GAM: 대형 통신 시스템을 위한 위험도 예측 모델

홍의석†

요 약

소프트웨어 개발 초기 단계의 문제점이 개발 후반부 산물의 품질에 심각한 영향을 미치기 때문에 설계 명세를 이용하여 결함경향성이 많은 부분을 예측하는 위험도 예측 모델은 전체 시스템 개발비용을 낮추는 데 중요한 역할을 하고 있으며, 이러한 예측 모델은 결과 산물이 매우 크고 실행 정확성이 요구되는 통신 소프트웨어 같은 실시간 시스템 설계에 더욱 필요하다. 판별분석, 인공신경망, 분류트리 등의 기법들을 이용한 모델들이 제안되었으나 이들은 결과에 대한 원인 분석의 어려움, 낮은 확장성 등의 문제점들을 지니고 있었다. 본 논문에서는 유전자 알고리즘을 이용한 새로운 모델인 GAM을 제안한다. GAM은 위험도 함수를 만들어 내므로 기존의 분류 모델들과는 다르게 설계 개체의 위험도 비교에도 사용가능하다. 여러 내부 특성들과 예측 정확도 비교를 통해 GAM을 잘 알려진 예측 모델인 역전파 신경망 모델(BPM)과 비교하였다.

GAM: A Criticality Prediction Model for Large Telecommunication Systems

Euy-Seok Hong†

ABSTRACT

Criticality prediction models that determine whether a design entity is fault-prone or non fault-prone play an important role in reducing system development costs because the problems in early phases largely affect the quality of the late products. Real-time systems such as telecommunication systems are so large that criticality prediction is more important in real-time system design. The current models are based on the technique such as discriminant analysis, neural net and classification trees. These models have some problems with analyzing causes of the prediction results and low extendability. This paper builds a new prediction model, GAM, based on Genetic Algorithm. GAM is different from other models because it produces a criticality function. So GAM can be used for comparison between entities by criticality. GAM is implemented and compared with a well-known prediction model, BackPropagation neural network Model(BPM), considering internal characteristics and accuracy of prediction.

1. 서 론

소프트웨어 생산에서 분석이나 설계 단계와 같은 초기 단계의 중요성은 소프트웨어 제작 기술이 발달함에 따라 더욱 중요해지고 있으며, 설계 단계를 정량화 하는 설계 메트릭들도 전체 시스템 개발비용을 낮추는 데 중요한 역할을 하고 있다. 또한 설계 단계 산물들을 정량화 하여 최종 개발물의 품질 인자들을 예측하는 예측 모델의

† 정 회 원: 성신여자대학교 컴퓨터정보학부 교수
논문접수: 2003년 1월 10일, 심사완료: 2003년 4월 10일
* 이 논문은 2002년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음

중요성도 매우 높다. 왜냐하면 초기 단계의 문제점이 개발 후반부 산물의 품질에 매우 심각한 영향을 미치고 대부분의 소프트웨어 결점들이 매우 적은 수의 모듈들로부터 발생한다고 알려져 있기 때문이다[1]. 이러한 설계 정량화와 예측 모델의 필요성은 결과 산물이 매우 크고 실행 정확성이 요구되는 통신 소프트웨어 같은 실시간 시스템 설계에 더욱 필요하다.

설계 단계 산물들의 복잡도 메트릭들을 이용하여 최종 개발물의 품질 인자들을 예측하는 예측 모델들에 대한 연구들이 많이 행해졌다. 한 설계 개체의 위험도란 개체가 구현되었을 때 갖는 결함경향성을 의미한다. 위험도 예측 모델이란 설계 개체를 입력으로 받아 그것이 결함경향 개체인지 아닌지를 판단하는 모델이다. 예측 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 모델(Classification model)들이 대부분이었다. 분류 모델 기법으로 사용된 것은 판별분석법, 역전파 신경망, 분류트리 등이 있으며, 이들 중 역전파 신경망 이용 모델이 비교적 좋은 예측 결과를 보인다고 알려져 있다[2]. 하지만 이들 기법들은 분류 트리 기법을 제외하고는 모델 내부에서 행해진 수행 부분을 역추적 하기가 거의 불가능한 블랙박스적인 모델이므로 위험도 결과의 원인을 분석하여 재설계 등의 조치를 취하기가 매우 어려운 문제점이 있다. 본 논문에서는 유전자 알고리즘을 사용하는 새로운 위험도 예측 모델인 GAM을 제안한다. GAM은 기존 모델들과 예측 정확도 면에서 뒤쳐짐이 없으며, 유전자 알고리즘의 수행 결과로 위험도 함수를 만들어내므로 설계 개체간의 위험도 비교가 가능하다는 장점이 있다. 이는 위에서 언급한 블랙박스 모델의 문제점을 어느 정도 극복함을 의미한다.

위험도 예측 모델은 수천, 수만 LOC 정도의 시스템이 아니라 수십만 LOC 이상의 대형 시스템을 개발하는 경우에 유용하다. 수십개의 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 널리 사용되고 있는 객체지향 실시간 시스템 명세 언어인 SDL(Specification and Description Language)로 작성한 설계 명세이며 이를 정량화하는 메트릭들은 [3]에서 제안한 SDL 메트릭 집합을 사용한다. 이 메트릭 집합의 이론적 타당성 및 유용성은 공리적 검증 방법과 차원 분석을 통해 검증되었다[4].

2장에서는 기존에 제안된 위험도 예측 모델들을 살펴보고 그들의 장단점을 논의하며 3장에서는 새로운 예측 모델의 구조를 설명한다. 4장에서는 모의실험을 통하여 분류 모델 중 좋은 성능을 보인다는 역전파 신경망 모델과 제안 모델의 예측 정확도를 비교하고 5장에는 결론과 향후 연구 과제에 대해 기술한다.

2. 위험도 예측 모델

시스템 개발 초기 단계에서 위험도를 예측하기 위한 관련 연구들은 크게 두 가지로 구분할 수 있다. 첫 번째는 과거 유사 프로젝트의 위험도 자료들 즉 훈련 데이터 집합에 기반한 모델을 만들어 현재 수행중인 프로젝트의 위험도 예측에 적용하는 것이다. 이들은 주로 입력 데이터들을 여러 개의 패턴으로 나누는 패턴 분류 기법들을 사용하며 예로는 판별분석[5], 분류트리[1], 역전파 신경망[2], 회귀분석[6], CBR(Case-Based Reasoning)[7], 유전 프로그래밍[8] 등이 사용되었다. [8]에서 제안한 모델은 소프트웨어 품질 분야에 진화 연산 모델을 적용한 거의 유일한 연구이지만, 문제 해결 결과로 여러 가지 요소들(항, 연산자의 종류)을 고려해 정의한 위험도 함수의 계수 스트링 형태가 아닌 위험도 분류 프로그램을 생성한다는 데서 본 연구와 차이가 있다. 즉 생성된 프로그램은 개체의 위험도 여부 판단에는 사용 가능하나 위험도 비교에는 부적합하다.

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 훈련데이터집합을 보유하고 있는 개발 집단이 거의 없다는 것과 보유하고 있다하더라도 과거 프로젝트와 현재 프로젝트의 개발 환경, 개발 시스템의 특성이 매우 유사하여야 한다는 문제점이 있다[9]. 이 문제점을 해결하기 위해 최근에 과거 프로젝트 데이터를 필요로 하지 않는 SOM 신경망을 이용한 모델이 제안되었다[10]. 이외에도 앞의 기술과 같이 예측 결과에 대한 원인 분석이 어렵다는 문제점도 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. 그러므로 각 개체의 위험도를 서로 비교할 수 있다는 장점이 있다. 그러나 기존 데이터들을 통한 경험적인 지식이 아니라 하나의 복잡도 메트릭 값에 의존한다는 단점이 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 기본 메트릭들을 조합하는 것은 매우 주의를 요한다. 여러 메트릭들을 조합함으로써 각 구성 요소들의 특성을 잃어버릴 가능성이 있기 때문이다[11]. 스칼라 메트릭으로 위험도를 예측하는 연구로는 Zage의 연구[12], Agresti의 연구[13]와 데이터 바인딩 기법[14] 등이 있다.

두 가지 방향의 관련 연구들 모두 소프트웨어의 복잡도 메트릭이 프로그램의 오류의 분포와 관련이 있음을, 즉 복잡도가 높은 모듈일수록 오류가 발생할 가능성이 높다는 점을 가정하고 있다.

제안 모델인 GAM은 기존 두 방향 연구들의 성질들을 모두 가지고 있다. 하나의 위험도 함수를 만들어 내므로 위험도 메트릭을 만드는 형태로 생각할 수 있으나 이 함수는 위험도 메트릭같이 어떤 프로젝트에서나 사용할 수 있는 일반적인 모델이 될 수는 없다. 왜냐하면 과거 유사 프로젝트의 데이터를 습득하여 만든 경험 모델이기 때문이다. 즉 첫 번째 언급했던 모델들의 문제점들을 가지고 있는 모델이지만 위험도 함수를 통해 개체들을 비교할 수 있다는 장점이 있다.

3. 제안 모델

본 논문에서 제안하는 새로운 위험도 예측 모델은 유전자 알고리즘을 사용하는 GAM이다. 유전자 알고리즘은 자연 선택과 돌연변이 같은 생물 진화의 원리에 기초한 최적화 기법의 하나로 임의로 선택한 모집단에 유전 연산을 적용하여 좀 더 개선된 해를 생성하는 알고리즘이다. 유전자 알고리즘은 탐색 공간이 크거나 분석적으로 해를 찾을 수 없는 문제에 적당하다. 알고리즘의 기본 생각은 주어진 문제의 해집단을 생성하는 과정에서 상대적으로 좋은 해들을 다음 생성 과정에서 생산하고 나쁜 해들을 제거하여 최적화된 해집단을 찾아내는 것이다. 이 과정에서 사용되는 기본 연산들은 선택, 교배, 변이이다.

3.1. 제안 모델 구조

GAM은 훈련 데이터 집합이 필요한 모델이다. 일단 유전자 알고리즘을 위험도 예측 모델에 적용하기 위해서는 예측 모델 문제의 정의가 필요하다. 위험도 예측 모델은 입력 메트릭 벡터의 크기가 k라 할 때 k 개의 실수 인자를 가지고 한 개의 실수를 출력하는 위험도 함수 F를 찾는 문제이다. 위험도 함수의 출력값이 0을 기준으로 클수록 위험도가 큰 경우이며 작을수록 위험도가 작은 경우이다.

$$F(X) = r, \quad F: R^k \rightarrow R \qquad (1)$$

위험도 함수를 예상되는 항들과 그의 계수로 표현하여 다음과 같이 정의하였다. 정확한 위험

도를 나타내기 위해서는 항의 개수를 잘 결정해야한다.

$$F(x_1,\ldots,x_k)=a_0+a_1x_1^{a_2}+a_3e^{x_1}+a_4\log x_1 + a_5x_2^{a_6}+a_7e^{x_2}+a_8\log x_2 + \cdots\cdots + a_{4k-3}x_k^{a_{4k-2}}+a_{4k-1}e^{x_k}+a_{4k}\log x_k \quad (2)$$

GAM 구현에서는 문제를 단순화하기 위해 위험도 함수 F의 값이 양수일 때는 1을, 음수일 때는 0을 출력하는 함수 f를 사용하였다. 함수 f가 1의 값을 가지면 위험 개체임을 나타낸다.

$$f(x_1,\ldots,x_k)=\begin{cases}1, & \text{if } F(x_1,\ldots,x_k)>0\\ 0, & otherwise\end{cases} \quad (3)$$

하나의 염색체는 함수의 계수들로 표현된다. 계수가 실수이기 때문에 실수들의 염색체로 표현되며, 모두 $4k+1$개의 계수가 필요하므로 염색체를 나타내는 자료형은 (그림 1)과 같은 원형을 갖는다. 계수를 나타내는 유전자는 -9.9와 9.9 사이의 임의의 값을 갖는다. 위험도 여부는 F 값이 0보다 크냐 작으냐에 달려 있으므로 유전자의 크기보다는 상대적으로 부호가 중요하다.

0.023	7.311	4.003	· · · ·	-3.713
a_0				a_{4k}

(그림 1) GAM 염색체의 예

모집단의 크기는 100으로 정했으며 초기 모집단을 형성하는 염색체의 유전자들은 계수 범위 내의 난수들로 초기화하였다. 또한 보다 넓은 검색 공간을 보장하기 위하여 세대형 유전자 알고리즘[2]을 사용하였다. 모집단의 크기가 100이고 한 세대에 100개의 자손이 생성된다.

선택과 교배는 일반적인 기법인 룰렛 바퀴 선택(roulette wheel selection) 방법과 일점 교배 방법을 사용하였다. 교배 위치는 임의의 위치이며 교배할 확률은 0.6으로 하였다. 변이는 -9.9에서 9.9 사이의 난수를 발생시켜 사용하였으며 변이를 할 확률은 0.0333으로 하였다. 교배의 결과로 생기는 자손들은 무조건 새로운 모집단에 넣지 않고, 부모와 비교해서 부모 중 어느 하나보다 품질이 더 좋은 경우에만 넣었다.

목적 함수는 염색체로 이루어진 위험도 함수가 훈련 데이터 집합의 자료를 얼마나 잘 판단하는지를 평가해야 한다. GAM의 목적 함수는 위험도 예측 모델의 성능 평가에 많이 사용되는 수치인 Type I 오류와 Type II 오류 결과를 이용한다. Type I 오류는 비위험 개체를 위험 개체로 판단하는 경우고 Type II 오류는 위험 개체를 비위험 개체로 판단하는 경우이다. 전자는 소프트웨어 검증 단계의 비용을 높이지만 후자는 제품의 납품 시기를 늦출 뿐만 아니라 생산 제품의 품질을 떨어지게 한다. 그러므로 후자가 전자보다 더욱 중요한 오류이며, 목적 함수는 Type II 오류가 적은 염색체를 더 우수하게 판단해야 한다. Type I 오류와 Type II 오류수를 각각 E_1, E_2라 했을 때 목적 함수 $O(X)$는 (4)와 같다. $O(X)$ 값은 위험도 예측 오류가 커질수록 작아지며 E_2에 2를 곱함으로써 Type II 오류에 가중치를 두었다[3].

$$O(X)=\frac{1}{E_1+2E_2} \quad (4)$$

$$O(X)=\frac{\sum_{j=1}^{m}|F(trm_{j1},\ldots,trm_{jk})-FP_j|}{m} \quad (5)$$

훈련 데이터 집합이 m개의 입력 설계 개체로 구성되고 j번째 설계 개체가 (trm_{j1}, ... ,trm_{jk})로 정량화 되었으며 이 설계 개체의 실제 위험도 결과는 FP_j라 할 때 식 (5)는 훈련 데이터 집합의 입력 벡터들을 함수 (2)에 입력한 결과와 실제 위험도 결과와의 차이들의 평균을 의미한다. 연구 초기에는 식 (5)를 목적 함수로 하였으나 모의실험 결과 식 (4)가, 발생시키는 오류 종류와

2) 만들어진 자손 염색체의 수가 모집단 염색체의 수에 근접함. GAM은 자손 염색체 수를 모집단의 크기와 같게 하였다.

3) type II 오류가 type I 오류보다 얼마나 중요한지에 대한 정량적 근거는 없다. 그러므로 실험적으로 E_2에 2, 3, 4를 곱해보고 제곱을 해보았으나 E_2 부분이 너무 커지면 목적 함수 값이 너무 E_2에만 의존하게 되고 값이 너무 작아져 변별력이 없어지는 문제가 있었다.

예측 정확성 측면에서 (5)보다 더 좋은 결과를 보였으므로 (4)를 목적 함수로 사용하였다. 목적 함수의 값이 클수록 궁극적으로 찾고자 하는 위험도 함수와 가까우므로 좋은 품질의 염색체로 간주한다.

GAM은 위에서 기술한 유전자 알고리즘의 인자들을 사용하며, 훈련 데이터 집합을 사용하여 최적해를 구하면 이를 이용한 위험도 함수 (2)와 (3)을 정의할 수 있다. 정의된 위험도 함수를 이용하여 검증 또는 적용 데이터 집합의 각 개체들을 함수의 입력으로 하여 출력값을 구한다. 이 함수의 출력값이 입력 개체의 위험도 여부이다.

4. 모의실험

4.1. 제작 데이터 집합

제안 모델의 유용성을 검증하기 위해 기존 분류 모델 중 가장 우수하다고 알려진 역전파 신경망 모델(BPM)과 예측 정확도를 비교하는 실험을 행하였다. 설계 개체 유형으로는 SDL 시스템 분석 단계에서의 블록을 사용하였으며 하나의 블록을 정량화하는 메트릭 벡터는 (BRS, EBS, EBC, BP, BS, BR)[4]이다[3]. [10]과 유사한 제약 조건을 가진 데이터 집합을 제작하였으며 각 블록의 위험도는[5] SDL을 이용한 통신 소프트웨어 시스템의 설계 경험이 있는 두명의 소프트웨어 공학자에 의해 결정되었다.

훈련 데이터 집합은 BPM과 GAM의 학습에 필요하며 이는 이들 모델의 성능을 결정하는 가장 중요한 요인이 된다. 훈련 데이터의 형태는 (BRS, EBS, EBC, BP, BS, BR, FP)로 적용 데이터에 FP가 첨가된 형태이다. FP는 모델의 출력값으로 1은 위험 개체를 0은 비위험 개체임을 나타낸다. 500개의 블록들의 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 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 그룹 간의 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.

모델 훈련에 사용되는 여러 인자들은 훈련 데이터 자체의 Type I, Type II 오류를 측정하여 좋은 성능을 보이는 인자들을 선택하였다. BPM은 학습률을 0.05, 운동량 변화율을 0.2로 하였으며, GAM은 교배율을 0.6, 변이율을 0.0333으로 하였다.

4.2. 모델의 예측 정확도 비교

평가 실험은 데이터집합1과 데이터집합2를 모두 사용하였으며, 평가 방법은 앞에서 기술한 Type I, Type II 오류의 수를 사용하였다. 다음은 각 모델의 예측 정확도를 평가하기 위하여 본 실험에서 사용한 측정치들이다.

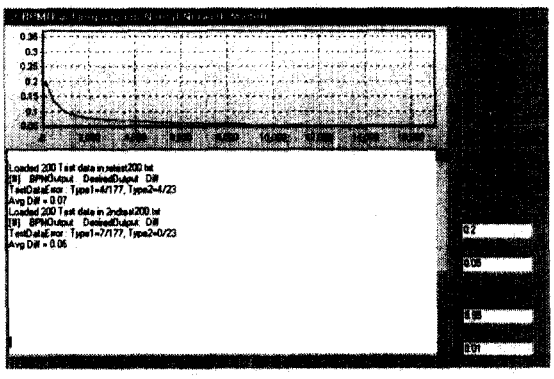
Type I 오류정보: 비위험 개체를 위험 개체로 선정한 수 / 비위험 개체수

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

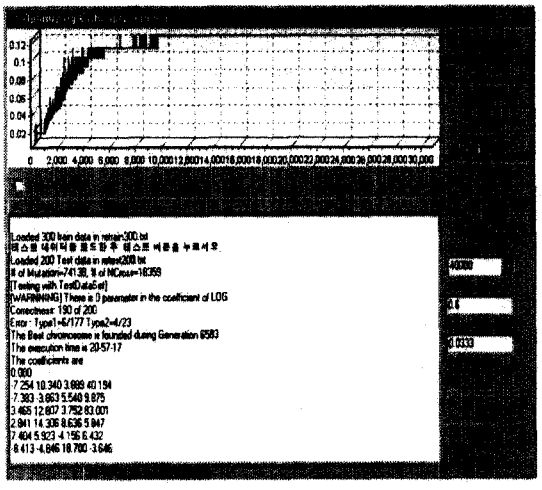
BPM과 GAM은 두 종류의 훈련 데이터 집합을 사용한 모델에 두 종류의 검증 데이터 집합을 적용시켜 네가지의 결과를 얻었다. (그림 2)는 훈련데이터집합1로 훈련시킨 BPM의 훈련 과정과 검증데이터집합1과 검증데이터집합2의 예측 과정을 나타낸다. 그래프의 x축은 입력 집합 전체의 훈련을 한 순환이라 할 때 순환값들을 나타낸다. 그래프 값은 각 순환에서 출력층 뉴런의 출력값과 해당 입력 벡터들의 실제 위험도값들의 차이들의 평균값을 나타낸 것이다. 그림 상의 훈련에서는 10000번 정도의 순환을 거친 후에는 0.05에 수렴하였으므로 훈련을 멈추었다.

4) 개체 정량화에 사용된 메트릭들에 대한 설명은 [3]을 참조하기 바란다.

5) 식 (5)의 FP_i



(그림 2) BPM의 훈련과 위험도 예측



(그림 3) GAM의 훈련과 위험도 예측

(그림 3)은 훈련데이터집합1로 훈련시킨 GAM의 훈련 과정과 검증데이터집합1의 예측 과정을 나타낸다. 그래프의 x축은 세대수, y축은 목적 함수 값을 나타낸다. 그림에서 보듯이 10000번 정도의 세대를 거친 후에는 수렴하였으므로 훈련을 멈추었으며 예측 결과값으로 오류수와 최적해 값이 출력되었다. 이 최적해 값들을 식(2)에 나타낸 위험도 함수의 계수에 넣으면 위험도 함수가 완성된다. 한 가지 주의할 점은 이 위험도 함수값은 측정 이론(measurement theory) 측면에서 보면 서수 스케일(ordinal scale)이라는 것이다. 즉 이를 이용하여 두 개체의 위험도를 비교할 때 어느 개체가 더 크다 라고는 할 수 있으나 얼마정도 또는 몇 배로 더 크다 라고는 할 수 없다.

<표 1>은 두 훈련 데이터 집합으로 훈련하였을 때 BPM과 GAM의 훈련 결과이다. 이는 훈련데이터 집합을 두 모델이 얼마나 잘 학습했는가를 나타낸다. 훈련 결과, 위험 그룹과 비위험 그룹 분포의 차이를 크게 한 훈련데이터집합2의 학습 오류가 적었으며 GAM이 근소한 우위를 보였다.

<표 1> BPM, GAM의 학습 오류 결과

훈련오류 / 모델	훈련데이터집합1		훈련데이터집합2	
	Type I	Type II	Type I	Type II
BPM	5/268	3/32	2/266	0/34
GAM	4/268	2/32	1/266	0/34

<표 2>는 BPM과 GAM의 예측 결과들을 나타낸 것이다.

<표 2> BPM, GAM의 예측 오류 결과

검증오류 / 모델		검증데이터집합1		검증데이터집합2	
		Type I	Type II	Type I	Type II
훈련데이터집합1	BPM	4/177	4/23	7/177	0/23
	GAM	6/177	4/23	2/177	0/23
훈련데이터집합2	BPM	6/177	11/23	0/177	0/23
	GAM	4/177	12/23	0/177	0/23

<표 2>에는 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 두 모델은 매우 많은 오류를 냄을 볼 수 있다. 하지만 반대의 경우인 훈련데이터집합1로 훈련시킨 모델을 검증데이터집합2에 적용시킨 결과는 비교적 만족스러운 결과를 보였다. 이는 과거 프로젝트 훈련 데이터 집합이 없을 때 설계하고 있는 시스템의 크기에 대한 가정과 여러 제약 조건에 따라 제작한

가상 훈련 데이터 집합을 사용하여 훈련시킨 모델을 위험도 예측에 사용할 수 있다는 가능성을 보여준다.

훈련데이터집합1로 훈련시킨 모델들에 검증데이터집합1을 적용한 결과는 BPM이 Type I 오류 부분에서 GAM보다 우수하지만 검증데이터집합2를 적용하였을 때는 반대의 결과가 나왔다. 훈련데이터집합2로 훈련시킨 모델에 검증 데이터2를 적용한 결과는 두 모델 모두 오류를 하나도 내지 않았다. 이는 데이터의 분포가 위험 그룹과 비위험 그룹 간에 어느 정도 차이가 있다면 두 모델 모두 정확한 결과를 냄을 의미한다. 결과적으로 GAM이 BPM보다 매우 근소한 차이의 좋은 성능을 보였지만 두 모델은 예측 정확도 부분에서 비슷한 성능을 보였다. BPM은 모델의 인자들을 변화하여 훈련함에 따라 결과값에 많은 변화가 일어나지 않았지만 GAM은 매우 많은 변화가 일어났다. 그러므로 GAM은 미세한 인자 조정을 통해 보다 좋은 결과를 내는 모델을 얻을 수 있으리라 생각된다.

5. 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 개발비용을 낮추는 데 큰 몫을 하고 있다.

기존의 예측 모델들은 판별 분석, 분류 트리, 역전파 신경망 등을 이용한 분류 모델들이 대부분이었으며, 본 논문에서는 유전자 알고리즘을 이용한 예측 모델인 GAM을 제안하였다. GAM의 설계를 위해 여러 연산자 및 항과 계수들로 구성된 위험도 함수 형태를 정의하였으며, 유전자 알고리즘의 수행 결과로 위험도 함수를 얻었다. GAM 역시 대부분의 기존 모델들과 같이 훈련데이터집합을 필요로 하고 프로젝트 수행 환경에 의존하는 문제점을 가지고 있지만 서수 스케일의 위험도 함수를 사용하여 설계 개체의 위험도를 비교할 수 있다는 장점이 있다. 즉 GAM은 좀 더 위험한 개체에 더 많은 자원 할당을 가능케 함으로써 단순히 설계 개체들을 위험한가 아닌가를 나타내는 두 그룹으로 나누어 자원 할당을 하는 방법에 비해 프로젝트 전체 비용을 크게 감소시키는 효과를 낸다.

GAM의 특성 평가뿐만 아니라 예측 정확도에 대한 성능 평가를 위해 기존의 모델들 중 성능면에서 좋다고 알려진 BPM과의 성능을 모의실험을 통해 비교하였다. 비교 결과 제안 모델이 근소한 성능 우위를 보임을 보였다. GAM과 BPM 모두 특성상 프로젝트 환경에 의존하는 모델이므로 이 실험 결과가 GAM이 모든 경우에 기존 모델들보다 좋은 성능을 보인다는 것을 의미하진 않지만 기존 모델들에 비해 성능면에서 뒤쳐지지 않는다는 것을 의미한다. 또한 제안 모델의 훈련 인자들을 보다 많은 경우로 변형시킴으로서 제안 모델의 성능을 높일 수 있음을 발견하였다.

향후 연구 과제는 많은 실제 프로젝트 관리에 본 모델을 적용시켜 유용성을 증명하는 것이며 계속적인 훈련 인자 변화를 통해 보다 나은 성능의 모델을 제작하는 것이다.

참 고 문 헌

[1] 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, pp. 207-215, 2001.

[2] 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, pp. 85-91, 1995.

[3] 홍의석, 홍성백, 김갑수, 우치수, "SDL 설계 복잡도 메트릭 집합," 정보과학회 논문지(B), 제24권 제10호, pp. 1053-1062, 1997.

[4] 홍의석, 정명희, "SDL 메트릭 집합의 분석적 검증," 정보처리학회 논문지, 제7권 제4호, pp. 1112-1121, 2000.

[5] T. M. Khoshgoftaar and E. B. Allen, "Early Quality Prediction: A Case Study

in Telecommunications," *IEEE Software*, vol. 13, no. 1, pp. 65-71, Jan. 1996.

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

[7] K. E. Emam, S. Benlarbi, N. Goel and S. N. Rai, "Comparing case-based reasoning for predicting high risk software components," *J. Systems Software*, vol. 55, 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] N. Ohlsson and H. Alberg, "Prediction Fault-Prone Software Modules in Telephone Switches," *IEEE Trans. Software Eng.*, vol. 22, no. 12, pp. 886-894, Dec. 1996.

[10] 홍의석, "훈련데이터집합을 사용하지 않는 소프트웨어 품질예측 모델," 정보처리학회 논문지, 제10-D권 제3호, 2003.

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

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

[13] W. W. Agresti and W. M. Evanco, "Projecting Software Defects From Analyzing Ada Designs," *IEEE Trans. Software Eng.*, vol. 18, no. 11, Nov. 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, Feb. 1991.

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@cs.sungshin.ac.kr