베이지언 사용한 패키지 소프트웨어 인증을 위한 시험 메트릭 선택 기법

(A Method of Selecting Test Metrics for Certifying Package Software using Bayesian Belief Network)

이 종 원 † 이 병 정 †† 오 재 원 ††† 우 치 수 ††††
(Chongwon Lee) (Byungjeong Lee) (Jaewon Oh) (Chisu Wu)

요 약 오늘날 급속한 패키지 소프트웨어 제품의 증가 추세에 따라서, 소프트웨어 제품에 대한 품질 시험 요구 또한 증가하였다. 소프트웨어 제품 시험 시 중요한 요소는 무엇을 시험할지 기준이 되는 메트릭의 선정이다. 본 연구에서는 패키지 소프트웨어 종류를 특성 벡터들로 표현하여 메트릭들과의 연관 관계를 확률로서 세밀하게 표현한다. 특성 벡터란 소프트웨어의 형식 분류 지시자라고 할 수 있으며 특정한 패키지 소프트웨어가 다른 것들과 어떻게 구별되는지 나타낼 수 있다. 분류된 각각의 소프트웨어 형식별로 메트릭을 선정하기 위해서 과거 시험 데이타를 분석하여 활용한다. 베이지언망이 과거 데이타 분석에 이용되며 특성 벡터와 메트릭 간의 의존 관계 네트워크를 구축한다. 구축된 베이지언망은 새로운 패키지 소프트웨어 시험 작업에 적절한 메트릭을 찾아내는데 활용된다.

키워드 : 패키지 소프트웨어 제품, 특성 벡터, 메트릭, 베이지언망

Abstract Nowadays, due to the rapidly increasing number of package software products, quality test has been emphasized for package software products. When testing software products, one of the most important factors is to select metrics which form the bases for tests. In this paper, the types of package software are represented as characteristic vectors having probabilistic relationships with metrics. The characteristic vectors could be regarded as indicators of software type. To assign the metrics for each software type, the past test metrics are collected and analyzed. Using Bayesian belief network, the dependency relationship network of the characteristic vectors and metrics is constructed. The dependency relationship network is then used to find the proper metrics for the test of new package software products.

Key words : package software product, characteristic vector, metric, Bayesian belief network

1. 서 론

오늘날의 개발자 및 사용자들은 필요로 하는 소프트웨어 시스템을 스스로 구축하는 대신 원하는 기능을 즉각적으로 제공하는 패키지 소프트웨어 제품을 구매하려는 경향을 보이고 있다. 또한 미국 국방부 자료에 의하면 컴퓨터 실행 프로그램의 99% 이상이 패키지 소프트웨어 형태인 COTS(Commercial Off The Shelf)라고 나온바 있다[1]. COTS 제품은 변경되지 않고 곧바로 사용될 수 있는 소프트웨어 시스템을 의미한다[2]. 따라서 대부분의 패키지 소프트웨어 제품은 소스 코드를 공개하지 않으므로 품질 평가를 위하여 외부적인 동작 특성을 측정한다. 그러한 품질 평가를 위한 시험 시 무엇을 어떻게 측정할 것인가 하는 판단 기준을 메트릭이라고 한다. 소프트웨어 시스템은 인간이 지금까지 구축한 기술적 시스템들 중 가장 복잡한 것으로 대규모 시스템에서는 최대 10^{20}개수의 변경 가능한 상황을 처리해야 한다[3]. 따라서 소프트웨어 시스템은 여타 다른 시스템에 비해서 완벽한 신뢰성을 갖추기가 어렵다[4]. 결국

· 이 연구는 재단법인 대상문화재단의 학술연구비 지원에 의해 이루어진 연구입니다.
† 학생회원 : 서울대학교 전기·컴퓨터공학부
ljw@selab.snu.ac.kr
†† 종신회원 : 서울시립대학교 컴퓨터공학부 교수
bjlee@venus.uos.ac.kr
††† 정 회 원 : 삼성전자 모바일연구소 연구원
jwoh@selab.snu.ac.kr
†††† 종신회원 : 서울대학교 전기·컴퓨터공학부 교수
wuchisu@selab.snu.ac.kr
논문접수 : 2006년 1월 31일
심사완료 : 2006년 9월 1일

소프트웨어 품질 평가는 완벽을 추구하기보다는 평가를 받는 소프트웨어 제품의 특성이나 사용 환경을 고려할 때 만족할만하다고 보는 정도의 품질 수준을 정의하는 보증 생성을 목적으로 한다[5].

현재 세계 각국에서 시행하고 있는 패키지 소프트웨어 제품 품질 보증 방법은 세부 품질에 대한 표시 없이 단지 보증 마크만을 부여하는 방식이다. 이로 인해 소비자는 소프트웨어의 품질에 대해서 충분한 지식을 얻을 수 없다. 결국 무엇을 테스트 했는지 확실히 명기할 필요가 있으며 그에 따라 어떤 테스트 기준 즉 메트릭을 선택했느냐가 문제가 된다. 보통 소프트웨어 제품 패키지들은 원본 소스 코드 없이 최종 사용자들에게 제공되므로 그러한 경우의 시험 방법은 오직 블랙박스 형식 평가를 하는 것 뿐이다. 따라서 *ISO/IEC 9126-2*[6]에 명시된 외부 소프트웨어 품질 메트릭들이 패키지 소프트웨어 제품 시험 시 평가 집합을 생성하기 위해 사용된다.

현재 소프트웨어 도메인 즉 특정한 사용 영역별로 명확히 정해진 메트릭 선택 국제 기준은 나와 있지 않다. 그러나 다수의 패키지 소프트웨어 혹은 COTS 소프트웨어를 이용한 시스템 구축 시에는 다른 소프트웨어 시스템 구축 방법과는 다르게 평가해야 할 상황들이 발생한다[7]. 소프트웨어는 유형 별로 서로 다른 시험 방법을 요구한다. 왜냐하면 사용 환경 또는 소프트웨어의 특성에 따라 품질을 평가하기 위한 메트릭의 선택이 달라질 수 있기 때문이다. 데이타베이스를 예로 들어볼 경우 공통적으로 테스트를 받아야 하는 메트릭들이 있을 수 있다. 데이타 레코드의 무결성 보존 지원이라든지 시간당 트랜잭션 처리 성능 등이 그에 해당할 수 있다. 그러나 핸드폰 같은 내장형 시스템에 탑재되는 것은 일반적인 데이타베이스와는 달리 좁은 메모리 공간과 느린 프로세서의 성능을 감당하면서도 원하는 수준의 기능을 제공하는 것을 확인할 수 있는 메트릭들이 추가되어야 하고 상황에 따라서 또 다른 메트릭들이 추가되어야 할 수도 있다.

또한 그러한 테스트들을 통해 축적된 과거 데이타를 활용하기 위한 메커니즘을 생각해 본다면 우선적으로 통계적 기법이 거론될 수 있다. 그러나 소프트웨어 메트릭과 메트릭에 의해 측정되는 소프트웨어 품질 사이에는 연관 관계가 있을 수 있어도 통계적 방법이 그와 같은 연관 관계를 확실히 뒷받침하지는 못 하는 상황이다[8]. 따라서 확실히 드러나지 않는 메트릭과 소프트웨어 품질 사이의 상관 관계를 잡아내기 위해 데이타 마이닝 기법 등이 몇몇 연구에서 시도되었다. 완전한 데이타마이닝 기법은 아니지만 비슷한 맥락에서 메트릭과 소프트웨어 분류 그룹간의 관계를 표출하기 위해 인증 메타모델[9,10]이 제안되었다. 그러나 [9]와 [10]에서는 소프트웨어 분류 상태와 특정 메트릭들과의 연관 관계를 단순한 그룹별 분류로서만 표현할 수 있었다.

본 연구에서는 특정 패키지 소프트웨어 종류를 특성 벡터(characteristics vectors)들로 표현하여 정형화하며 연관 관계를 확률로서 더 세밀하게 표현한다. 특성 벡터란 소프트웨어의 형식 분류 지시자라고 할 수 있으며 특정한 패키지 소프트웨어가 다른 것들과 어떻게 구별되는지 명확하게 나타낼 수 있다. 기본적으로 과거 시험 데이타를 분석하여 활용하는 과정을 자동화하는 것에 초점을 둔다. 이러한 귀납적 방법을 실현하기 위해 베이지언망(Bayesian Belief Network)[11]을 이용하였다. 베이지언망 이용 시에는 과거 시험 데이타를 분석하여 특성 벡터와 메트릭 간의 의존 관계(dependency) 네트워크를 구축한다.

본 논문의 구성은 다음과 같다. 2장에서 소프트웨어 품질 시험 메트릭과 관련된 기존 연구 및 메타모델과 베이지언망을 살펴본다. 3장에서 특성 벡터를 설명하며 또한 특성 벡터와 메트릭간의 연관성을 파악하기 위하여 베이지언망을 적용하는 방법을 설명한다. 4장에서 패키지 소프트웨어 시험 메트릭 선택 예제를 보인다. 5장에서 실제 운용에 필요한 지식을 얻기 위해 예제의 범위를 확대한 사례 연구를 제시한다. 그리고 6장에서 결론을 제시한다.

2. 관련 연구

이 장에서는 기존의 소프트웨어 메트릭 선택 방법과 메타모델에 관해 살펴본다. 그리고 베이지언망의 개념에 대해 간략히 기술한다.

2.1 소프트웨어 메트릭 선택

[12]는 소프트웨어의 품질을 평가하기 위해 객체 지향 언어로 작성된 소스 코드의 특성을 정량화하는데 적합한 메트릭들을 선별하려 시도하였다. 기존 소프트웨어 객체의 품질 데이타와 메트릭 데이타를 확보한 후에 유전 알고리즘을 이용하여 전문가에 의해서 매겨진 품질 순위에 가장 근접한 결과를 나타낸 다양한 소프트웨어 메트릭들의 집합을 결정하였다. 그러나 객체 지향 시스템 도메인에 속하는 메트릭들이 객체의 특성과 적절한 연관 관계를 갖지 못하여 메트릭과 객체의 특성을 분명히 구별하지 못하고 취급하는 단점이 있다.

또 상호연결 다층 신경망 그리드를 이용하여 시행착오에 의한 훈련을 거쳐 메트릭을 선별하는 방법이 있다[13]. 신경망 동작 시에 원하는 값을 얻기 위한 입력 데이타는 분량이 적어도 되므로 일관성이 있는 분류 결과를 내놓을 수 있다. 그러나 이 경우 신경망의 특정 훈련 데이타에 대한 과적응(over fitting) 현상이 문제가 될

수 있다. 즉 신경망을 훈련시키는 아키텍트의 성향이나 의도하는 방향에 따라서 메트릭 선택 및 분류 결과가 편향되거나 한 쪽으로 치우친 결과가 발생 가능한 단점이 있다.

[8]은 기존에 구축된 소프트웨어 메트릭 데이타베이스의 분석을 위해 퍼지 클러스터링(clustering) 사용을 시도하였다. 개별적인 특성 및 특정 메트릭과 대응되는 소프트웨어 모듈들은 결함 발생 횟수에 따라서 분류되고 클러스터를 이룬다. 주안점은 비감독 학습(unsupervised learning) 형식의 인공지능 기법을 이용한 것이다. 처음부터 학습 가이드라인을 지정하지 않고 모든 가능한 경우의 수를 탐색하여 보다 최적의 솔루션을 구할 수 있는 가능성을 항상 열어놓는다. 그러나 기존 데이타에 대한 과적응 현상이 발생하면 통계적 기법과 마찬가지로 편향된 판단을 내릴 수 있어 관리자에 의한 데이타 재수집 및 조정이 주기적으로 필요하다.

베이지언망 또한 메트릭 선정 및 발생 가능성이 높은 소프트웨어의 결함을 추론해내기 위해 적용되었다[14-16]. [14]와 [15]는 [8]과 마찬가지로 통계적 기법의 한계를 지적하면서 불확실한 조건에서 결정을 내리는데 베이지언망이 유용하게 쓰일수 있음을 지적하고 있다. 즉, 소프트웨어 개발 시에 고려될 수 있는 모든 요소, 프로그래머 역량의 경우 숙련도 및 문제 복잡도 등의 요소들과 베이지언망 상의 의존관계를 맺어 상황에 따른 적절함을 확률로 나타내도록 하였다. 그러나 소프트웨어 개발 환경, 프로세스 및 운용 환경 등 상당히 광범위한 요소들을 적절한 도메인에 따라서 분할하지 않고 동시에 고려하도록 하여 다량의 사례 데이타 취합을 위한 적절한 가이드라인이 필요할 것으로 보인다.

[16]에서도 베이지언망 적용을 고신뢰도 시스템 개발에서의 위험 요소 예측을 위한 테스트 메트릭 선별에 적용하는 예를 보이고 있다. 일반적인 통계적 회귀모델에 의하면 테스트 중 많은 결함을 보인 소프트웨어 모듈은 전체 시스템과의 통합 후에도 문제를 일으킬 것으로 예상되지만 오히려 그 반대인 경우가 많이 보고된다. 대안으로 제시된 모듈 결함 예측 모델은 베이지언망에서 전체 소프트웨어 모델 생명주기에 관련되는 최종 소프트웨어 제품과 프로세스 관련 인자들 간에 관계를 엮어서 표현하도록 하고 있다. 그러나 제시된 모델은 개별적인 모듈들을 어떻게 특성화 시켜야 하는지를 언급하지 않고 있다.

2.2 인증 메타모델

인증 메타모델이란 인증 기관이 인증 모델, 인증 프로그램을 결정하기 위해 참조하는 모델을 말한다[9]. 메타모델을 이용하기 위해서는 제품 평가에 적용될 메트릭들을 반드시 계층적인 그래프 구조상의 적절한 그룹에 할당한다. 실제 인증 시 메타모델은 인증 모델로 변환되고 다시 최종적으로 실질적인 제품 평가에 적용되는 인증 프로그램을 생성한다. 그러한 메타모델의 계층 그래프 구조를 여타 분류 구조와 구별짓는 특징은 전문화에 의한 그룹 기반 접근 방식(Group-based approach with specialization)[9]이라고 할 수 있다. 이러한 구조는 귀납적 방법을 이용한 것으로 과거의 사례에서 점진적으로 인증을 위한 분류 구조를 확장하기 위한 시도이다. 즉 효율적으로 과거 인증 데이타를 이용함으로써 동일한 인증 프로그램들이 구성되는 중복 작업을 막는 효과가 있다. 그와 같은 기반 구조에 의해 평가 및 인증 대상 패키지 소프트웨어의 유형에 따른 개별 메트릭의 적합성 여부가 표현 가능하다.

그러한 메타모델에서도 번거로운 부분이 있는데 계층적 패키지 소프트웨어 제품 분류 구조를 수작업으로만 생성하여야 하고 분류 자동화를 위한 메커니즘이 결합되어있지 않다는 것이다. 수작업에 의한 분류는 평가자에 의존하므로 인증기관이나 평가자에 따라서 서로 다른 메트릭 할당이 나오게 될 수 있다. 서로 간에 상이하게 다른 메트릭 할당 구조는 소프트웨어 품질 평가에 있어 논란을 불러 올 수 있으므로 보편적 원리에 기반을 두면서도 자동적인 할당 알고리즘의 편리성을 갖춘 메커니즘이 필요하다.

그러한 할당 메커니즘 보완을 위한 시도로서 [10]에서 정형 개념 분석을 이용하여 메트릭 할당 자동화를 시도하였다. 정형 개념 분석은 사전에 정해진 소프트웨어 분류 그룹들을 위해 특정한 메트릭들을 자동적으로 연관시키는 경우에 편리하다. 그러나 연관 정도를 확률적 수치로 표현할 수 없어 세밀하게 표현하지 못하는 단점을 가지고 있다.

2.3 베이지언망(Bayesian Belief Networks)

베이지언(Bayesian)이라는 것은 데이타를 분석하는 하나의 방법으로서 통계학과 연관이 있다. 베이지언에서는 데이타 분석 시 기존 과거 데이타 또는 분석자의 판단까지 포함할 수 있으므로 상황 변화에 민감한 결론을 얻을 수 있다. 현재의 상황만 알고 있는 것보다 과거의 상황인 사전 확률을 알고 추정을 할 때 더 정확하게 추정할 수 있으므로 사전 확률까지 고려해서 분석하는 것을 베이지언 분석이라고 한다[11].

이러한 베이즈 정리를 활용한 것들 중에서 보다 고려해야 할 변수가 많은 확률 분포를 실용적으로 처리하기 위해 고안된 것이 베이지언망이다. 베이지언망은 사후 확률을 예측하기 위해 고려해야 할 변수들의 집합에서 존재하는 결합 확률 분포를 묘사하는 것으로 정의할 수 있다. 각각의 변수들은 하나의 노드를 형성하면서 각 변수들 간의 부분적 의존 관계를 확률로 표현하는 네트워

크를 구성한다[17]. 베이지언망에서 네트워크 변수 튜플인 $<Y_1, ..., Y_n>$에 대해 $<y_1, ..., y_n>$이라는 값들을 할당할 경우의 결합 확률을 계산하는 수식은 아래와 같이 나타낸다. $Parents(Y_i)$는 네트워크 상에서 Y_i의 직접적인 부모 노드 집합을 나타낸다. 여기서 $P(y_i|Parents(Y_i))$의 값들은 정확하게 노드 Y_i와 연관된 조건부 확률 테이블에 표기되어 있는 값들이라 할 수 있다.

$$P(y_1, ..., y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

이론적으로 정의하면 베이지언망은 조건부 확률 집합과 함께 비종속성 조건 가정 집합을 명시함으로써 어떤 하나의 변수 집합을 관장하는 확률 분포를 묘사한다[17]. 표 1은 각각의 변수 즉 동물들이 갖고 있을 수 있는 특성을 나열하고 대응되는 특성을 갖는 동물이 있을 경우 해당 사항을 표시해 놓은 것이다. 여기서 변수들의 튜플은 동물과 각 동물들의 특성들에 해당하고 각 변수가 가질 수 있는 가능한 확률은 개별 동물들이 특성에 얼마만큼 부합하는지 나타내는 것임을 알 수 있다.

최종적으로 원하는 것은 결합 공간에서의 확률 분포이므로 각각의 변수들을 의존 관계 네트워크를 표현하는 노드들로 구성할 필요가 있다. 그림 1은 그러한 의존 관계 네트워크를 노드와 노드를 연결하는 화살표로써 보여주고 있다. 화살표가 향하는 노드가 부모 노드로부터 영향을 받는 자식 노드이다. 이러한 베이지언망 이용은 유전 알고리즘, 신경망 및 퍼지 클러스터링과는 대조적으로 감독자 및 운영자의 적극적인 개입이 필요하다. 그러나 장시간의 훈련 시간을 요하지 않고 수시로 노드

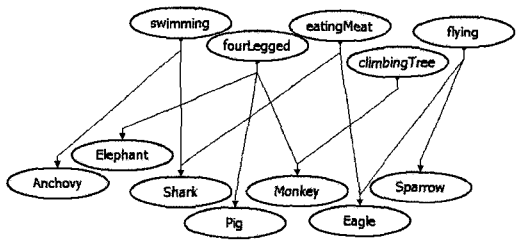


그림 1 변수들의 의존 관계 네트워크

네트워크 설정을 바꿀 수 있으므로 학습 데이타의 급격한 증가에 대응하는 것이 용이하다.

3. 특성 벡터와 베이지언망

이 장에서는 특성 벡터를 설명하며 또한 베이지언망을 특성 벡터와 메트릭간의 연관성을 파악하기 위한 기반 메커니즘으로 선택한 이유도 설명한다.

3.1 특성 벡터

완제품 기반 소프트웨어 시험에 관한 연구로서 소프트웨어 분류 프레임워크가 제안되었다[5]. 소프트웨어를 분류하기 위해 8개의 차원(eight dimensions)을 정의하고 각각의 차원으로부터 분류 대상 소프트웨어에 해당하는 요소를 선택한다. 해당 8개 차원으로부터는 8064개의 조합이 가능하며, 이 조합 중 하나가 특정한 소프트웨어 종류를 나타내게 된다.

본 연구에서 패키지 소프트웨어는 특성 벡터로 대응된다. 특성 벡터는 위 프레임워크와는 달리 계층성을 가진다. 특성 벡터의 각 성분(components)은 소프트웨어와 관련된 속성을 나타낸다. 각 벡터 성분에는 특정 소프트웨어의 속성을 보여주는 벡터들의 구성 요소가 들어간다. 위와 같은 특성 벡터는 하나의 벡터가 특정 소프트웨어의 특성을 나타내는 성분들을 갖는 튜플로서 표현된다.

$CV = (Val(CV_1), Val(CV_2), ..., Val(CV_l))$, l은 특성벡터 CV의 차원

$CV_i = (Val(CV_{i1}), Val(CV_{i2}), ..., Val(CV_{im_i}))$, $1 \leq i \leq l$, m_i는 CV_i의 차원

$CV_{ij} = (Val(CV_{ij1}), Val(CV_{ij2}), ..., Val(CV_{ijn_{ij}}))$, $1 \leq j \leq m_i$, n_{ij}는 CV_{ij}의 차원

$$Val(CV_i) = \begin{cases} 1, & \text{if } \sum_{j=1}^{m_i} Val(CV_{ij}) \geq 1 \\ 0, & otherwise \end{cases}$$

where

if CV_{ij} is a leaf node, $Val(CV_{ij})$ is only a scalar value (0 or 1).

표 1 객체와 특성간의 관계

객체 \ 특성	four Legged	swimming	eating Meat	flying	climbing Tree
Elephant	True	False	False	False	False
Pig	True	False	False	False	False
Shark	False	True	True	False	False
Eagle	False	False	True	True	False
Sparrow	False	False	False	True	False
Anchovy	False	True	False	False	False
Monkey	True	False	False	False	True

if CV_{ij} is not a leaf node, CV_{ij} may have child nodes recursively.

여기서 특성 벡터 CV는 또 다른 특성 벡터 CV_i나 CV_{ij}의 정보를 벡터의 성분으로서 재귀적으로 포함한다. CV_{ij} 이하로 보다 더 세분화된 하위 특성 벡터를 포함시키는 것도 가능하다. 벡터의 성분은 스칼라 값만을 취하며 그러한 성분들은 특정 소프트웨어와 대응할 때 1의 값, 그 외 0의 값을 가진다. CV는 포함하는 성분들 중 하나라도 1의 값을 갖는 것이 있다면 그 자신도 1의 값을 가지며 아니면 0의 값을 갖게 된다. 만약 CV 자신이 더 이상의 하위 특성 벡터를 포함하지 않는 말단 노드(leaf node)라면 오직 0 혹은 1의 값만을 가진다. 결국 벡터의 성분이라는 것은 특정 소프트웨어의 특성(characteristics)들을 나타낸다고 볼 수 있다. 그림 2는 이와 같은 특성 벡터에 속하는 성분들 간의 관계를 보이고 있다. 이해를 돕기 위해 그림 2를 특성 벡터의 수식 형태로 나타낸다면 다음과 같이 될 수 있다.

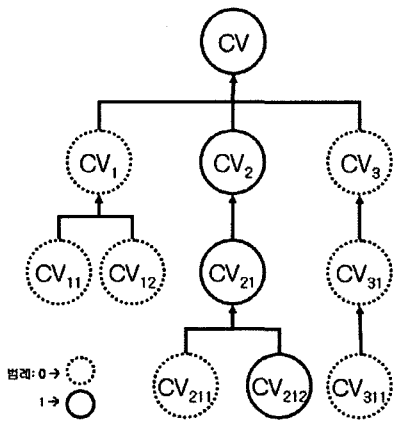


그림 2 특성 벡터 성분들 간의 관계

$CV = (0, 1, 0)$
$CV_1 = (0, 0),\ CV_{11} = (0),\ CV_{12} = (0)$
$CV_2 = (1),\ CV_{21} = (0,1),\ CV_{211} = (0),\ CV_{212} = (1)$
$CV_3 = (0),\ CV_{31} = (0),\ CV_{311} = (0)$

실제 적용 시 이해를 돕기 위해 표 2에 각각의 특성 벡터 별로 가능한 성분들을 나열하였다. 표 2는 메타모델의 소프트웨어 도메인으로 설정되었던 한국정보통신기술협회[1](TTA)의 TTAS.KO-11.0026[18]에 기반 하여 작성되었다. 표 2로부터 특성 벡터는 다음과 같이 설정된다.

1) http://www.tta.or.kr

표 2 특성 벡터 예제

벡 터	성 분
Application Software (CV_1)	Office Management (CV_{11})
	Financial (CV_{12})
	Manufacturing (CV_{13})
	Education (CV_{14})
	Credit Cards (CV_{15})
N/W & Communication (CV_2)	Internet (CV_{21})
	Resolution (CV_{22})
	WAP (CV_{23})
	Security (CV_{24})
	E-Mail (CV_{25})
H/W & System (CV_3)	O/S & System (CV_{31})
	Embedded (CV_{32})
	Monitoring (CV_{33})
	Load Management (CV_{34})
	Server (CV_{35})
Platform (CV_4)	UNIX (CV_{41})
	WINDOWS (CV_{42})

$CV_1 = (Val(CV_{11}), Val(CV_{12}), Val(CV_{13}), Val(CV_{14}), Val(CV_{15}))$
$CV_2 = (Val(CV_{21}), Val(CV_{22}), Val(CV_{23}), Val(CV_{24}), Val(CV_{25}))$
$CV_3 = (Val(CV_{31}), Val(CV_{32}), Val(CV_{33}), Val(CV_{34}), Val(CV_{35}))$
$CV_4 = (Val(CV_{41}), Val(CV_{42}))$

만약 CV_2에 속한 성분인 CV_{21}이 보다 더 세분화된 하위 특성 벡터라면 CV_{21}의 성분을 추가적으로 다음과 같이 나열하는 것이 가능하다.

$CV_{21} = (Val(CV_{211}), Val(CV_{212}))$

앞서 기술한 바와 같이, 예를 들어 CV_1의 성분인 CV_{11}, CV_{12}, CV_{13}, CV_{14} 그리고 CV_{15} 중에서 하나라도 어떤 특정 소프트웨어에 해당하는 것이 있다면 그것은 1의 값이 되어 그대로 CV_1도 1을 결과값으로 갖는다. 그 외의 경우 0이 결과값이 되며 성분 중 어떠한 것이 1의 값을 갖느냐에 따라 서로 다른 소프트웨어의 유형을 나타내는 일종의 소프트웨어의 유전자라고 할 수 있다.

예를 들어 CV_2의 성분 상태가 $(0, 0, 0, 1, 1)$이라고 가정하자. 이 특성 벡터는 패키지 소프트웨어들 중에 한 가지 특정한 종류를 표현하며 특정 도메인에 대응하는 패키지 소프트웨어 종류에 따르는 특정 소프트웨어 특성의 성분들을 갖고 있다. 그러한 CV_2의 성분에 해당하는 소프트웨어 특성은 Security와 E-MAIL이며 해당 패키지 소프트웨어 특성이 보안과 전자우편에 관련된

것임을 알 수 있다. 특성 벡터의 성분은 고정되어 있지 않으며, 기존의 소프트웨어 분류 형식의 구성 요소에 자유로이 매핑될 수 있다.

3.2 베이지언망 적용

소프트웨어 개발 프로세스와 최종 소프트웨어 제품 생산에는 많은 불안정한 요소들이 있다[19]. 그러한 불안정한 요소들을 다루기 위해 본 연구는 베이지언망을 선택했으며 그 이유는 다음과 같다[20].

첫째, 베이지언망은 여러 가지 변수들 간의 확률적 관계를 표현하는 그래픽 모델로서 모든 변수들 간의 의존 관계를 표현하기 때문에 일부분의 데이타만을 가지는 상황에 적절히 대처할 수 있다. 소프트웨어 품질 시험 담당자가 시험을 받는 패키지 소프트웨어의 종류를 베이지언망 위에서 특성 벡터 노드들을 설정하여 표현할 때 불명확하고 불충분한 데이타로 인하여 전체 노드의 일부분만을 설정하는 경우가 있을 수 있다. 그러나 일부분만이 설정되었더라도 다른 노드들과의 의존 관계에 따라서 어떠한 종류의 패키지 소프트웨어인가를 자동적으로 유추하는 것이 가능하다. 또한 기존 시험 사례 데이타를 이용하여 베이지언망 구축 시 시험에 사용된 메트릭 데이타가 각 소프트웨어 도메인별로 골고루 분포가 되어 있지 않을 수 있다. 베이지언망이 가지는 장점인 의존 관계 표현 및 의존 관계 학습이 그러한 과거 데이타 불충분을 대처할 수 있을 것으로 기대한다.

둘째, 변수들 간의 의존 관계를 학습하는데 사용될 수 있기 때문에 응용 분야에 대한 이해를 도울 수 있다. 패키지 소프트웨어 시험 작업을 위한 메트릭 선정 시 시험을 받는 패키지 소프트웨어가 속한 도메인에만 관련된 메트릭들이 우선적으로 검토 대상이 되며 다른 도메인에 속한 메트릭들은 무시될 수 있다. 그러나 베이지언망의 노드 네트워크에서는 관련된 노드에 속한 모든 메트릭들이 검토되어야 한다. 그에 따라 평상시 지나치기 쉬운 결함 관련 테스트를 하게 될 가능성을 높이게 된다. 이전에 시험을 받은 것과 똑같은 업무를 처리하는 패키지 소프트웨어라도 처리하는 데이타의 범위가 넓다는 특성을 가지게 되면 데이타 오버플로우가 발생하는지 여부를 테스트해야 한다는 메트릭이 추가되는 경우가 해당될 수 있다.

셋째, 모델 자체가 원인(causality)과 확률적 의미(probabilistic semantics)를 표현하고 있기 때문에 사전 지식(prior knowledge)과 학습 데이타를 결합하는데 적합하다. 기존 시험 사례 데이타를 베이지언망에서 이용하는 소프트웨어 품질 시험 작업의 경우 각 노드의 사전 확률 설정이 사전 지식을 활용하는 것이라고 볼 수 있다. 사전 지식을 활용하여 메트릭 선별 작업을 진행하게 되면 그 과정에서 새로이 시험 작업에 관련된 데이타를 획득할 수 있다. 전에 없었던 새로운 패키지 소프트웨어의 특성 벡터 설정 데이타나 그러한 새로운 패키지 소프트웨어에 필요한 메트릭들에 관한 정보들이 새로운 학습 데이타에 해당될 수 있다. 베이지언망은 감독자나 운영자가 노드 네트워크 구조를 변경하는 것을 허용하므로 손쉽게 새로운 학습 데이타를 노드 네트워크에 표현할 수 있다.

기존 연구인 [9]에서는 인증 메트릭 선별을 위한 선행 작업으로서 계층적 소프트웨어 분류 체계의 구축이 요구되었다. 그러한 분류 체계는 정형화된 방식에 의하지 않고, 인증 업무 수행자에 의해서 구축된다. 본 연구에서도 베이지언망 또한 노드 네트워크 그래프라는 기본 구축 사항이 선행 조건으로서 필요하기는 하지만 처음부터 의도적으로 구축되는 것은 아니며 기존 사례를 참고하여 샘플 데이타에서 서로 간에 관련 있는 변수들을 노드로 지정하여 번개가 천둥을 유발한다는 것과 같은 적절한 의존 관계에 따라 각 노드가 연결 관계를 갖는다. 기본적으로 메트릭은 자식 노드로서, 패키지 소프트웨어의 유형을 나타내는 특성 벡터는 부모 노드로서 관계를 맺는다. 또한 자식 노드 메트릭은 부모 노드의 확률 상태에 종속되는 사전 확률 테이블을 갖는다. 실제 시험을 받게 되는 패키지 소프트웨어의 유형에 따라 부모 노드들의 확률 값이 변경되면 그에 따라서 영향을 받는 자식 노드 메트릭의 사후 확률 값이 계산된다. 그러한 사후 확률 값은 자식 노드 메트릭의 중요도를 표현한다고 할 수 있다.

패키지 소프트웨어 품질 시험의 경우 기존 과거 시험 사례 데이타를 획득할 수 있고 그러한 과거 데이타에서 어떤 소프트웨어의 특성이 특정한 메트릭에 부합할 것이냐는 의존 관계적 변수도 사전에 추론할 수 있으므로 베이지언망 적용이 손쉽다는 것을 알 수 있다. 또한 이론보다는 실제 현장 적용에 더 큰 관심이 있는 실무자들의 경우에도 베이지언망은 시험 대상 소프트웨어의 도메인에 관한 기본적인 의존 관계 네트워크만을 이해하면 되므로 시험 업무를 맡는 최종 평가자들에게 가는 부담이 적다[21].

4. 특성 벡터와 메트릭 간 연관성

이 장에서는 베이지언망 이용 가능성을 검증하기 위해 특성 벡터와 메트릭간의 구조적인 연관성을 파악하는 예제를 보인다.

4.1 특성 벡터와 메트릭 간 설정 사항

서로 다른 패키지 소프트웨어 제품은 제품별로 정의된 고유의 도메인에서 운영된다. 따라서 품질 시험을 위한 기초 작업으로서 반드시 각 소프트웨어 종류 별로 고유한 도메인을 결정해야 하며 각 도메인별로 시험에

필요한 메트릭들이 선택된다.

그러한 도메인 설정을 위해 [9]에서는 하향식 분화에 의한 그룹화 접근방식을 이용했다. 수작업으로 구축되는 계층적 소프트웨어 분류 구조의 각 그룹에는 인증 작업을 위한 적절한 메트릭들이 할당 된다. 본 연구에서는 처음부터 전체 특성 벡터 및 메트릭 노드들의 네트워크 구조 구축을 지원할 메커니즘 결합을 주안점으로 삼았다. 이 접근 방법에서 모든 패키지 소프트웨어의 종류는 특성 벡터로써 표현된다. 그러한 상태에서 과거 시험 사례 데이타를 참조하여 특정 메트릭들과 부모와 자식 노드들 간의 연결 관계로 표현되는 베이지언망의 네트워크 구조를 형성한다.

본 예제는 한국정보통신기술협회의 과거 인증 사례 데이타를 참조하였고, 기존 인증 데이타 중에서 6개 패키지 소프트웨어들의 품질 데이타를 발췌하였다. 예제를 위해 데이타들을 정리하고 색인을 하고 나서 그 결과로 얻은 기초 데이타가 표 3에 있다. 알아보기 용이하도록 표 2에 나열되어 있는 특성 벡터 중에서 2가지 종류만이 포함 되도록 패키지 소프트웨어들을 지정하였다. 즉 Application Software인 CV_1과 N/W & Communication인 CV_2 특성 벡터가 전체 패키지 소프트웨어들을 크게 2가지 부류로 분류하는 기준이 되며, 또한 특성 벡터의 성분에 어떠한 패키지 소프트웨어의 특성이 해당하는가에 따라서 세부 분류가 갈라지게 된다. 표 3에는 특성 벡터 CV_1과 CV_2의 성분들 중에서 기존 인증 사례의 패키지 소프트웨어에 대응하는 것만을 나열하였다.

표 3에는 또한 각 패키지 소프트웨어의 시험에 어떤 메트릭이 쓰였는지를 해당 패키지 소프트웨어 옆의 괄호 안에 표기해 놓았다. 기존 시험 사례 데이타 활용을 추구하는 본 연구에서는 매우 중요한 데이타라고 할 수 있다. 위와 같이 기본 참조 데이타 수집 및 설정이 이루어지고 나서 그러한 데이타를 기반으로 원하는 분야에 활용할 수 있는 형태로의 전환이 필요하다. 본 연구에서 사용하는 데이타 형식은 베이지언망이다.

베이지언망에서는 네트워크상에서 각 노드의 사후 사건 발생 확률을 예측하기 위해 노드들의 집합에서 존재하는 결합 확률 분포를 계산해야 한다. 그러한 결합 확률 분포를 수작업으로 구성하는 것은 대단히 어려우므로 별도의 베이지언망 실험용 도구를 이용할 수 있다. 대부분의 베이지언망 구축 도구는 복잡한 확률 계산을 할 필요 없이 시각적으로 베이지언망을 구성하는 노드들을 생성하고 각 노드 간의 네트워크 연결 상태를 시각적으로 설정하면 복잡한 결합 확률 분포를 손쉽게 구성할 수 있도록 도와준다.

그림 3은 각 노드를 의존 관계를 고려하여 네트워크로서 연결한 것을 보여준다. 이 경우 특성 벡터의 성분이 위치하는 노드들이 부모 노드가 되고 메트릭 노드들이 자식 노드가 된다. 그림 3에서 "CV" 접두사가 붙는 노드 이름은 특성 벡터 성분들을 나타낸다. "M" 접두사가 붙은 노드들은 메트릭을 나타낸다. 각 특성 벡터는 또한 기존 품질 시험 사례 참조 데이타에 의해서 특정한 특성 벡터에 어떠한 메트릭이 쓰였는지 여부에 대해 에지(*edge*)로 연결되는 네트워크 구조로써 의존 관계를 표현하고 있다.

표 3 패키지 소프트웨어와 메트릭 및 특성 벡터들의 대응 예제

	CV_{12}	CV_{14}	CV_{11}	CV_{22}	CV_{24}
$P_1(M_1,M_3,M_8)$	•				
$P_2(M_{11},M_{12})$		•			
$P_3(M_5,M_{11})$			•		
$P_4(M_2,M_6)$				•	
$P_5(M_4,M_7)$					•
$P_6(M_9,M_{10})$			•		•

* P_1~P_6 : 기존 인증 사례의 패키지 소프트웨어들

* M_1~M_{12} : 개별 패키지 소프트웨어 품질 시험에 사용된 외부 소프트웨어 품질 메트릭

M_1 : Functional adequacy
M_2 : Functional implementation completeness
M_3 : Functional implementation coverage
M_4 : Functional specification stability
M_5 : Computational accuracy
M_6 : Data exchangeability
M_7 : Access controllability
M_8 : Access auditability
M_9 : Functional compliance
M_{10} : Interface standard compliance
M_{11} : Failure resolution
M_{12} : Incorrect operation avoidance

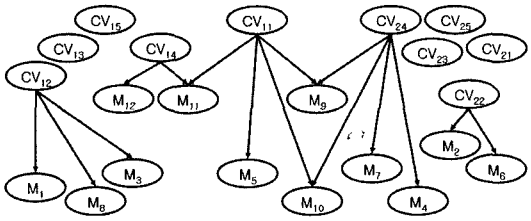


그림 3 특성 벡터와 메트릭 노드들의 네트워크 (CV: 특성 벡터, M: 메트릭)

다음 절에서는 위에서 구축해 놓은 특성 벡터와 메트릭 노드들의 베이지언망 상에서 결합 확률 분포를 조작하고 설정한다. 즉 베이지언망 상에서 각 특성 벡터의 관측 값을 설정한 후에 새로이 시험할 패키지 소프트웨어에 어떠한 메트릭이 확률적으로 적합할지 예측하는 것을 보인다.

4.2 시험 메트릭 선택을 위한 베이지언망 활용

베이지언망에서 노드가 자리를 잡기 위해서는 각 노드별로 두 가지 정보가 필요하다. 직접적인 부모 노드와의 의존 관계 정보와 그러한 부모 노드들 간의 확률 분포를 묘사하는 조건부 확률 테이블이 그것이다. 따라서 의존 관계를 표현하는 네트워크 구조가 결정되어있다면 각각 노드별로 부모 노드의 가능한 상태 조합에 따라 나타날 수 있는 확률들을 묘사하는 결합 확률 분포 테이블이 설정되어야 한다. 특정한 변수를 나타내는 노드와 관련이 있는 부모 노드들의 가능한 여러 가지 상태에 따라서 특정한 확률 집합을 지정하고 설정할 수가 있다. 가능한 상태는 "True" 혹은 "False" 형식의 2가지다. 어떤 노드의 부모 노드가 2개라면 조합에 의해 모든 가능한 상태는 4가지가 존재 가능하다. 그 중 하나의 상태에 대해 해당 노드가 True의 확률을 1.0으로 갖는다면 해당되는 특정한 부모 노드의 상태에 따라서 해당 노드가 영향 받는 확률이 100%라는 것이 된다. 반대로 False의 확률이 1.0이라면 영향 받지 않을 확률이 100%가 된다.

패키지 소프트웨어 품질 시험에 실질적으로 참조되는 메트릭들은 각각의 메트릭들이 개별적인 노드를 이루면서 특성 벡터에 속하는 성분 노드들과 연결 관계를 이룬다. 그러한 경우의 확률 분포 설정 예제가 표 4에 나와 있으며 "Interface standard compliance"라는 메트릭 노드 M_{10}에 관련된 부모 노드들의 확률 분포 상태에 따른 가능한 여러 가지의 사전 확률들을 보여주고 있다.

메트릭 노드 M_{10}의 부모로서 나열된 노드들은 그림 3에서 묘사된 베이지언망의 의존 관계에 따라서 형성된 것이다. 여기서 CV_{11}과 CV_{24}는 특성 벡터의 결과값을 나타내며 True는 결과값 1을 나타내고 False는 0을 나타낸다. 메트릭 노드 M_{10}은 부모 노드가 아닌 여타 다른 노드들에 대해서는 독립적이다. 즉 부모 노드의 상태만이 사후 확률에 영향을 미친다. 이를 일반적인 확률적 연관 관계 수식으로 표현하면 다음과 같다.

표 4 그림 3의 메트릭 M_{10}의 부모 노드 상태에 따른 확률 분포

Parent Nodes (True = 1, False = 0)		Node: M_{10}
CV_{11}	CV_{24}	Probability of Being True
True	True	1.0
	False	0.5
False	True	0.5
	False	0.0

$P(M_{10}=True \mid CV_{11}=True, CV_{24}=True)$ 혹은
$P(M_{10}=True \mid CV_{11}=True, CV_{24}=False)$ 혹은
$P(M_{10}=True \mid CV_{11}=False, CV_{24}=True)$ 또는
$P(M_{10}=True \mid CV_{11}=False, CV_{24}=False)$

노드 M_{10}을 비롯한 메트릭 노드들의 경우에는 각각의 메트릭 노드가 시험을 하려는 패키지 소프트웨어에 필요한 정도가 어느 정도인지 확률적으로 가늠할 수 있도록 하기 위해 세 가지의 경우를 고려하여 사전 확률을 설정하였다. 즉 메트릭 노드의 모든 부모 노드가 전부 1의 값을 가질 경우나 여러 부모 노드들 중 하나 이상이 1의 값을 가질 경우, 혹은 그 어떤 부모 노드도 해당되지 않아 모두 0의 값을 가질 경우다. 여기서는 모든 메트릭의 부모 노드는 전부 시험을 하려는 패키지 소프트웨어의 특징을 나타내는 특성 벡터에 해당한다.

표 4를 보면 그와 같은 세 가지 경우가 사전 확률적 분포로서 나와 있음을 볼 수 있다. 4개의 가능한 사전 확률들 중에서 제일 위에 있는 것은 모든 부모 노드가 True일 때, 즉 CV_{11} = True이고 CV_{24} = True인 경우를 나타낸다. 제일 아랫줄은 부모 노드가 모두 시험을 하려는 패키지 소프트웨어의 특성에 전혀 해당이 되지 않을 때, 즉 CV_{11} = False이고 CV_{24} = False인 경우라 할 수 있다. 제일 아랫줄과 윗줄을 제외한 가운데 부분에 있는 확률들은 특성 벡터들 중에서 하나라도 해당하는 것이 있을 때, 즉 CV_{11} = True이거나 CV_{24} = True인 경우를 나타낸다.

지금까지 예를 들어본 바와 같이 베이지언망에서 특성 벡터와 메트릭 노드들 간의 네트워크 및 각 노드들의 확률적 경우에 따른 사전 확률 설정이 끝나면 실제 패키지 소프트웨어 품질 시험을 위한 메트릭 선택 시에는 각각의 특성 벡터 관련 노드들을 실제로 시험을 하려는 새로운 패키지 소프트웨어의 특징에 부합하도록 설정하면 된다. 그러한 설정은 징후 값(evidence)을 입

력하여 결합 확률 분포의 평가(evaluation)를 행해짐으로써 이루어진다. 이 예제에서는 품질 시험 메트릭 선택 실험을 위해 다음과 같은 특성을 갖는 패키지 소프트웨어를 선택하였다.

$(CV_{11} = True,\ CV_{24} = True)$

여기서는 True로 설정이 되지 않는 특성 벡터 노드들은 모두 False라고 가정하며 이미 앞서 구축한 베이지언망 상에서의 의존 관계를 고려한다면 다음과 같은 확률적 연관 관계를 도출할 수 있다. CV_{11}과 CV_{24} 노드의 영향을 받는 자식 노드인 메트릭 노드들은 M_4, M_5, M_7, M_9, M_{10} 그리고 M_{11} 이 있음을 의존 관계 상에서 판별할 수 있다. 베이지언망의 결합 확률 산출 방식에 따라서 각 메트릭 노드의 사후 확률은 직접적인 부모 노드들이 갖고 있는 사전 확률들의 영향을 고려하여 얻어진다.

$P\ (M_4=True \mid CV_{24}=True) = 1.0 = 100\%$
$P\ (M_5=True \mid CV_{11}=True) = 1.0 = 100\%$
$P\ (M_7=True \mid CV_{24}=True) = 1.0 = 100\%$
$P\ (M_9=True \mid CV_{11}=True, CV_{24}=True) = 1.0 = 100\%$
$P\ (M_{10}=True \mid CV_{11}=True, CV_{24}=True) = 1.0 = 100\%$
$P\ (M_{11}=True \mid CV_{11}=True, CV_{14}=False) = 0.5 = 50\%$

위의 확률적 의존 관계에 따른 계산 결과로 얻어진 메트릭 노드들의 결합 확률 분포에 따른 사후 확률을 막대그래프 형태로 보면 그림 4와 같다. 인과적으로 독립적이어서 영향을 받지 않는 메트릭 노드들은 사후 확률이 0%임을 알 수 있다.

이 경우 100%로 해당 특성 벡터에 부합하는 메트릭들은 M_4, M_5, M_7, M_9와 M_{10}이다. 그림 3에 따르면 특성 벡터 노드 CV_{11}과 CV_{24}의 영향을 직접 받는 노드들이 해당 메트릭 노드들임을 알 수 있다. 따라서 이들 5

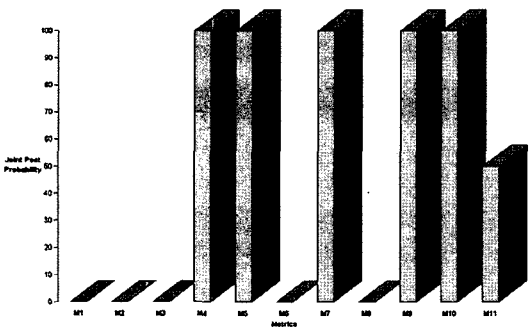


그림 4 메트릭 노드들의 사후 확률 분포

가지 메트릭들이 우선적으로 가장 높은 적용 순위를 갖는다.

문제는 M_{11} 노드가 CV_{14} 노드와 CV_{11} 노드 2개로부터 동시에 영향을 받는다는 점이다. CV_{14}는 현재 관측된 징후 값이 입력되어있지 않다. 이러한 기존 사례 데이타 불충분 상황에서 변수들의 의존 관계가 대략적인 확률을 도출하여 메트릭 선택에 도움을 줄 수 있다. 앞서 구축한 베이지언망의 각 변수 노드들 간의 의존 관계 상 모든 메트릭 노드들은 각각의 두 가지 특성 벡터에 종속되어 있으므로 각 특성 벡터가 1의 값을 갖는다면 그에 따라 메트릭들도 사후 확률로서 0 이상의 값을 갖게 된다. 사후 확률은 사전 확률 설정 상태에 의해 영향을 받으므로 명확한 사전 확률 정의 기준이 필요하다. 가장 먼저 고려해야 할 사항은 특정 메트릭 노드에 연관된 부모 노드인 특성 벡터 노드들 중에서 징후 값이 True인 것의 개수가 얼마인가에 따라서 해당 메트릭 노드의 사후 확률 값이 어떻게 변화할 것인가 하는 것이다. 본 연구에서는 평균치를 활용하였다. 즉, 특정 메트릭 노드에 연관된 특성 벡터 노드의 개수가 2개이고 그 중 하나가 True라면 1/2의 평균값을 적용하여 50%의 사전 확률을 부여하는 방식을 사용하였다. 그에 따라서 M_{11} 메트릭 노드는 사후 확률로서 50%의 값을 갖게 된 것이다.

이와 같이 베이지언망에 의해 산출된 각각의 메트릭별 사후 확률들은 일종의 수치적 참고 자료이므로 실제로 패키지 소프트웨어 품질 시험 작업에 적용되려는 시점에는 상황에 따른 적절한 해석을 위한 가이드라인을 마련해두는 것이 좋다. 사후 확률이 100%로 나온 메트릭은 일차적으로 점검되어야 할 것이며 50%인 메트릭은 필요시 선택적으로 점검받아야 한다는 의미로 해석될 수 있다. 패키지 소프트웨어의 유형에 따라 베이지언망에 의해 자동적으로 산출된 메트릭별 적용 우선순위는 모든 가능성을 고려한 결과이므로 일반적인 소프트웨어가 아닌 치명성(criticality)이 높은 소프트웨어의 경우 낮은 사후 확률을 갖는 메트릭들이라도 세심하게 점검해야할 필요가 있다.

지금까지의 예에서 기존 시험 사례 데이타 활용 시 베이지언망의 의존 관계 네트워크가 패키지 소프트웨어 품질 시험을 위한 메트릭 선택 시 여러 가지 패키지 소프트웨어 특성에 따른 각 메트릭들의 적용 우선순위를 비교적 간단한 방법으로 산출한다는 것을 알 수 있다. 베이지언망을 변경하고 유지 보수하는 것은 비교적 용이하여 각 노드들 간의 의존 관계 네트워크를 새로운 시험 사례가 보고 될 때 마다 추가하거나 수정하면 된다. 또한 베이지언망의 의존 관계 및 의존 관계 표현 형식에 의해 기존 사례 데이타가 불충분 할 경우에도 대처가 용이하다.

표 5 특성 벡터에 의한 메트릭 선택 기법과 관련 연구 비교

	특성 벡터	[14]	[15]	[16]
베이지언망 채용 여부	채용	채용	채용	채용
시각적 노드들간 연결 표현 여부	표현	표현	표현	표현
지향하는 소프트웨어 품질 시험 방식	블랙 박스	화이트 박스	화이트 박스	블랙 박스
메트릭 선택을 위한 도메인 지정 방식	국부적	포괄적	포괄적	포괄적
사전 확률 설정 기반	과거 사례	참고 자료	전문가	전문가
도메인 전문가의 지원 필요성 여부	불필요	필요	필요	필요

4.3 기존 연구와의 비교 및 토의

베이지언망의 기본 토대인 베이지언 확률론 자체는 그 역사가 오래되었으나 최근의 알고리즘 개선과 활발한 소프트웨어 도구 개발에 힘입어 많은 분야에서 본격적으로 활용되고 있는 추세를 보이고 있다[22]. 유전 알고리즘, 신경망 혹은 퍼지 등의 방법은 복잡한 모델을 직접적으로 조작하거나 구현하기가 곤란하나 베이지언망은 체계적으로 표현하고 제어할 수 있다. 예를 들어, 신경망의 경우는 노드에 대응하는 퍼셉트론 혹은 뉴런들간의 네트워크를 시각적으로 표현하는 것은 가능하나 학습 데이타에 의한 가중치 조절을 위해 출력값 되먹임을 위한 기본 네트워크 구조를 준수해야만 한다. 따라서 특성 벡터 등과 같은 별도의 고유한 노드간 연결 체계 구축이 쉽지 않다. 베이지언망은 직관적 및 시각적으로 노드간 연결을 자유로이 구성할 수 있어 새로운 네트워크 모델 구성이 쉽다고 볼 수 있다.

그 밖에 본 연구의 특성 벡터에 의한 메트릭 선택 기법이 기존의 베이지언망 이용 기법[14-16]에 대해 차이점 혹은 장점이 될 수 있는 것이 있다면 명확한 메트릭 적용 도메인 구별에 있다고 볼 수 있다. 기존 베이지언망 이용 관련 연구는 시험 대상 소프트웨어의 특성 종류를 명백히 구별하지 않고 개발 프로세스와 같이 묶어서 고려하는 방식으로 주관적인 요소를 배제하지 않은 상태에서 메트릭 선별을 시도하였다. 본 연구는 블랙박스 형식 시험(black-box testing)을 위한 의사 결정 시스템 구성을 목표로 하였다. 따라서 소스 코드 혹은 개발 프로세스와는 무관하게 소프트웨어의 유형 분류 도메인에 의거한 특성 벡터 및 메트릭 노드들간의 객관적인 네트워크 구축을 우선시 했다. 운용자는 베이지언망을 구성하는 노드들간의 연결 상태 및 사전확률 조정에만 주력하게 되어 전문가의 주관적인 의견 반영을 허용하면서도 확률에 기반한 최대한 객관적인 메트릭 선별 참고 자료를 얻을 수 있다. 표 5는 특성 벡터에 의한 메트릭 선택 기법과 관련 연구를 비교한 결과를 정리하여 보여주고 있다.

5. 적용 예제

앞서 베이지언망의 메트릭 선별 메커니즘 적용 가능성을 간단한 예제를 통해 보였다. 실제로 다종다양한 패키지 소프트웨어의 유형에 따라서 베이지언망 구축을 시도할 경우 노드의 개수가 많아지므로 원활한 운용에 필요한 지식을 축적할 필요가 있다. 그림 5는 한국정보통신기술협회에서 공개한 20여개의 과거 패키지 소프트웨어 제품 시험 및 인증 데이타에 기반하여 시험 메트릭 선별 가능 여부를 실험한 베이지언망의 모습이다. 베이지언망 구축 도구로는 "GeNIe 2.0[2)]"이라는 공개 소프트웨어 도구를 이용하였다. 전체 노드 개수는 300여개이며 각 노드간의 의존 및 영향 관계를 나타내는 연결선인 아크(arc)의 개수는 600여개가 된다. 연결선의 굵기는 부모 노드가 자식 노드에 영향을 끼치는 정도를 나타내며 굵을수록 강한 영향력을 나타낸다. 각각의 노드들은 3개의 노드 그룹 중 하나에 속하는데 그림 5의 상단에 위치한 것이 특성 벡터 노드 그룹, 중앙이 패키지 소프트웨어 노드 그룹, 그리고 하단이 메트릭 노드 그룹에 속한다.

특별히 특성 벡터 노드들간의 네트워크는 베이지언망의 장점을 살려 계층성을 그대로 표현하도록 설정되었다. 각 벡터를 구성하는 요소인 성분들과 하위 특성 벡터들은 부모 노드가 되어 자식 노드인 상위 특성 벡터에 영향을 끼친다. 그림 5에서 예를 들면 "CV-HW_and_System_Related" 노드는 "CVL-OS_and_System_Support" 노드, "CVL-HW_or_System_Internal_Control" 노드, "CVL-HW_or_System_External_Control" 노드, 그리고 "CVL-HW_and_System_Security_Support" 노드 등 4개 부모 노드와 연결되어 있다. 특성 벡터의 구성 요소를 이루는 이들 부모 노드들 중에서 한 개라도 징후값이 "True"인 것이 있다면 상위 특성 벡터 노드의 사후 확률 값은 무조건 "True"가 되도록 사전 확률이 설정된다.

과거 시험 및 인증 사례 데이타에 의해 자식 노드로서 특성 벡터 노드와 연결되는 패키지 소프트웨어 노드의 경우 "PK2-Be_Project" 노드가 "CVL-OS_and_System_Support" 노드와 연결되어있음을 볼 수 있다. 이것은 과거에 품질 인증 시험을 받은 "PK2-Be_Project"

2) http://genie.sis.pitt.edu

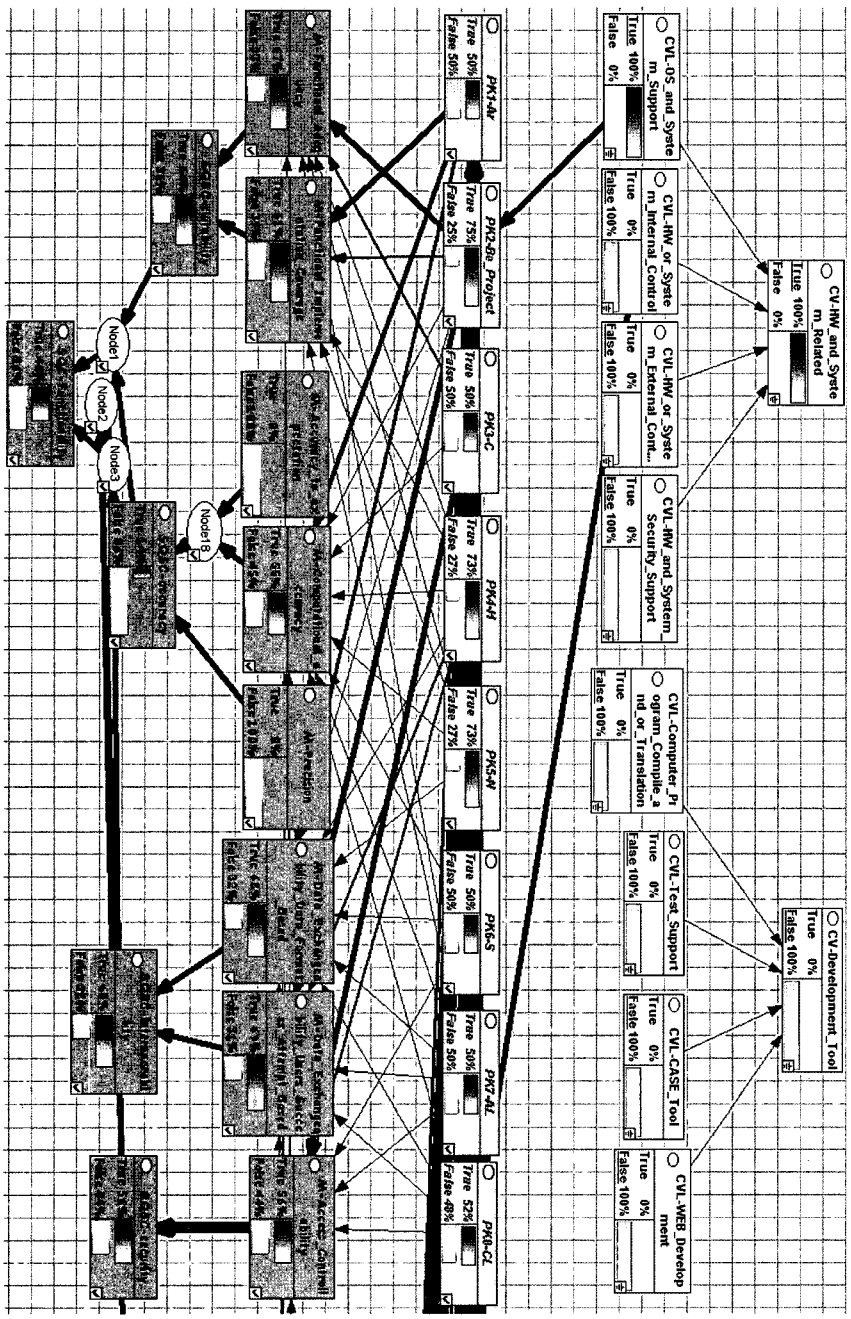


그림 5 패키지 소프트웨어 시험 메트릭 선별을 위한 실험적 베이지언망

라는 패키지 소프트웨어 제품이 운영체제와 시스템 지원 기능 관련 특성을 가졌음을 의미한다. "CVL-OS_and_System_Support" 노드가 "True"로 설정된 상태여서 연결된 "PK2-Be_Project" 노드 역시 "True"가 75%가 되어 사후 확률에 영향을 받은 것을 알 수 있다.

자식 노드로서 "PK2-Be_Project" 노드의 영향을 받는 여러 메트릭 노드들 중의 하나인 "M-Functional_Adequacy" 또한 과거 시험 사례 데이타를 참조하여 연결된 것이다. 사전 확률 설정 및 특성 벡터 노드들의 징후값 설정에 따라서 새로이 시험을 하려는 패키지 소프트웨어 제품에의 "Functional_Adequacy" 메트릭에 대한 적합도가 67%임을 알 수 있다. 그림 6은 또 다른

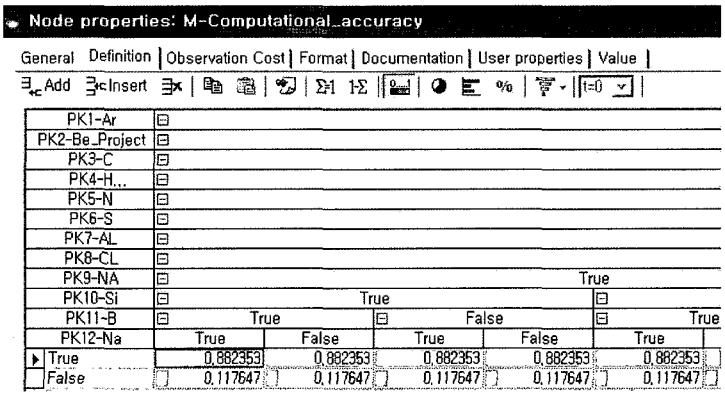


그림 6 베이지언망 도구에서의 노드 사전 확률 설정

메트릭 노드인 "M-Computational_accuracy"를 도구를 이용하여 사전 확률을 설정하는 화면을 보여주고 있다. 그림 5에서는 보이지 않은 패키지 소프트웨어 노드들을 포함하여 12개의 부모 노드가 사전 확률 설정에 쓰였음을 알 수 있다. 그림 6 하단의 "True"와 "False"행들이 사전 확률들을 나타내며 "True"행의 0.882353이라는 확률값은 10개 이상의 패키지 소프트웨어 노드들이 "True"인 경우를 나타낸다. 그와 같은 설정 상태에서 그림 5를 보면 메트릭 노드 "M-Computational_accuracy"의 사후 확률이 여러 가지 서로 다른 부모 노드의 영향을 받아 55%로 되었음을 알 수 있다.

GeNIe 2.0은 진단(diagnosis) 기능이라는 것도 제공하는데 목표(target)로 지정된 노드들에 대한 사후 확률 값 크기에 따른 순위(rank)를 보여주는 기능이다. 그러한 순위는 곧 메트릭들을 새로운 패키지 소프트웨어 시험에 대해 적용해야할 우선 순위로 표현한 것이라고 볼 수 있다. 그림 7을 보면 "Message_Understandability_in_use" 메트릭을 비롯한 5개 메트릭들이 상위 등급에 0.900의 확률(probability)로 올라와 있음을 볼 수 있다. 그와 같은 메트릭들은 우선적으로 적용되어야 할 것이라고 볼 수 있고 "Audit_Trail_Capability"처럼 사후 확률이 0.429로 낮은 것들은 선택적으로 적용될 것이다. 그리고 "Change_Cycle_Efficiency"처럼 0%의 확률을 갖는 것은 일차적인 고려 대상에서 제외될 수 있을 것이다. 그림 7에서 오른쪽 부분의 "Evidence" 영역은 목표 노드들의 순위를 결정하는데 있어서 참조된 부모 노드들의 상태를 나타낸다.

이와 같이 구축된 베이지언망의 적용 실험은 이전에 사용한 시험 메트릭이 무엇인지를 이미 알고 있는 10여 개의 패키지 소프트웨어 제품 유형별 특성 벡터 설정에 의해 행해졌다. 베이지언망이 산출한 시험 메트릭들의 사후 확률은 과거에 인증 기관이 선택한 메트릭인 경우에 대해 높은 수치를 보였다. 몇 가지 예를 보면, 실험 대상인 패키지 소프트웨어 제품 "N"의 경우 높은 사후 확률을 나타낸 시험 메트릭 중 83%가 과거 사례에서 쓰였던 메트릭들과 일치했다. 또 다른 패키지 소프트웨어 제품 "Sc"의 경우 89%, 그리고 "EB"의 경우 96%가 과거 사례와 일치했다.

6. 결 론

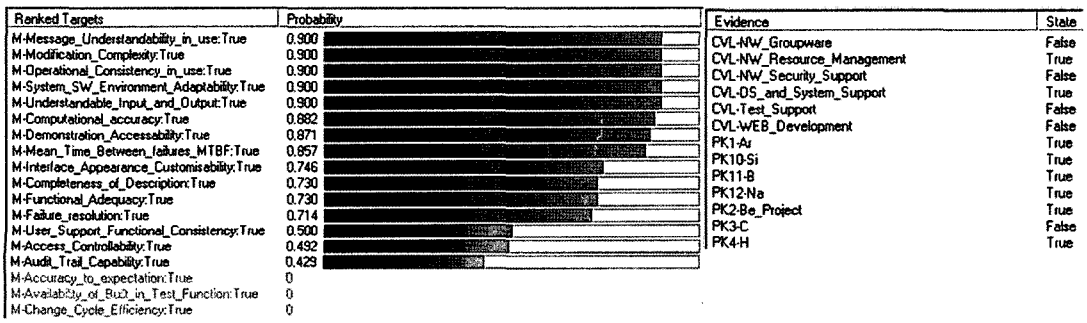


그림 7 메트릭 노드들의 사후 확률 순위 목록

본 연구는 기존 시험 사례 데이타를 활용하는 귀납적 추론에 의한 패키지 소프트웨어 품질 시험 메트릭 선택 방법을 제안하였다. 많은 패키지 소프트웨어들은 서로 같은 특성을 공유하여 적절한 사전 분류 작업이 없는 시험 프로그램 생성은 예전 프로그램과 중복이 되는 경우를 가져올 수 있다. 또한 수많은 경우의 조건을 처리해야 하는 소프트웨어 시스템의 특성상 과거 사례를 참조하여 모든 발생 가능한 결함을 방지하고자 하는 테스트는 필수적이며, 어떠한 테스트가 필요할 지를 지정하는 메트릭들의 체계적인 분류 및 선택 메커니즘이 필요하다.

그에 따라서 기존 시험 사례 데이타를 신속하게 분석하고 활용하도록 귀납적 추론 원리에 충실한 베이지언망이 적용되었다. 베이지언망에서는 각각의 패키지 소프트웨어들의 고유한 특징을 특성 벡터 노드들이 나타낸다. 각 특성 벡터 노드들은 기존 시험 사례 데이타를 참조하여 메트릭 노드들과 적절한 의존 관계 네트워크를 형성한다. 그렇게 하여 형성된 네트워크 상의 특성 벡터 노드들에 실제 시험을 행하고자 하는 패키지 소프트웨어의 특성에 따라 징후 값을 설정하게 되면 그에 따라 각각의 메트릭 노드들은 우선적으로 시험에 적용될 순위를 나타내는 사후 확률 값을 나타낸다. 베이지언망의 특성에 따라 기존 시험 사례 데이타가 불충분할 경우에도 어느 정도 확률에 기반한 추론이 가능하다. 또한 베이지언망의 생성 과정 자체가 도구 지원이 되므로 기존 시험 사례 데이타 추가에 따른 유지 보수가 용이하다.

패키지 소프트웨어 제품 품질 시험 기관에서 위와 같이 본 연구에서 제안하는 방법에 따라 패키지 소프트웨어 품질 시험 작업에 필요한 메트릭 선별을 시도한다면 그에 따른 몇 가지 기대 효과를 예상할 수 있다. 일반적인 시험 작업에서는 시험에 적용할 메트릭의 적용 여부를 단순히 예, 아니오 하는 이분법적 방식으로만 표기한다. 본 연구의 베이지언망 적용 방법에서는 특정 패키지 소프트웨어 종류와 메트릭들과의 연관 관계를 확률로서 자세하게 표현한다. 따라서 아니오 라고 표기되어 단순히 무시되는 것과는 달리 적용 가능 여부를 나타내는 사후 확률이 1% 이상이라면 최소 한번은 관련 메트릭들이 태스트에 사용된다. 결국 조금이라도 적용 가능성이 있는 메트릭들을 폭 넓게 테스트함으로써 평상시에 발견하기 어려운 패키지 소프트웨어의 결함을 찾을 가능성을 높이게 된다. 또한 시험 작업 시간을 사후 확률이 다른 메트릭들보다 높게 나온 메트릭들에게 보다 많이 우선적으로 할당하여 시험 작업의 효율화를 추진할 수 있다. 그러한 메트릭별 우선 순위 할당은 결과적으로 총체적인 품질 시험 작업 프로그램 작성을 위한 로드맵 역할을 하는데 기여할 것이다.

본 연구에서는 베이지언망의 노드 네트워크 구축 시 결합 확률 분포에 따른 사후 확률을 계산하여 얻어내기 위해 먼저 사전 확률을 개별 노드마다 설정해 주어야 하는 것을 선행 조건으로 삼고 있다. 그러한 사전 확률을 설정하는데 있어 미리 정해진 조건이나 정형화 되어 있는 방식이 없이 소프트웨어 시험 업무 수행자나 베이지언망 구축 작업자의 판단에 의하는 것은 실제 시험 업무에 있어 유연성을 확보할 수 있는 동시에 약간의 모호함을 가져올 수도 있다. 그러한 모호함은 기존 시험 사례 데이타를 다량으로 취합하고 분석하는 과정에서 어느 정도 일관화 되어 있는 사전 확률 설정 방식을 마련함으로써 해소 가능하리라 예상된다. 또한 품질 시험을 하려고 하는 패키지 소프트웨어의 유형이나 도메인 전문가(domain expert)의 협조 하에 사전 확률을 설정하기 위한 가이드라인을 사전에 마련하는 방법도 생각해 볼 수 있다.

또 다른 고려할 점은 기존 시험 작업에 사용되던 베이지언망이 새로운 메트릭 추가나 특성 벡터의 소프트웨어 분류 구조 세분화 및 변화로 인해 신규 노드를 추가하거나 사전 확률 재설정시 전체적으로 어느 정도의 변동이 있을 수 있는가 하는 것이다. 베이지언망은 기본적으로 인공지능 분야에 사용되던 것으로서 잡음 데이타에 강하고 새로운 데이타 추가에 의해 자동적으로 새로운 환경에 적응하기 위한 진화(evolution)를 할 수 있다. 따라서 새로운 노드 추가에 의한 변동은 크지 않을 것으로 예상된다.

향후 연구 과제로서 이러한 과거 시험 사례 데이타 분석에 기반하여 새로운 소프트웨어 패키지 제품의 품질 시험에 필요한 메트릭을 선정하는 것이 실제 시험 업무에서 유효한지를 검증하기 위해 실질적인 대규모 사례 데이타를 이용한 실험이 필요하다. 즉 패키지 소프트웨어 제품 품질 시험 기관의 기존 시험 사례 데이타를 획득하여 그에 따라 패키지 소프트웨어의 종류와 메트릭간의 의존 관계 네트워크를 베이지언망으로 구축한다. 구축한 베이지언망에 이미 기존 시험 사례 데이타에 존재하는 패키지 소프트웨어 몇 개의 종류별 징후값을 입력하고 결과값으로 나온 개별 메트릭들의 사후 확률 값들을 조사한다. 베이지언망이 예측한 메트릭별 적용 우선 순위가 실제 시험 사례에서 쓰인 메트릭들을 보여주고 있는지 여부가 검증 자료로서 쓰일 수 있을 것이라고 기대된다.

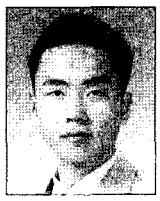
참 고 문 헌

[1] Boehm, B., "Managing software productivity and reuse," *IEEE Computer*, Vol. 32, No. 9, pp. 111-113, Sep. 1999.

[2] Sommerville, I., *Software Engineering* (7th Ed.), Addison Wesley, 2004.
[3] Friedman, M. A., and Voas, J. M., *Software Assessment: Reliability, Safety, Testability*, New York: John Wiley & Sons, Inc., 1995.
[4] Jones, C., *Software Assessments, Benchmarks, and Best Practices*, New York: Addison-Wesley, 2000.
[5] Voas, J., "Limited software warranties," *In Proc. of 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pp. 56-61, 2000.
[6] ISO/IEC, "DTR 9126-2: Software Engineering - Product quality Part 2 - External metrics," *ISO/IEC JTC1/SC7 N2419*, 2001.
[7] Basili, V. R., and Boehm, B., "COTS-based systems top 10 list," *IEEE Computer*, Vol. 34, No. 5, pp. 91-95, May 2001.
[8] Dick, S., and Kandel, A., "Fuzzy clustering of software metrics," *In Proc. of The 12th IEEE International Conference on Fuzzy Systems 2003*, FUZZ '03, Vol. 1, pp. 642-647, May 2003.
[9] Oh, J., Lee, B., Park, D., Lee, J., Hong, E., and Wu, C., "Using hierarchical classification to certify software packages," *In Proc. of 1st ACIS International Conference on Software Engineering Research & Applications (SERA '03)*, pp. 270-275, 2003.
[10] Lee, C., Oh, J., Lee, B., and Wu, C., "Selecting metrics for package software certification using Formal Concept Analysis," *In Proc. of 2nd International Conference on Software Engineering Research & Applications (SERA '04)*, pp. 291-297, May 2004.
[11] 김용대, *베이지안 통계학*, 자유 아카데미, 2000.
[12] Vivanco, R. A., and Pizzi, N. J., "Identifying effective software metrics using genetic algorithms," *In Proc. of Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 1305-1308, 2003.
[13] Alexiuk, M. D., and Pizzi, N. J., "Discriminatory software metric selection via a grid of interconnected multilayer perceptrons," *In Proc. of Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 1131-1134, May 2003.
[14] Fenton, N. E., and Neil, M., "A critique of software defect prediction models," *IEEE Transactions on Software Engineering*, Vol. 25, No. 5, pp. 675-689, Sep./Oct. 1999.
[15] Neil, M., and Fenton, N. E., "Predicting software quality using Bayesian belief networks," *In Proc. of 21st Anniversary Software Engineering Workshop, NASA Goddard Space Flight Centre*, pp. 217-230. Dec. 1996.
[16] Fenton, N., Krause, P., and Neil, M., "Software measurement: uncertainty and causal modeling," *IEEE Software*, Vol. 19, No. 4, pp. 116-122, Jul./Aug. 2002.
[17] Mitchell, M. T., *Machine Learning*, McGraw-Hill, 1997.
[18] Telecommunications Technology Association (TTA), "Standard for a classification scheme of software reuse," *TTAS.KO-11.0026*, 2000.
[19] Ziv, H., and Richardson, D., "Bayesian confirmation of software testing uncertainties," *In Proc. of International Conference on Software Maintenance*, Sep. 1997.
[20] Heckerman, D., "A tutorial on learning with Bayesian networks," *Technical Report MSR-TR-95-06, Microsoft Research*, Redmond, WA, 1995.
[21] Fenton, N. E., and Neil, M., "Software metrics: Roadmap," *The Future of Software Engineering, Anthony Finkelstein (Ed.), ACM Press* 2000.
[22] Lauritzen, S. L., and Spiegelhalter, D. J., "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *J. R. Statistical Society Series B*, Vol. 50, No. 2, pp. 157-224, 1988.

이 종 원
2001년 연세대학교 정보산업공학과 졸업(학사). 2003년 서울대학교 대학원 전기·컴퓨터공학부 졸업(석사). 2003년~현재 서울대학교 대학원 전기·컴퓨터공학부 박사과정. 관심분야는 소프트웨어 품질 시험/평가/인증

이 병 정
정보과학회논문지 : 소프트웨어 및 응용 제 33 권 제 6 호 참조



오 재 원
1997년 서울대학교 계산통계학과 졸업(학사). 1999년 서울대학교 대학원 전산과학과 졸업(석사). 2004년 서울대학교 대학원 전기·컴퓨터공학부 졸업(박사). 2004년~현재 삼성전자 SW연구소 모바일 SW플랫폼팀 책임 연구원. 관심분야는 소프트웨어 품질 시험/평가/인증, 모바일SW플랫폼, 실시간 태스크 스케줄링

우 치 수

1965년~1972년 서울대학교 응용수학과 졸업(학사). 1972년~1974년 한국과학기술원 연구원. 1975년~1977년 서울대학교 대학원 전산과학과 졸업(석사). 1977년~1982년 서울대학교 대학원 전산과학과 졸업(박사). 1978년 영국 라퍼러 대학 연구원. 1975년~1982년 울산대학교 전자계산학과 조교수, 부교수. 1985년~1986년 미국 미시간대학교 postdoc. 1988년~1990년 한국정보과학회 총무 이사. 1990년~1992년 한국정보과학회 학회지 편집위원장. 1992년~1993년 한국정보과학회 부회장. 1996년~1998년 한국정보과학회 소프트웨어 공학연구회 운영위원장. 1982년~현재 서울대학교 전기·컴퓨터공학부 교수