRS고려)	C1	5/177	5/23	1/177	1/23
	C2	6/177	6/23	2/177	2/23

HMM-AC는 타당성 실험에서 살펴보았듯이 검증데이터집합1에 대해서는 가파른 상승 부분이 없으므로 적당한 위험 그룹의 크기를 찾지 못하여서 불만족스러운 결과가 나온다. 그러나 가파른 상승 부분이 있는 검증데이터집합2에 대해서는 18개의 위험 블록을 선정하였다. Type I 오류가 0개 이므로 18개 모두 맞는 위험 블록들을 선정하였음을 알 수 있으나 실제 위험 블록수 23개보다 적은 수의 블록들을 선정함으로써 Type II 오류가 5개 발생했음을 알 수 있다. BPM이 의미없는 실험 결과인 훈련데이터집합2-검증데이터집합1의 결과를 제외한 세가지 결과에서 위험 블록들을 23, 30개를 선정하였으므로 이들과의 비교를 위해 HMM-HC의 |FP|를 23, 30으로 하였다. 결과는 C1이 C2보다 더 좋은 성능을 보였으며, 검증데이터집합2에 대해 검증데이터집합1보다 훨씬 높은 적중률을 보였다.

BPM보다 적중률이 낮은 이유를 분석한 결과 BRS를 혼성 복잡도 메트릭 제작에 첨가하지 않았다는 것과 훈련 데이터 및 검증 데이터 제작 시 IBS, OBS, IBC, OBC 분포를 함께 고려하지 않았다는 점, 그리고 나머지 문제점은 조합 메트릭이 부분 메트릭들의 성질들을 모두 가질 수 없다는 혼성 메트릭 자체의 문제점이었다. 그러므로 혼성 메트릭 값이 높으면서 BRS 값이 상위 50% 안에 드는 것들만을 23개 선정해보았다. 그 결과 BRS를 고려하지 않은 HMM-HC 결과보다 상당히 좋은 결과가 나왔다. 실험 결과를 보듯이 BRS를 고려한 HMM-HC는 BPM에 근접하는 성능을 보였다.

5. 결 론

수백명의 개발 인력이 수년간 투여되는 대형 소프트웨어 개발 프로젝트에서 초기 위험도 예측 모델은 시스템 개발비용을 낮추는데 큰 몫을 하고 있다. 신경망, 판별 분석, 분류 트리 등을 사용한 기존의 모델들이 단순히 위험 그룹과 비위험 그룹을 나누는 분류 모델이었던데 비해 본 논문에서는 각 개체의 위험도 비교가 가능한 새로운 예측 모델인 HMM을 제안하였다. HMM은 훈련 데이터 집합을 필요로 하지 않는 일반화된 소프트웨어 품질 모델이지만 스칼라 형태의 조합 메트릭에 의존한다는 문제점도 있다. 메트릭 제작을 위해 검증된 혼성 메트릭 제작 프레임워크를 사용하였으며 클러스터링 방법에 따라 모델을 HMM-HC와 HMM-AC로 분류하였다. 실제 프로젝트에 적용 시 위험 그룹의 크기를 입력 집합의 약 10%로 하여 HMM-AC와 함께 HMM-HC를 사용하는 것이 바람직하다.

타당성 실험을 통해 HMM-AC는 가중곱 메트릭을 사용하는 것이 적당하다는 결론을 내렸다. HMM의 취약점이라 예상된 예측 정확도 성능 평가를 위해 기존 모델들 중 우수하다고 알려진

BPM과의 성능 비교를 모의실험을 통해 수행하였다. 실험 결과 BRS를 고려치 않은 HMM은 BPM에 떨어지는 성능을 보였지만 BRS를 고려한 HMM은 BPM에 근접한 성능을 보였다.

참 고 문 헌

[1] J. Tian, A. Nguyen, C. Allen and R. Appan(2001). Experience with identifying and characterizing problem-prone modules in telecommunication software systems. *J. Systems Software,* 57, pp. 207-215.

[2] N. Ohlsson and H. Alberg(1996). Prediction FaultProne Software Modules in Telephone Switches. *IEEE Trans. Software Eng.,* 22(12), pp. 886-894.

[3] 홍의석, 홍성백, 김갑수, 우치수(1997). SDL 설계 복잡도 메트릭 집합. 정보과학회 논문지(B), 24(10), pp. 1053-1062.

[4] 홍의석, 김태균(2001). 혼성 메트릭을 이용한 소프트웨어 개체 복잡도 정량화 기법. 정보처리학회 논문지, 8-D(3), pp.233-240.

[5] 홍의석, 정명희(2000). SDL 메트릭 집합의 분석적 검증. 정보처리학회 논문지, 7(4), pp. 1112-1121.

[6] T. Khosgoftaar and E. Allen(1996) Early Quality Prediction: A Case Study in Telecommunications. *IEEE Software,* 13(1) pp. 65-71.

[7] T. Khoshgoftaar and D. Lanning(1995). A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase. *J. Systems Software,* 29, pp. 85-91.

[8] L. Briand, J. Daly, V. Porter and J. Wüst(1998). Predicting fault-prone classes with design measures in object-oriented systems. *Proc. Int'l Symp. Software Reliability Engineering,* pp. 334-343.

[9] T. Khoshgoftaar, B. Allen and J. Deng(2002). Using regression trees to classify fault-prone software modules. *IEEE Trans. Reliability,* 51(4), pp. 455-462.

[10] K. Emam, S. Benlarbi, N. Goel and S. Rai(2001). Comparing case-based reasoning for predicing high risk software components. *J. Systems Software,* 55, pp. 301-320.

[11] 홍의석(2003). GAM: 대형 통신 시스템을 위한 위험도 예측 모델. 컴퓨터교육학회 논문지, 6(2), pp.33-40.

[12] W. Zage and D. Zage(1993). Evaluating Design Metrics on Large-Scale Software. *IEEE Software,* pp.75-80.

[13] W. Agresti and W. Evanco(1992). Projecting Software Defects From Analyzing Ada Designs. *IEEE Trans. Software Eng.* 18(11).

[14] R. Selby and V. Basili(1991). Analyzing error-prone system structure. *IEEE Trans. Software Eng.,* 17(2), pp. 141-152.

[15] N. Fenton(1994). Software Measurement: A Necessary Scientific Basis. *IEEE Trans. Software Eng.,* 20(3), pp. 199-206.

[16] G. Lecall, M. Adam, H. Derriennic, B. Moreau and N. Valette(1990). Studies on measuring software. *IEEE J. Selected Areas Commun.,* 8(2), pp. 234-245.

홍 의 석

1992 서울대학교 계산통계학과 (학사)

1994 서울대학교 계산통계학과 (석사)

1999 서울대학교 전산과학과 (박사)

1999 ~ 2002 안양대학교 디지털미디어학부 교수

2002 ~ 현재 성신여자대학교 컴퓨터정보학부 교수

관심분야: 소프트웨어공학, 소프트웨어 품질예측 모델, 웹기반 컴포넌트 응용 기술 등

E-Mail: hes@sungshin.ac.kr