SVM을 이용한 위험모듈 예측

(An Estimation of Risky Module using SVM)

김 영 미 † 정 충 희 ††
(YoungMi Kim) (ChoongHeui Jeong)
김 현 수 †††
(HyeonSoo Kim)

요 약 안전-필수 분야에 사용되는 소프트웨어의 신뢰도(dependability)를 보장하기 위해 소프트웨어의 테스팅과 확인 및 검증활동이 매우 중요하다. 본 연구에서는 위험수준이 높은 소프트웨어 모듈을 소프트웨어 수명주기 초기에 예측하여, 테스팅과 확인 및 검증 활동에 대한 자원할당을 도울 수 있게 해준다. 다중 클래스 분류를 지원하는 SVM (Support Vector Machine)을 이용하여 소프트웨어 모듈의 잠재위험수준 을 예측한다. 잠재위험수준이 상대적으로 높게 나온 모듈들에 대해 테스팅과 확인 및 검증을 집중적으로 실시함으로써 보다 효과적으로 소프트웨어의 품질을 향상시킬 수 있다. 또한, 원전의 계측제어계통에 사용되는 안전-필수 소프트웨어의 안전성 심사를 위한 대상 모듈을 샘플링할 때 활용할 수 있을 것으로 기대된다.

키워드 : 안전-필수 소프트웨어, 소프트웨어 신뢰도, SVM, 소프트웨어 테스팅, 소프트웨어 V&V

Abstract Software used in safety-critical system must have high dependability. Software testing and V&V (Verification and Validation) activities are very important for assuring high software quality. If we can predict the risky modules of safety-critical software, we can focus testing activities and regulation activities more efficiently such as resource distribution. In this paper, we classified the estimated risk class which can be used for deep testing and V&V. We predicted the risk class for each module using support vector machines. We can consider that the modules classified to risk class 5 and 4 are more risky than others relatively. For all classification error rates, we expect that the results can be useful and practical for software testing, V&V, and activities for regulatory reviews.

Key words : Safety-Critical Software, Software Dependability, SVM, Software Testing, Software V&V

· 본 연구는 교육과학기술부가 출연하고 한국원자력안전기술원이 시행한 원자력기술개발사업의 일환으로 수행됨
· 이 논문은 제35회 추계학술대회에서 'SVM을 이용한 위험모듈 예측에 관한 연구'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 한국원자력안전기술원 공학연구실 선임연구원
ymkim@kins.re.kr
†† 정 회 원 : 한국원자력안전기술원 공학연구실 책임연구원
k148jch@kins.re.kr
††† 종신회원 : 충남대학교 전기정보통신공학부 컴퓨터전공 교수
hskim401@cnu.ac.kr

논문접수 : 2008년 12월 13일
심사완료 : 2009년 4월 6일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제6호(2009.6)

1. 서 론

최근 정보처리기술의 발전과 더불어 원자력 발전소의 계측제어시스템과 같은 안전-필수 시스템에서도 소프트웨어 기반의 디지털 기술을 채택하기 시작했다. 안전-필수 분야에 사용되는 소프트웨어의 신뢰도(dependability)를 보장하기 위해 소프트웨어의 테스팅과 확인 및 검증활동이 매우 중요하다. 본 연구에서는 소프트웨어 요구사항단계에서 정의되는 위험수준에 대한 정보와 설계 및 구현단계에서 얻을 수 있는 소프트웨어 모듈의 여러가지 메트릭스를 이용하여 소프트웨어 모듈에 대한 상대적인 잠재위험을 예측하는 시스템을 제안한다. 위험수준이 높은 모듈을 예측하여 그 모듈에 대하여 테스팅과 확인 및 검증에 대한 자원을 보다 많이 할당함으로써 소프트웨어 신뢰도를 보다 효과적으로 향상시킬 수 있다. Boehm과 Papaccio는 소프트웨어 시스템의 다수의 결함이 소수의 컴포넌트에서 발생하며, 소프트웨어의 확인 및 검증은 높은 위험도를 가지는 문제들을 확인하고 제거하는 것에 집중해야한다고 하였다[1]. 본 연구에서는 다중 클래스 분류를 지원하는 SVM(Support Vector Machine)을 이용하여 소프트웨어 모듈의 위험수준을 예측한다.

본 논문은 총 6장으로 구성되어 있다. 2장에서는 본 연구와 관련된 선행 연구들에 대한 내용을 제시한다. 3장에서는 SVM에 대한 기본적인 내용을 보이고, 4장에서는 소프트웨어 위험모듈 예측을 위한 절차를 제시한다. 5장에서는 실험을 위한 환경 및 데이터와 실험의 결과를 보여준다. 6장에서는 연구결과 요약 및 향후 연구계획으로 결론을 맺는다.

2. 관련 연구

2.1 소프트웨어 메트릭스

지금까지 소프트웨어 메트릭스와 소프트웨어의 결함

과의 연관성에 관한 많은 연구가 이루어져 왔다. 소프트웨어 메트릭스를 이용하여 소프트웨어의 결함을 예측하는 방법은 통계적인 기술을 이용한 방법과 기계 학습을 이용한 방법으로 나눌 수 있다. 통계적인 기술을 이용한 방법으로는 Discriminant Analysis[2]와 Factor Analysis[3]등이 있으며, 기계 학습을 이용한 방법으로는 Decision Trees[4], Artificial Neural Networks[5], Support Vector Machine(SVM)[6] 등을 이용한 방법이 있다. 이 연구들의 많은 부분은 어떤 소프트웨어 메트릭스가 소프트웨어의 결함과 가장 많은 연관성이 있는지를 다루고 있다. 어떤 소프트웨어 메트릭스의 조합이 소프트웨어 모듈의 결함을 가장 잘 표현하는지는 다소 복잡한 문제이다.

2.2 SVM을 이용한 결점보유모듈 예측 모델

SVM을 소프트웨어 모듈의 결점보유여부를 예측하는 데에 활용한 연구로는 Gondra의 연구[7]와 Elish의 연구[8]가 있다. Gondra의 연구에서는 NASA에서 제공하는 소프트웨어 시험데이터 집합[1] 중에서 JM1 데이터 집합을 이용하여 SVM이 ANN(Artificial Neural Network)보다 결함보유모듈 예측에 뛰어남을 보여주었다. 또한, 각 소프트웨어메트릭스에 대하여 결함모듈 예측에 미치는 영향에 대하여 민감도 분석을 수행하였다. Elish는 NASA에서 제공해주는 데이터 집합 중에서 CM1, PC1, KC1, KC3의 데이터 집합을 이용하여, SVM을 포함하여 9개의 분류알고리즘으로 결함모듈예측을 수행하였다. 그 결과 SVM이 다른 알고리즘들에 비해 성능이 좋음을 보여주었다.

3. Support Vector Machine(SVM)

3.1 개요

Support Vector Machine(SVM)은 1995년 Vapnik에 의해 소개된 커널 기반의 학습 알고리즘이다[9]. SVM은 일반화의 오류를 줄일 수 있는 구조적 위험 최소화(Structural Risk Minimization: SRM) 원칙을 사용하였으며, 적은 학습 데이터를 이용해서도 높은 차원의 공간을 일반화할 수 있다. 일반화란 학습과정에서 이용되지 않은 새로운 데이터 표본에 대해서도 올바른 분류가 될 수 있게 하는 것을 말하는데 분류기의 경우 일반화 성능이 높아야 한다.

그림 1은 2차원의 특징 공간(feature space)에서 두 개의 클래스로 분류하기 위한 초평면(hyperplane)을 보여준다. SVM에서는 초평면에서 가장 가까운 학습 데이터까지의 거리를 최대화시킬 수 있는 최적평면(optimal hyperplane)을 찾는다. 이 거리(maximum margin)가 클수록 분류기의 복잡도가 낮아지고 분류 성능이 좋아진다. 이 최적의 초평면은 $w \cdot x+b=0$으로 표현될 수 있다. 각 학습 데이터 x_i에 대해 $y_i \in \{+1, -1\}$가 결과값으로 주어진다. 최적의 평면을 찾기 위해서는 다음의 수식을 최소화해야 한다.

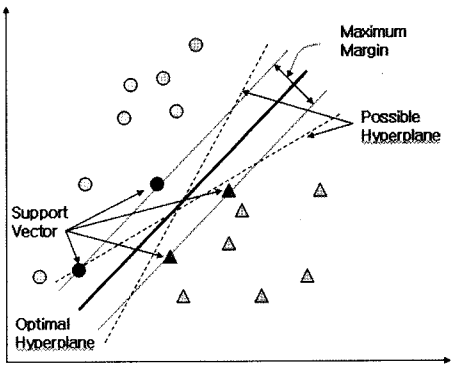


그림 1 SVM을 이용한 최적 분류 평면

$$\min_{w,b} \frac{1}{2}\|w\|^2, \quad y_i(w \cdot x_i+b) \geq 1, i=1,2,\cdots,n \tag{1}$$

하지만 일반적으로 대부분의 입력 데이터들은 이진 분류방법에 의해 오류없이 분류되지 않는다. 이러한 경우는 오류를 최소화하여 학습 데이터 집합을 분류하는 것이 목표가 된다. 이를 위해 양의 값을 가지는 슬랙 변수(ξ)와 패널티 함수(C)를 도입하였다. 이 경우에는 다음과 같은 수식이 최소화되도록 최적화되어야 한다.

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i, \quad y_i(w \cdot x_i+b) \geq 1-\xi_i, i=1,2,\cdots,n \tag{2}$$

패널티 함수 C는 마진의 최대화와 분류오류의 최소화 사이에서 트레이드 오프 관계를 가지는 조절 변수이다. 즉, C는 분리되지 않는 데이터에 대해 패널티로 작용하는 변수로써 C의 값이 커지면 최적의 평면을 제공하여 분류오류를 최소화시키려는 경향이 있으며, C의 값이 작아지면 마진을 극대화시키려는 조건으로 최적화하려는 경향이 있다.

SVM이 선형적으로 이진 분류를 하지 못하는 경우 비선형 사상 ϕ를 이용하여 입력 벡터의 차원보다 높은 선형분류가 가능한 차원으로 변형한 후 선형 분류를 하게 된다[9]. 비선형 사상은 커널 함수를 통해 이루어지는데 커널 함수는 SVM을 이용한 분류의 성능을 결정

1) NASA의 "IV&V Facility Metrics Data Program(MDP)"의 목적은 소프트웨어 메트릭스 데이터를 수집하고 정리하고 저장하는 것이 목적으로 일반에게 데이터를 개방하고 있다.

짓는 매우 중요한 변수 중의 하나이다. 커널 함수를 지원해주는 커널로는 Linear, Polynomial, Gaussian, Sigmoid가 있다.

3.2 입력변수의 선정

위험도가 높은 소프트웨어 모듈을 찾기 위해 사용될 입력변수의 선정은 매우 중요하다. 기계학습 알고리즘의 설계에서 입력변수의 선정은 매우 중요한 부분이다. 대표적인 입력변수 선정방법으로는 Principal Component Analysis(PCA)가 있다[7]. PCA 방법의 단점은 원래의 입력변수들과 비교해볼 때 유도된 디멘젼들이 직관적으로 해석이 되지 않는다는 것이다. 다른 방법으로는 Gondra의 연구에서 입력변수의 선정을 위해 사용한 방법으로 각각의 입력 변수들의 중요성을 계산하는 민감도 분석이 있다. 각 입력변수 값에 대해 결과 값이 변화하는 정도를 정량화하여 각 입력변수의 중요도를 판단한다[7].

Elish는 입력변수들 간의 연관성을 기반으로 하여 선정하는 기술을 이용하였다. 이 방법은 correlation-based feature selection technique(CFS)라 불린다[8]. 입력변수들의 모든 조합을 이용하여 어느 조합이 예측을 위해 가장 적합한지를 찾는다. 결과 예측을 위해 사용되는 입력 값들 사이의 중복되는 정도를 고려하여 최선의 입력 값들의 조합을 찾아낸다. 표 1은 대표적인 소프트웨어 메트릭스를 보여준다.

표 1 소프트웨어 모듈의 메트릭스

소프트웨어 메트릭스	설명
LOC	Total number of lines
LOCc	Number of comment-only lines
LOCb	Number of blank lines
LOCcc	Number of lines of code and comment
LOCs	Number of lines of statement
DC(m)	McCabe's design complexity
CC(m)	McCabe's Cyclomatic Complexity
EC(m)	McCabe's Essential Complexity
PD(h)	Halstead's program difficulty
PL(h)	Halstead's program length
V(h)	Halstead's volumn
OO(h)	Halstead's total number of operands and operators
IC(h)	Halstead's Intelligent count
EWP(h)	Halstead's effect to write program
T(h)	Halstead's time to write program
EE(h)	Halstead's effort Estimate
BR	Number of branches
UOpr	Number of unique operators
UOpd	Number of unique operands
Opr	Number of operators
Opd	Number of operands
Risk Level	Risk level (from requirement specification)

4. 위험모듈 분류절차

그림 2는 예측되는 위험수준이 높은 모듈을 선별해내기 위한 기본적인 절차를 보여준다. 먼저 해당 소프트웨어 어플리케이션의 특성을 조사하여 어떤 학습데이터 집합을 이용할 것인지를 결정한다. 어플리케이션의 특성은 여러가지가 있을 수 있다.

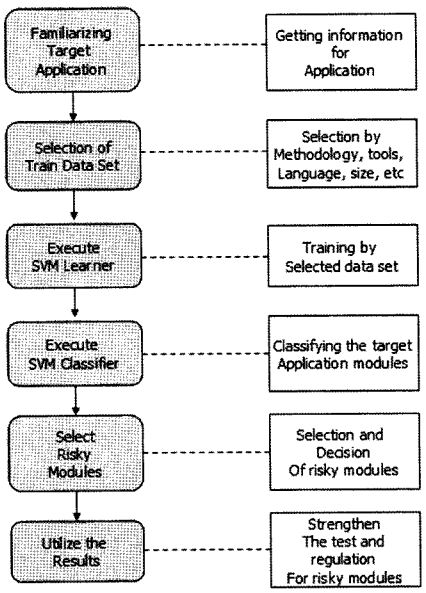


그림 2 기본적인 위험모듈 분류 절차

개발방법론, 개발도구, 개발언어, 프로그램 규모, 운영환경 등에 따라 알맞은 학습데이터 집합을 선택할 수 있다. 학습데이터 집합이 결정되면 SVM Learner를 통해 학습을 시키고, 해당 데이터 모델을 만든다. 타겟 소프트웨어의 각 모듈에 대한 메트릭스를 구하고 생성된 모델을 이용하여 잠재위험수준을 예측한다.

5. 실 험

5.1 데이터 및 실험환경

본 연구에서는 Elish와 Gondra의 연구와 마찬가지로 NASA에서 제공하는 시험 데이터 집합을 이용하였다. NASA의 데이터 집합 중에서 요구사항의 위험수준 정보가 제공된 CM1과 PC1 데이터를 이용하였다. CM1은 20KLOC의 C언어로 개발된 NASA 항공기의 계측 시스템에 대한 소프트웨어 메트릭스와 오류정보에 대한 데이터를 보유하고 있다. PC1은 40KLOC의 C언어로 개발되었으며, 인공위성의 비행 소프트웨어이다. 각 데이터 집합은 요구 사항에 대한 위험도 수준을 1부터 3으로 표시하고 있으며, 해당 요구사항과 구현된 모듈과

표 2 소프트웨어 메트릭스의 선택

시험 집합	소프트웨어 메트릭스
Case 1	CM1 : LOC, DC(m), D(h), LOCc, LOCb, Risk Level
	PC1 : CC(m), LOCcc, LOCc, LOCb, UOpr, Risk Level
Case 2	LOC, CC(m), BR, UOpr, PL(h), DC(m), D(h), Risk Level
Case 3	Case1과 Case2
Case 4	21개의 메트릭스와 Risk Level

의 관계 정보를 제공하고 있다. 실험을 위하여 표 2와 같이 네 가지의 시험 집합을 구성하여 각각 다른 소프트웨어 메트릭스를 적용하여 실험을 하였다. Case 1은 Elish의 연구에서 correlation-based feature selection technique(CFS)을 이용하여 선택한 메트릭스 값들을 이용하였다[8]. Case 2는 Gondra의 연구에서 민감도 분석을 통해 선택한 특성값들을 사용하였다[7].

이 경우는 민감도 값이 가장 높게 나타난 메트릭스 7개를 입력변수로 선택하였다. Case 3에서는 Case 1과 Case 2에서 사용한 메트릭스를 모두 입력변수로 채택하였으며, Case 4에서는 NASA의 데이터 집합에서 제공해주는 21개(표 1 참조)의 메트릭스를 입력변수로 선택하였다. 그리고, 요구사항에 대한 위험수준을 모든 시험 케이스에 하나의 입력 변수로 추가하였다. 하나의 모듈이 여러 개의 요구사항을 구현하고 있는 경우에는 여러 요구사항들 중에서 가장 높은 위험 수준을 가진 것을 찾아 해당 모듈에 할당하였다. NASA에서 제공한 데이터 집합에는 요구사항의 위험 수준이 1부터 3까지의 수로 표현되어 있었다(클수록 위험이 높음). 학습 데이터 집합에 있는 모듈의 위험수준은 요구사항의 위험 수준 ($Risk\_Level_r$)과 구현된 모듈에서 발생한 오류 ($Error\_Count_m$)의 수를 이용하여 1부터 5까지의 예측위험수준으로 분류하여 학습하였다. 모듈의 예측위험수준 Rm은 다음과 같이 정의하였다.

$$R_i = Risk\_Level_r + Risk\_Level_r * Error\_Count_m \quad (3)$$

$$R_m = \begin{cases} Risk\_Level_r & R_i \le \alpha \\ Risk\_Level_r + 1 & \alpha < R_i \le \beta \\ Risk\_Level_r + 2 & R_i > \beta \end{cases} \quad (4)$$

본 실험에서는 R_i의 값이 4(α)이상 10(β)이하이면, 예측위험수준(R_m)을 4로, 10이상이면 예측위험수준 (R_m)이 5의 값을 가지도록 하였다. 시험환경은 *MATLAB*과 $SVM^{multiclass}$ *Version 2.12*을 이용하였다[12]. 마진 크기와 학습 오류사이의 트레이드오프로 사용되는 C값은 5000을 이용하였으며, Linear 커널을 이용하여 시험하였다. CM1은 총 507개의 모듈로 구성되어 있으며, 그 중 254개는 학습 데이터로 나머지 253개는 시험 데이터로 사용하였다. PC1은 총 1108개의 모듈로 구성되어 있다. 그 중에서 절반인 554개를 학습 데이터로 나머지를 시험 데이터로 사용하였다. 각각의 데이터 집합의 모듈들에 대하여 위험클래스의 분포는 표 3과 같다.

표 3 데이터 모듈의 잠재위험 분포

잠재위험 수준	학습 데이터 집합		시험 데이터 집합	
	CM1	PC1	CM1	PC1
1	199 (78.35%)	464 (83.91%)	201 (79.45%)	470 (84.99%)
2	26 (10.24%)	55 (9.95%)	31 (12.25%)	41 (7.41%)
3	10 (3.94%)	14 (2.35%)	11 (4.35%)	33 (5.97%)
4	16 (6.30%)	13 (2.35%)	6 (2.37%)	8 (1.45%)
5	2 (0.79%)	7 (1.27%)	3 (1.19%)	2 (0.36%)

5.2 시험결과

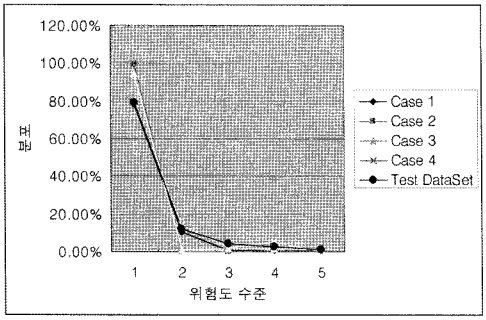
본 실험의 목적은 다른 모듈들에 비해 상대적으로 높은 잠재위험수준을 가지는 모듈들을 예측하는 것이다. 잠재위험수준이 4 혹은 5로 분류된 모듈들은 그렇지 않은 모듈들에 비해 상대적으로 위험수준이 높다라고 예측할 수 있다. 본 실험에 의하면 각 시험 집합 별로 분류오류율(zero/one-error)은 CM1 데이터 집합의 경우 Case 1이 22.13%, Case 2가 21.34%, Case 3은 23.72%, 그리고 Case 4는 35.18%가 나왔다. PC1 데이터 집합은 분류 예측결과 Case 1의 경우는 15.52%, Case 2는 16.06%, Case 3은 16.43%, 그리고 Case 4는 26.90%의 오류율(zero/one-error)이 나왔다.

분류결과 앞의 세가지 케이스의 경우는 분류의 오류율은 비슷하나 21개의 소프트웨어 메트릭스로 학습을 시킨 Case 4의 경우는 오류율이 높게 나왔다. 본 실험에서는 해당 시험 집합이 가지는 오류율보다는 잠재위험모듈 4와 5로 예측되는 모듈들에 대한 변별력에 초점을 두었다. 그림 3은 시험케이스별로 예측된 각 모듈들의 잠재위험수준 분포비율을 보여준다. 잠재위험수준 4와 5를 구별해내기 위해 가장 적합한 시험케이스는 CM1의 경우 Case 3과 Case 4, PC1의 경우는 Case 1의 경우로 나타났다. 하지만, 감내할 수 있는 분류 오류의 정도에 따라 여러가지 선택이 가능할 것으로 본다.

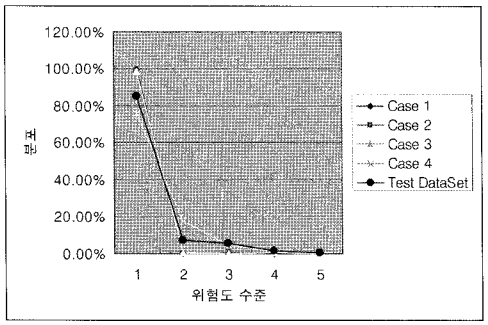
예측된 잠재위험수준이 4 혹은 5에 속하는 모듈들은 그렇지 않은 모듈들에 비해 상대적으로 높은 위험도를 가지고 있다고 예측할 수 있다. 이 실험에 의하면 상대적으로 잠재위험수준이 높은 모듈들을 설계 및 구현단계에서 선별해 낼 수 있으며, 테스팅과 확인 및 검증 그리고 규제기관에서의 주요검토 대상 모듈 선정 시에 활용할 수 있을 것으로 기대된다.

6. 결 론

원전에서 사용되는 안전-필수 소프트웨어는 안전성과



CM1 데이터 집합의 시험결과



PC1 데이터 집합의 시험결과

그림 3 잠재위험 모듈 예측 결과

신뢰도의 보장이 매우 중요하다. 본 논문에서는 SVM을 이용하여 안전-필수 소프트웨어를 구성하는 모듈들 중에서 잠재위험이 높은 모듈을 소프트웨어의 개발주기 초기에 선별할 수 있는 예측시스템을 제시하였다.

본 시스템에서 예측되는 잠재위험수준은 절대적인 값으로 사용되기보다는 상대적인 값으로 사용된다. 높은 위험클래스로 분류된 모듈에 대해 집중적으로 테스팅과 확인 및 검증을 위한 자원을 투자하여 효율적인 소프트웨어의 품질 관리가 가능할 것으로 본다. 또한, 규제측면에서의 심사 대상모듈 선정시에도 활용이 가능할 것으로 본다. 하지만, 이 시스템을 보다 실용적으로 활용하기 위해서는 다음과 같은 문제점이 있다. 먼저, 입력변수(특징 공간)의 선택 및 변수값의 선정, 커널의 선정 등에 따라 결과가 달라진다. 또한, 분류의 일반화의 오류를 줄이기 위해서는 적용하고자하는 소프트웨어의 데이터 집합과 유사한 학습 데이터를 구해야 한다는 한계점이 있다. 향후에는 위의 문제점들을 해결하기 위한 연구와 함께 다른 학습알고리즘에 적용하여 결과를 비교분석하는 연구도 진행되어야 한다.

참 고 문 헌

[1] Boehm, B.W., Papaccio, P.N., Understanding and controlling software costs. IEEE Transactions on Software Engineering, Volume 14, Issue 10, pp. 1462-1477, 1988.

[2] Briand, L.T., Basili, V.R., Hetmanski, C., Developing interpretable models for optimized set reduction for identifying high-risk software components. IEEE Transactions on Software Engineering, Volume 19, Issue 11, pp. 1028-1034, 1993.

[3] Khoshgoftaar, T.M., Munson, J.C., Prediction software development errors using complexity metrics. IEEE Journal on Selected Areas in Communications, Volume 8, Issue 2, pp. 153-261, 1990.

[4] Porter, A., Selby, R., Empirically guided software development using metric-based classification trees. IEEE Software, Volume 7, Issue 2, pp. 46-54, 1990.

[5] Khoshgoftaar, T.M., Lanning, D.L., Pandya, A.S., A comparative study of pattern recognition techniques for quality evaluation of telecommunications software, IEEE Journal on Selected Areas in Communications, Volume 12, Issue 2, pp. 208-217, 1994.

[6] Xing, F., Guo, P., Lyu, M.R., A novel method for early software quality prediction based on support vector machine, Proceedings of IEEE International Conference on Software Reliability Engineering, pp. 213-222, 2005.

[7] I. Gondra, Applying machine learning to software fault-proness prediction, The Journal of Systems and Software, Volume 81, Issue 2, pp. 186-195, 2008.

[8] Karim O. Elish, Mahmoud O. Elish, Predicting defect-prone software modules using support vector machines, The Journal of Systems and Software, Volume 81, Issue 5, pp. 649-660, 2008.

[9] Burges, C., A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998.

[10] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods-Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[11] Steve R, Gunn, Support Vector Machines for Classification and Regression, Technical Report, University of Southampton, 10 May 1998.

[12] http://svmlight.joachims.org/svm_multiclass.html