가상 훈련 데이터를 사용하는 소프트웨어 품질 분류 모델

Software Quality Classification Model using Virtual Training Data

홍의석
성신여자대학교 컴퓨터정보학부

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

요약

소프트웨어 개발 프로세스의 초기 단계에서 결함경향성이 많은 모듈들을 예측하는 위험도 예측 모델은 프로젝트 자원할당에 도움을 주어 전체 시스템의 품질을 개선시키는 역할을 한다. 설계 복잡도 메트릭에 기반을 둔 여러 예측 모델들이 제안 되었지만 대부분 훈련 데이터 집합을 필요로 하는 모델들이었고 훈련 데이터 집합을 보유하고 있지 않은 대부분의 개발 집단들은 이들을 사용할 수 없다는 문제점이 있었다. 본 논문에서는 잘 알려진 감독형 학습 모델인 오류 역전파 신경망 모델에 SDL 시스템 명세를 정량화하여 적용한 예측 모델을 개발하였으며, 기존 학습 모델들의 문제점을 해결하기 위해 이 모델을 여러 제약조건을 가지고 만든 가상 훈련데이터집합으로 학습시켰다. 제안 모델의 사용가능성을 알아보기 위해 몇가지 모의 실험을 수행 하였으며, 그 결과 제안 모델이 훈련 데이터 집합이 없는 개발 집단에서는 실제 데이터로 훈련된 예측 모델의 대안으로 사용될 수 있음을 보였다.

■ 중심어 : | 분류 모델 | 소프트웨어 품질 | 가상 훈련 데이터 |

Abstract

Criticality prediction models to identify most fault-prone modules in the system early in the software development process help in allocation of resources and foster software quality improvement. Many models for identifying fault-prone modules using design complexity metrics have been suggested, but most of them are training models that need training data set. Most organizations cannot use these models because very few organizations have their own training data. This paper builds a prediction model based on a well-known supervised learning model, error backpropagation neural net, using design metrics quantifying SDL system specifications. To solve the problem of other models, this model is trained by generated virtual training data set. Some simulation studies have been performed to investigate feasibility of this model, and the results show that suggested model can be an alternative for the organizations without real training data to predict their software qualities.

■ keyword : | Classification Model | Software Quality | Virtual Training Data |

* 본 논문은 2007년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었습니다.

접수번호 : #080414-004
접수일자 : 2008년 04월 14일
심사완료일 : 2008년 05월 27일
교신저자 : 홍의석, e-mail : hes@sungshin.ac.kr

I. 서 론

소프트웨어 품질은 모든 소프트웨어 개발 집단의 성공을 결정하는 핵심 요소이다. 한정된 프로젝트 예산과 자원을 이용하여 고품질의 소프트웨어를 제작하는 것은 성공적인 프로젝트의 목표이지만 개발 시스템이 매우 커지거나 복잡하게 될수록 이런 목표를 달성하기는 매우 어려워진다. 따라서 소프트웨어 개발 프로세스를 전체적으로 관리할 수 있는 소프트웨어 프로세스 개선이 필요하고 이의 기반이 되는 것은 프로세스와 산물을 정량화하는 소프트웨어 메트릭이다.

분석이나 설계 단계가 소프트웨어 산물의 품질인자(quality factor)에 큰 영향을 미치게 되므로 설계 단계를 정량화 하는 설계 메트릭들도 전체 시스템 개발비용을 낮추는 데 중요한 역할을 하고 있다. 또한 설계 메트릭들을 사용하여 최종 개발물의 품질 인자들을 예측하는 예측 모델의 중요성도 매우 높다. 왜냐하면 품질에 큰 영향을 미치는 핵심 부분들을 선정하고 이에 맞게 한정된 자원들을 잘 분산 배치함으로써 주어진 시간 내에 가장 고수준의 품질을 보장하는 소프트웨어를 제작할 수 있기 때문이다. 이러한 설계 정량화와 예측 모델의 필요성은 결과 산물이 매우 크고 실행 정확성이 요구되는 통신 소프트웨어 같은 실시간 시스템 설계에 더욱 필요하다. 복잡도 메트릭들을 이용한 소프트웨어 품질 예측 모델에 대한 연구들은 최근까지 많이 행해졌으며 주로 관련된 품질인자들은 유지보수성이나 신뢰성, 신뢰성에 영향을 미치는 결함경향성에 대한 것들이었다. 본 연구는 이들 중 결함경향성 예측에 대한 것이다.

한 설계 개체의 위험도란 개체가 구현되었을 때 갖는 결함경향성을 의미한다[1]. 위험도 예측 모델이란 설계 개체를 입력으로 받아 그것이 결함경향 개체인지 아닌지를 판단하는 모델이며, 모델의 입력은 설계 개체들을 정량화 한 메트릭 벡터 형태가 된다. 기존에 제안된 위험도 예측 모델들은 많이 알려진 복잡도 메트릭들로 구성된 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.

이와 같은 기존 모델들의 가장 큰 문제점은 과거 유사한 개발 환경에서 얻은 실제 프로젝트 데이터인 훈련 데이터 집합을 필요로 한다는 것이다. 대부분의 개발 집단은 이러한 훈련 데이터 집합을 보유하고 있지 않으며[2], 설사 보유하고 있다 하더라도 모델을 적용하려는 현재 프로젝트의 참여 인력, 개발 환경이 과거 프로젝트와 매우 유사하여야만 한다는 문제점이 있다. 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 이 같은 문제들을 해결하기 위해 현재 프로젝트의 제약조건들을 고려한 가상 훈련 데이터 집합을 생성해 이를 사용하는 [그림 1]과 같은 훈련 모델을 제안한다. 그림에서 기존의 훈련 모델들은 윗부분에 나타낸 과거 프로젝트 수행 시 훈련 데이터 수집이 반드시 필요하며 제안 모델은 이와 같은 과정이 필요 없다.

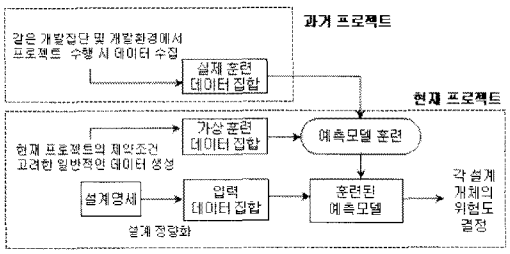

그림 1. 기존 훈련 모델과 제안 모델 비교

위험도 예측 모델은 수천, 수만 LOC 정도의 시스템이 아니라 수십만 LOC 이상의 대형 시스템을 개발하는 경우에 유용하므로 위험도 예측에 관한 연구는 주로 대형 통신 시스템 등을 개발하는 개발 집단을 중심으로 행해져왔다. 제안 모델의 입력 대상은 ITU-T의 표준안으로 널리 사용되고 있는 객체지향 실시간 시스템 명세 언어인 SDL(Specification and Description Language)로 작성한 설계 명세이며, 훈련 알고리즘은 잘 알려진 감독형 학습 알고리즘인 오류 역전파 신경망을 사용한다.

2장에서는 기존에 제안된 위험도 예측 모델들을 살펴보고 그들의 장단점을 논의하며 3장에서는 SDL을 지원하는 역전파 신경망 모델의 구조를 설명한다. 4장에

서는 모의실험에 사용한 데이터 제작기법, 구현된 모델의 최적화 및 타당성 실험, 가상데이터로 훈련시킨 모델의 성능에 대해 설명하고 5장에는 결론에 대해 기술한다.

II. 관련 연구

복잡도 메트릭을 이용한 위험도 예측 관련 연구들은 모두 소프트웨어의 복잡도 메트릭이 프로그램의 오류의 분포와 관련이 있음을, 즉 복잡도가 높은 모듈일수록 오류가 발생할 가능성이 높다는 점을 가정하고 있다. 관련 연구들은 크게 두 가지로 구분할 수 있으며, 첫 번째는 서론에서 기술하였듯이 과거 위험도 데이터를 학습해 현재 프로젝트의 위험도 예측에 사용하는 훈련 모델들이다. 이들은 주로 입력 데이터들을 여러 개의 패턴으로 나누는 패턴 분류 기법들을 사용하며 예로는 판별분석[3], 분류트리[4], 역전파 신경망[5], 로지스틱 회귀분석[6], CBR(Case-Based Reasoning)[7], 유전 프로그래밍[8] 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 앞에서 기술한 훈련 모델의 문제점들을 가지고 있다.

두 번째는 프로그램의 위험도를 추정(estimate)할 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 추정한다. 그러므로 각 개체의 위험도를 서로 비교할 수 있다는 장점이 있다. 물론 소프트웨어 측정 이론의 관점에서 보면 이러한 메트릭들은 비율 스케일(ratio scale)이 아니라 서수 스케일(ordinal scale)이므로 순서 비교만이 가능하지 값을 통하여 몇 배가 더 위험하다는 식의 결론을 내릴 수는 없다[9]. 이 방법은 기존 데이터들을 통한 경험적인 지식이 아니라 하나의 이론적인 복잡도 메트릭 값에 의존한다는 단점이 있다. 품질 인자와 소프트웨어 메트릭 값은 서로 비선형 관계에 있으므로 선형적인 하나의 메트릭 값으로 정량화하는 것은 불가능하다[10]. 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[11]. 스칼라 메트릭으로 위험도를 예측하는 연구로는 Zage의 연구[12], Agresti의 연구[13]와 데이터 바인딩 기법[14] 등이 있다.

모델의 성능을 생각하면 당연히 전자인 훈련 모델이 뛰어나겠지만 앞에서 기술한 바와 같이 대부분의 개발 집단은 훈련 데이터 집합을 가지고 있지 않다. 이를 해결하기 위해 [15]는 설계 개체들의 복잡도 메트릭 벡터들을 입력으로 받아 Kohonen SOM 신경망을 사용하여 입력 집합을 위험 그룹과 비위험 그룹의 클러스터로 나누는 모델을 제안하였다. 하지만 이 모델은 클러스터를 판별하기 위한 사람의 분석 과정이 필요하고, 입력 메트릭 벡터의 분포에 결과가 의존하며, 최적화된 클러스터의 수를 자동 결정하는 적절한 클러스터링 알고리즘이 거의 없다는 문제를 가지고 있다. 그러므로 본 연구의 중요한 목적 중 하나는 개발 집단의 환경에 의존성이 매우 적은 일반적인 훈련 데이터 집합을 제작해서 훈련시킨 모델을 사용함으로써 훈련 데이터 집합이 없는 개발 집단에게 비훈련 예측모델보다는 좋은 성능을 갖는 모델을 제공하는 것이다.

III. SDL기반 역전파 신경망 모델

본 논문에서는 SDL 시스템을 정량화한 SDL 메트릭 집합과 이를 입력으로 하는 훈련 모델로 역전파 신경망 모델을 사용한다.

1. 모델의 입력 메트릭

SDL은 실시간 프로세스들의 동작을 EFSM (Extended Finite State Machine) 개념을 도입하여 표준화한 설계 명세 언어로서 ATM 교환기 시스템과 같은 대형 통신 시스템의 명세서와 설계서를 형식적으로 나타내는데 사용된다. 블록은 프로세스들과 이들의 상호작용으로 이루어진 기능 단위이며, 설계 방법은 EFSM을 작성하고 이를 기초로 하여 블록을 나누는 상향적인 방법과 블록 단위로 시스템을 나누고 이를 EFSM으로 세분화하는 하향적인 방법이 가능하다. 후자가 일반적인 방법이므로 일반적인 설계 단계는 구조 설계 단계와 상세 설계 단계로 나뉜다.

각 설계 단계에서 정량화 가능한 개체는 구조 설계 단계의 블록, 상세 설계 단계의 블록, 상세 설계 단계의 프로세스이다. 본 모델의 입력 개체는 구조 설계 단계의 블록으로 하였다. 이는 나머지 두 개체 형태들을 위한 메트릭들 역시 같은 형태의 메트릭 구조를 따르며 구조설계 단계의 측정치들이 상세 설계 단계의 측정치들보다 유지보수성이나 위험도에 더 많은 영향을 미치기 때문이다.

구조 설계 단계에서 각 블록은 하나의 부시스템을 나타내며 이는 블록 계층화 모델에서 해당 블록을 루트로 하는 부트리로 나타난다. 이러한 부시스템 즉 블록을 가장 상세히 나타내면 후손 끝단 블록들의 자식인 프로세스들과 그들의 상호작용을 나타내는 라우트들 그리고 신호들로 나타낼 수 있다. 이렇게 하나의 블록을 가장 상세히 나타낸 BID(Block Interaction Diagram)를 해당 블록의 FBID(Flat BID)라고 한다.

표 1. 구조 설계 단계의 블록 정량화

종 류	메트릭	정 의
기본 메트릭	IBC,OBC, IBS,OBS	number of Input/Output Channels/Signals of a Block 블록의 입출력 채널/신호수
	BP, BR, BS	number of Processes/Routes/Signals composing a Block 블록의 FBID 내의 프로세스/라우트/신호수
복잡도 메트릭	BRS	Balance Rate of a Subsystem 블록 부시스템의 끝단 블록 수준들의 분산
	EBC	number of External Channels of a Block IBC+OBC
	EBS	number of External Signals of a Block IBS+OBS
	BP,BR,BS	

[표 1]은 구조 설계 단계의 블록을 정량화 하기위한 기본 메트릭들과 그들을 이용하여 정의한 복잡도 메트릭들을 나타낸 것이다. EBC, EBS는 블록의 입출력 양을 나타내는 외부 복잡도 메트릭들이다. BP, BS, BR은 블록의 내부 복잡도의 양을 나타내는 내부 복잡도 메트릭들이다. BRS는 블록의 부시스템의 구조 관련 복잡도를 나타내는 복잡도 메트릭이다. 잘 균형화된 부시스템일 수록 끝단 블록들의 수준 분포는 밀집된 분포를 보이기 때문에 BRS는 작아진다. BRS는 다른 기본 메트릭들이 가지는 성질로 표현할 수 없는 복잡도 메트릭이므로 모델의 입력에 추가하였다.

메트릭 벡터를 이루는 각 원소들은 서로 독립적이어야 한다. 외부 복잡도와 내부 복잡도가 완전히 독립적인 것은 아니지만 외부 입출력이 많더라도 내부 복잡도는 작아질 수 있으므로 하나의 블록을 정량화하는 메트릭 벡터는 (BRS, EBS, EBC, BP, BS, BR)로 하였다.

구조 설계 단계의 블록을 입력으로 하지 않고 상세 설계 단계의 블록이나 프로세스를 입력으로 한다고 해도 역시 각각에 해당하는 기본 메트릭들을 중심으로 하여 입력 메트릭 벡터를 정의할 수 있다.

2. 모델의 구조

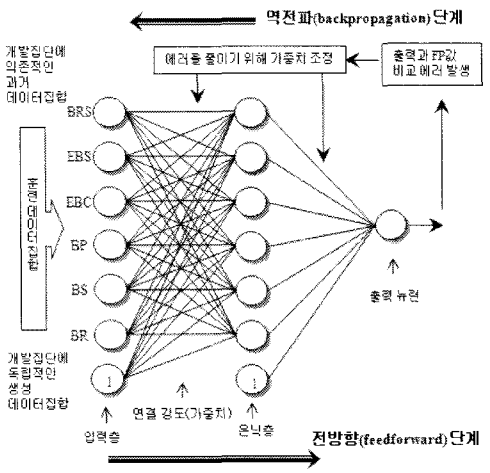

그림 2. SDL기반 역전파 신경망 모델 구조

[그림 2]는 구조 설계 단계 블록의 위험도 정보를 학습하고 예측하기 위한 오류 역전파 신경망의 구조이다.

역전파 신경망을 사용한 이유는 훈련데이터를 사용하는 감독형 알고리즘 중에서 성능이 좋다고 알려져 있는 가장 일반적인 감독형 다중 신경망이기 때문이다.

입력층은 블록의 메트릭 벡터를 구성하는 각각의 메트릭들이 입력 뉴런들이 된다. 문제의 종류에 따라 역전파 신경망은 여러개의 은닉층을 갖기도 하지만 하나의 은닉층으로 대부분의 문제들을 충분히 해결할 수 있음이 증명되어 있으므로[16] 하나의 은닉층을 사용하였다. 또한 은닉층의 뉴런수는 입력층의 뉴런수와 같게 하였다. 출력은 출력 뉴런이 하나일 경우라도 연속적인 출력값을 갖게 한다면 그 범위를 얼마나 세분화하는가에 따라 여러개의 클래스로 구분이 가능하다. 그림의 모델은 하나의 출력 뉴런의 출력값으로 두 개의 클래스(위험개체와 비위험 개체)를 판단한다. FP는 위험 개체는 1, 비위험 개체는 0을 가지며 훈련 데이터 집합에 속해있는 각 블록들의 실제 위험도 값이다.

모델의 훈련 단계에서는 훈련 데이터 집합에 있는 각 블록의 메트릭 벡터를 입력으로 받아 전방향 단계, 계산된 에러의 역전파 단계, 가중치 수정 단계를 반복적으로 거쳐 훈련이 완료된 모델이 된다. 활성화 함수는 이진 시그모이드(sigmoid) 함수를 사용하였으며, 운동량 방법을 학습에 사용하였다. 학습 시 여러 개의 파라미터 조율 과정을 거쳐 학습률을 0.05, 운동량 변화율을 0.2로 하였으며, 일반적인 역전파 알고리즘 학습과 같이 가중치들은 -0.5~0.5의 난수값으로 초기화 하였다. 훈련 데이터로는 그림에 나타내었다시피 개발 집단에 의존적인 과거 실제 데이터 또는 개발 집단에 독립적인 가상 데이터를 사용할 수 있다. 이와 같은 예측 모델의 구조는 기존의 역전파 신경망을 사용한 모델들[5][15]과 유사하지만 기존 모델과 제안 모델의 차이점은 훈련 프로세스에서 기존 모델들은 과거 프로젝트 데이터를 사용하며 제안 모델은 가상 데이터를 사용한다는 점이다. 즉, 제안 모델은 [그림 1]에서 나타낸 모델의 훈련 및 예측 프로세스에서 과거 프로젝트 정보가 전혀 필요없이 현재 프로젝트 정보만으로 사용될 수 있다.

훈련 단계가 끝나고 학습된 신경망은 위험도 예측 모델로 사용된다. 위험도를 측정하고자하는 블록의 입력값을 넣으면 출력 결과로 입력 블록의 위험도를 판단하게 된다. 앞에서 기술한 것과 같이 대부분의 개발 집단은 과거 훈련 데이터 집합을 보유하고 있지 않으므로 실제 데이터로 훈련시키는 기존 모델들은 일반적인 개발 집단에서는 사용할 수 없다. 그러므로 예측 성능은 어느 정도 떨어지지만 개발 집단에 독립적인 일반화된 가상 데이터 집합을 제작해 훈련 데이터 집합으로 사용하는 일반화된 모델이 필요한 것이다.

IV. 모의 실험

1. 실험 데이터 생성

가상 데이터 생성 방법과 실험에 사용할 훈련 데이터 집합이나 검증 데이터 집합의 종류들을 정의한다.

1.1 데이터 가정 및 제약 조건

설계 정보들에 대한 특별한 제약이나 소프트웨어 관련 메트릭들의 통계 분포가 알려진 것이 없기 때문에 데이터 생성은 여러 제약 조건들을 두어 그것을 만족하는 난수값들을 택하였다.

입력 메트릭들의 최대, 최소값들을 각각 종류별로 설정하고 메트릭 값을 이 메트릭이 가질 수 있는 범위의 난수를 발생시켜 생성한다. 예를 들어 입력 개체 집합의 EBC 값이 1부터 10까지라면 입력 개체의 EBC 값들은 이 범위에 속하는 난수 값이 된다. 난수를 사용하므로 생성 데이터는 균등 분포의 성질을 갖는다. 이 문제점을 최소화 하기 위해 몇 가지 가정과 제약 조건을 사용한다. 가정과 제약 조건 기술 시 메트릭 M의 최소와 최대값을 MinM, MaxM이라 표기한다.

- 측정치 범위의 가정
  - $0.0 \le BRS \le 4.0$, $1 \le EBS \le 25$, $1 \le EBC \le 10$
  - $1 \le BP \le 20$, $2 \le BS \le 55$, $2 \le BR \le 30$

  : 한 블록을 형성하는 프로세스 수 BP의 최대값을 20으로 정하였다. 그 기준에 맞추어 예상되는 여러 메트릭값의 범위들을 정하였다. 모든 프로세스는 하나 이상의 입출력 라우트들과 입출력 신호들을 갖는 것을 가정한다. 그러므로 BR과 BS의 최소값이

2가 되며 이 경우는 IBC와 OBC가 각각 1인 경우이다.

- IBS > 0, OBS > 0, IBC > 0, OBC > 0
  : 입출력 값들의 곱 형태인 가중곱 외부 복잡도가 0이 되지 않게 한다.
- IBS + OBS = EBS, IBC + OBC = EBC
  : EBS, EBC를 기준으로 해서 IBS, OBS, IBC, OBC를 선정한다.
- IBS ≥ IBC, OBS ≥ OBC
  : 하나의 입출력 채널이 여러 개의 신호를 실을 수 있으며, 적어도 하나 이상의 신호를 전달한다.
- BS ≥ EBS, BR ≥ EBC
  : 블록의 FBID 내부의 신호는 외부에서 들어오고 나가는 신호들에다 내부에서 정의한 신호 정보들을 합한 것이다. FBID의 라우트들 역시 외부의 채널들과 내부에서 새로이 정의되는 라우트들 정보들을 합한 것이다.
- BS ≥ BR, BR ≥ 2BP, BS ≥ 2BP
  : 한 라우트에 여러 신호가 실릴 수 있다. 그리고 측정치 범위의 가정으로부터 모든 프로세스는 입출력 라우트와 신호들이 있으므로 BR과 BS의 최소값을 2BP로 나타낼 수 있다.
- $\frac{EBC}{MAXEBC} - 0.1 < \frac{EBS}{MAXEBS} < \frac{EBC}{MAXEBC} + 0.1$
  : EBC와 EBS는 블록이 외부와 상호 작용하는 다른 형태의 정보량이므로 서로 상관성이 있다.
- $\frac{BP}{MAXBP} - 0.2 < \frac{BS}{MAXBS} < \frac{BP}{MAXBP} + 0.2$,
  $\frac{BP}{MAXBP} - 0.2 < \frac{BR}{MAXBR} < \frac{BP}{MAXBP} + 0.2$
  : BP, BS, BR은 서로 상관성이 있으나 EBS와 EBC의 상관성보다는 작다고 예상되기 때문에 유사성 범위를 늘렸다.
- $if \frac{BP + BS + BR}{MAXBP + MAXBS + MAXBR} < 0.1$
  $then\ BRS = 0.0$
  : SDL 블록 계층화 모델에서 하위 수준 블록들일수록 전체 시스템을 상세하게 표현하므로 가장 바람직한 예측 모델의 입력 집합은 끝단 블록 집합이다. 그러므로 실제 프로젝트에서 사용하는 입력 개체들에는 끝단 블록들이 많이 존재할 것이고 끝단 블록들의 BRS값은 0.0이므로 블록의 내부 복잡도를 나타내는 요소들이 매우 작으면 BRS값을 0.0으로 하였다.

1.2 제작 데이터 집합

훈련 데이터 집합은 예측 모델의 학습에 필요하며 이는 이들 모델의 성능을 결정하는 가장 중요한 요인이 된다. 훈련 데이터의 형태는 (BRS, EBS, EBC, BP, BS, BR, FP)로 모델의 입력 데이터에 위험도 여부를 나타내는 데이터인 FP가 첨가된 형태이다. 300개 블록들의 데이터를 생성한 후 FP 값을 결정하여 훈련데이터집합으로 하였다. 각 블록의 위험도는 SDL을 이용한 통신 소프트웨어 시스템의 설계 경험이 있는 두명의 소프트웨어 공학자에게 SDL 복잡도 메트릭 집합을 설명한 후 그들의 경험에 의해 결정하였다. 이와 같이 생성된 훈련 데이터 집합을 훈련데이터집합1이라 한다.

훈련데이터집합1은 FP를 결정하는 판단 기준의 경계 부분에 많은 유사한 블록들이 존재하였으므로 서로 유사한 입력 메트릭 값 분포를 이루는 블록들이 근소한 차이에 의해 서로 다른 그룹에 속하게 결정되었다. 따라서 위험 그룹과 비위험 그룹간의 차이를 훈련데이터집합1보다 크게 한 훈련 데이터 집합을 제작하였다. 이는 훈련데이터집합1에서 판단이 애매했던 블록들의 입력 값들을 제약 조건을 만족하는 범위 내에서 고쳐 애매성을 없앤 것이다. 이를 훈련데이터집합2라 한다.

검증 데이터 집합은 훈련된 모델의 성능을 검증하기 위한 데이터 집합이다. 제안 모델의 훈련 데이터는 가상 데이터를 사용하지만 검증 데이터는 현재 프로젝트의 실제 데이터를 사용하여야한다. 하지만 본 연구에서는 실제 프로젝트 데이터를 구하지 못하였으므로 검증 데이터 역시 가상 데이터 집합을 제작하여 사용하는 모의실험을 수행하였다. 이 집합에 속하는 블록의 위험도는 훈련 데이터의 위험도와 같은 방법으로 정한 후 다른 한명의 연구자가 수정하여 최종 결정하였다. 이는 훈련 데이터와 검증 데이터의 위험도 분포를 다르게 하여 검증 데이터를 실제 데이터로 사용하지 않은 본 실험의 한계점을 어느 정도 극복하려는 이유에 기인한다. 검증 데이터 집합 역시 애매성을 없앤 집합을 함께 제

작하였으며 300개, 500개, 700개의 블록들로 구성된 세 부류의 집합을 만들었다. 애매성이 있는 집합은 사이즈별로 각각 검증데이터집합1-1, 검증데이터집합2-1, 검증데이터집합3-1이라 하며 애매성을 없앤 집합은 각각 검증데이터집합1-2, 검증데이터집합2-2, 검증데이터집합3-2라 한다.

2. 모델의 구조

두 가지 훈련 데이터 집합을 사용하여 모델을 훈련시켰다. 훈련에 사용되는 여러 인자들은 앞에서 언급한대로 훈련 데이터 자체의 예측 정확도를 나타내는 Type I, Type II 오류를 측정하여 좋은 성능을 보이는 인자들을 선택하였다. Type I 오류란 비위험 개체를 위험 개체로 판단하는 오류이고, Type II 오류는 위험 개체를 비위험 개체로 판단하는 오류이다. 전자보다는 후자가 제품의 품질을 떨어지게 하는 중요한 원인이 되므로 후자가 더욱 중요한 오류이다. 몇몇 연구들에서 이 측정치들을 개선한 ECM(Expected Cost of Misclassification), MECM 같은 모델 평가 측정치를 제안하였지만 이들은 Type I 오류의 중요성이 상대적으로 떨어지는 실시간 시스템 프로젝트에서는 큰 의미가 없으므로 두 측정치를 그대로 사용하였다[17]. 다음은 두 오류를 나타내는 측정치들이다.

- Type I 오류: 비위험 개체를 위험 개체로 선정한 수 / 비위험 개체수
- Type II 오류: 위험 개체를 비위험 개체로 선정한 수 / 위험 개체수

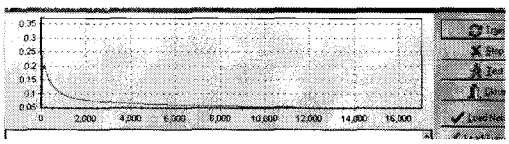

그림 3. 모델의 훈련 1

[그림 3]은 훈련데이터집합1을 사용하여 훈련시킨 과정을 보여준다. 그림에서 수평축은 신경망 알고리즘에서 한 순환을 나타내며, 수직축은 각 순환에서 출력 뉴런의 출력값과 실제 위험도 값들의 차이에 대한 평균값이다. 훈련 데이터 집합이 300개의 블록들로 이루어져 있으므로 한 순환은 300개 입력 메트릭들을 차례로 입력받아 신경망을 훈련시키는 하나의 훈련 단계를 의미한다. 그림 상의 훈련에서는 10,000번 정도의 순환을 거친 후에는 수직축의 값이 0.05에 수렴하였으므로 훈련을 멈추었다.

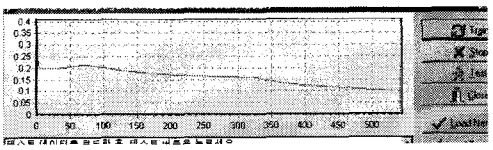

그림 4. 모델의 훈련 2

[그림 4]는 훈련데이터집합2를 사용하여 훈련시킨 과정을 보여준다. 애매성을 없앤 데이터 집합이므로 500 순환을 넘어선 후 갑자기 학습 에러를 나타내는 수직축 값이 0에 가까이 떨어져 학습을 멈추게 된다.

[표 2]는 두 훈련 데이터 집합으로 훈련한 훈련 결과이다. 이는 훈련 데이터 집합을 모델이 얼마나 잘 학습했는가를 나타낸다. 훈련 결과, 애매성을 없앤 훈련데이터집합2의 학습 오류가 적었다.

표 2. 훈련 데이터 학습 오류 결과

오류 / 훈련데이터	Type I	Type II
훈련1	5/268	3/32
훈련2	2/266	0/34

타당성 실험을 위해 블록수가 10, 50, 100, 300, 500, 1000, 3000개인 7개의 데이터 집합을 만들었다. 입력 블록수가 증가하면 예측된 위험 블록수가 증가하여야하며 선정되는 위험 그룹의 크기는 적절해야한다. [표 3]은 훈련데이터집합1로 훈련한 모델의 타당성 실험 결과이다. 훈련데이터집합1은 애매성을 없애지 않은 집합이므로 타당성 실험 집합들과 검증데이터집합1과 유사한 분포를 가지며 300개의 입력 블록들 중 32개의 위험 블록을 가지고 있는 집합이다. 훈련 모델은 훈련 데이터 집합과 비슷한 10%보다 약간 높은 수의 위험 그룹을 선정하였으며 이는 위험 그룹 크기로 매우 적절하다. 애매성을 없앤 검증1-2는 약간 위험 그룹의 크기가 커졌지만 역시 만족할만한 수준이다.

표 3. 타당성 실험 결과

입력수	10	50	100	300	500	1000	3000	검증 1-1	검증 1-2
예측수	3	7	14	38	61	127	404	38	42

3. 성능 평가

모델의 예측 정확도를 평가하기 위해 두가지 훈련 데이터 집합으로 훈련시킨 모델들에 여섯 가지 검증 데이터 집합들을 적용하였으며 [표 4]는 그 결과이다.

표 4. 예측 오류 결과

훈련데이터 / 검증데이터	훈련1		훈련2	
	Type I	Type II	Type I	Type II
검증1-1	8/265	5/35	7/265	13/35
검증1-2	7/265	0/35	0/265	0/35
검증2-1	13/444	8/56	11/444	19/56
검증2-2	11/444	1/56	1/444	0/56
검증3-1	16/624	8/76	18/624	21/76
검증3-2	13/624	1/76	1/624	0/76

[표 4]에서 훈련데이터집합2로 훈련시킨 모델에 검증데이터집합1-2, 2-2, 3-2를 적용한 결과는 거의 오류가 없었다. 왜냐하면 훈련 데이터나 검증 데이터나 모두 애매성을 없앤 집합이기 때문이다.

훈련데이터집합2로 훈련시킨 모델에 검증데이터집합 1-1, 2-1, 3-1을 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 모델은 매우 많은 오류를 냄을 볼 수 있다. 예를 들면 검증데이터집합1-1을 적용하면 20개의 예측 오류를 내며 이 중 중요한 Type II 오류도 13개나 냄을 볼 수 있다. 이는 위험 블록으로 선정하는 29개의 블록들 중 7개가 실제 비위험 블록이고 비위험 블록으로 선정한 블록들 중에는 실제 13개의 위험 블록이 존재한다는 것이므로 예측 모델로서의 의미가 없다.

하지만 반대의 경우인 훈련데이터집합1로 훈련시킨 모델에 검증데이터집합1-2, 2-2, 3-2를 적용시킨 결과는 비교적 만족스러운 결과를 보였다. 특히 중요한 Type II 오류는 거의 내지 않음을 볼 수 있다. 이는 이 모델을 검증데이터집합1-1, 2-1, 3-1에 적용한 결과에서 생긴 Type II 오류는 거의 위험도 판단이 애매한 블록들에서 발생한다는 것을 의미한다. 실제 프로젝트에서 이러한 경계선 상에 있는 블록들을 비위험 블록들로 판단한다 해도 큰 문제는 발생하지 않는다. 또한 실제 데이터는 위험 블록이 명확하게 가려지는 형태를 가지게 된다. 그러므로 이와 같은 결과는 과거 프로젝트 훈련 데이터 집합이 없을 때 설계하고 있는 시스템의 크기에 대한 가정과 여러 제약 조건에 따라 제작한 훈련데이터집합1과 같은 가상 훈련 데이터 집합을 사용하여 훈련시킨 모델을 위험도 예측에 사용할 수 있다는 가능성을 보여준다.

V. 결 론

소프트웨어 산업이 점차 발전하고 대형 소프트웨어 개발 프로젝트가 진행됨에 따라 초기 위험도 예측 모델은 효율적인 자원 할당과 재설계 부분의 자동 결정에 사용되므로 시스템 개발 비용을 낮추는 데 큰 몫을 하고 있다. 기존의 예측 모델들은 판별 분석, 분류 트리, 역전파 신경망 등을 이용한 분류 모델들이었으나 이들 대부분은 과거 훈련 데이터 집합이 필요하다는 큰 문제점이 있었다. 본 논문에서는 객체지향 실시간 시스템 명세 언어인 SDL로 작성한 설계 명세를 역전파 신경망 모델에 적용하였으며, 여러 가지 제약조건을 갖는 일반적인 가상 데이터 집합으로 훈련시킨 모델이 만족할만한 성능을 보임으로써 기존 모델들의 문제점을 해결할 가능성이 있음을 보였다.

향후 연구 과제는 검증 데이터를 실제 데이터로 사용할 수 있는 실제 프로젝트 관리에 본 모델을 적용시켜 유용성을 증명하는 것이며 여러 대형 프로젝트들의 성격과 제약조건들을 분석해 몇 가지 형태의 일반적인 가상 훈련 데이터 집합을 제작하는 것이다.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] K. E. Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case-based reasoning classifiers for predicting high risk software components," J. Systems Software, Vol.55, No.3, pp.301-320, 2001.

[8] Y. Liu and T. M. Khoshgoftaar, "Genetic programming model for software quality classification," Proc. IEEE Int'l Symp. High Assurance Systems Engineering, pp.127-136, 2001.

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

[10] M. M. Thwin and T. S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," J. Systems Software, Vol.76, No.2, pp.147-156, 2005.

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

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

[13] W. W. Agresti and W. M. Evanco, "Projecting Software Defects From Analyzing Ada Designs," IEEE Trans. Software Eng., Vol.18, No.11, 1992.

[14] R. W. Selby and V. R. Basili, "Analyzing error prone system structure," IEEE Trans. Software Eng., Vol.17, No.2, pp.141-152, 1991.

[15] 홍의석, "훈련데이터집합을 사용하지 않는 소프트웨어 품질예측 모델," 정보처리학회논문지, 제10-D권, 제4호, pp.689-696, 2003.

[16] L. Fausett, Fundamentals of Neural Networks, Prentice-Hall International, 1994.

[17] T. M. Khoshgoftaar, N. Seliya and A. Herzberg, "Resource-oriented software quality classification models," J. Systems Software, Vol.76, No.2, pp.111-126, 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 컴퓨터정보학부 교수

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