신뢰성응용연구
제16권, 제1호. pp.48-55, 2016

소프트웨어 신뢰성 예측을 위한 객체지향 척도 분석*

이 양 규†

서원대학교 글로벌경영대학 유통경영정보학과

Analysis of Object-Oriented Metrics to Predict Software Reliability

Yangkyu Lee†

Department of Distribution & Management Information Systems Seowon University

Purpose: The purpose of this study is to identify the object-oriented metrics which have strong impact on the reliability and fault-proneness of software products. The reliability and fault-proneness of software product is closely related to the design properties of class diagrams such as coupling between objects and depth of inheritance tree.
Methods: This study has empirically validated the object-oriented metrics to determine which metrics are the best to predict fault-proneness. We have tested the metrics using logistic regressions and artificial neural networks. The results are then compared and validated by ROC curves.
Results: The artificial neural network models show better results in sensitivity, specificity and correctness than logistic regression models. Among object-oriented metrics, several metrics can estimate the fault-proneness better. The metrics are CBO (coupling between objects), DIT (depth of inheritance), LCOM (lack of cohesive methods), RFC (response for class). In addition to the object-oriented metrics, LOC (lines of code) metric has also proven to be a good factor for determining fault-proneness of software products.
Conclusion: In order to develop fault-free and reliable software products on time and within budget, assuring quality of initial phases of software development processes is crucial. Since object-oriented metrics can be measured in the early phases, it is important to make sure the key metrics of software design as good as possible.

Keywords: Object-Oriented Metrics, Software Fault-Proneness, Software Reliability, ROC analysis, Logistic Regression, Artificial Neural Network

1. 서 론

소프트웨어 개발과정에서 초기단계의 품질은 소프트웨어 개발과정 전체에 큰 영향을 미친다. Boehm [1]은 소프트웨어 개발 과정에 있어서 초기 단계에 개발된 산출물의 오류는 사후 단계에도 계속적으로 영향을 주어서 오류를 발견하고 수정하는데 후기 단계의 오류보다 더 많은 비용이 소요된다고 하였다. 따라

† 교신저자 yangkyu.lee@gmail.com
2016년 3월 2일 접수; 2016년 3월 8일 수정본 접수; 2016년 3월 16일 게재 확정.

서 소프트웨어 개발 초기단계에서 산출물의 품질특성을 평가하여 이를 개선할 수 있는 지침을 활용하는 것은 매우 유용하다. 소프트웨어의 신뢰성은 소프트웨어 품질특성 중에서도 매우 중요하며 신뢰성 저하로 인한 소프트웨어의 작동오류 및 실행중단 등에의한 손실의 심각함은 더 이상 언급할 필요가 없다.

현재 소프트웨어 개발은 객체지향 방법론이 정착되어 있으며 Unified Modeling Language(UML)를 이용한 개발이 산업 표준으로 정착되었다. UML의 초기 단계 핵심 산출물은 클래스 다이어그램(class diagram)과 유스케이스 다이어그램(use case diagram)이다. 클래스와 유스케이스 다이어그램에 대한 품질 척도는 여러 연구자들에 의하여 개발되었다. 이러한 다이어그램에 대한 척도는 오류가 발생할 가능성이 많은 설계상의 문제점을 진단하게 하고, 대안 설계를 모색하게 하며, 유지보수성, 신뢰성 등 품질특성을 예상할 수 있게 한다.

기존의 연구에서는 클래스 다이어그램이나 유스케이스 다이어그램 각각에 대한 품질 척도를 개발하였으며 대체적으로 클래스 기준의 척도를 개발하였다. 본 연구에서는 클래스 다이어그램 척도를 실증적으로 분석하여 소프트웨어 개발 초기단계에서 제품의 신뢰성을 향상시키기 위한 주요 설계특성을 제시하려고 한다.

2. 관련 연구

2.1 클래스 다이어그램 품질 척도

Chidamber and Kemerer[2]는 객체지향 설계에 대한 여섯 개의 척도를 제시하였다. 이 척도의 목적은 설계 복잡성을 측정하고 유지보수성, 재사용성 등의 품질특성과의 관련성을 밝히려는 것이다. 제안된 여섯 개의 척도는 크게 클래스 상속 구조, 클래스의 복잡도, 클래스 간 연관관계로 구분할 수 있다.

클래스 상속 구조에 대한 척도는 클래스 상속 트리 구조의 깊이를 측정하는 DIT(Depth of Inheritance Tree), 그리고 하위 클래스의 개수를 측정하는 NOC (Number of Children)이다. DIT는 "클래스 상속의 깊이"로 정의한다. 이는 클래스에 영향을 미치는 상위 클래스의 수를 측정한다. NOC는 "클래스에 관련된 하위 클래스의 수"로 정의한다.

클래스의 복잡도 척도는 클래스 당 가중치가 부여된 메서드의 개수를 측정하는 WMC(Weighted Methods per Class), 전달되는 메시지의 수를 측정하는 RFC(Response For Class), LCOM(Lack of COhesion in Methods)이다. WMC는 "클래스 메서드의 복잡도의 합"으로 정의된다. 시스템 설계 수준에서는 각 메서드의 복잡도를 측정할 수 없으므로 가중치를 단위 값인 1로 설정하여 클래스의 메서드의 수를 WMC로 산정한다. RFC는 "클래스 객체가 받은 메시지에 대한 응답으로 수행될 가능성이 있는 메서드의 수"로 정의된다. LCOM은 "클래스의 필드를 서로 공유하지 않아서 서로 관련되지 않는 메서드의 수"로 정의된다.

클래스의 연관관계 척도는 클래스간의 결합도를 측정하는 CBO(Coupling Between Objects)가 있다. CBO는 메서드 호출, 필드 액세스, 상속, 인수, 반환 유형, 예외 등의 결합을 통하여 특정 클래스에 결합된 클래스의 수를 측정한다. 이것은 몇 개의 실증적 연구를 통하여 유용성이 검증되어 추후에 연구에서 척도의 근거로 많이 사용되었다. Daly 외[3]는 상속성의 깊이(DIT)가 3인 시스템의 유지보수성이 상속이 없는 시스템보다 높아진다고 주장하였고, Briand 외[4]는 DIT가 커지면 오류가 발생할 가능성이 높아지며 NOC가 커지면 오류 가능성이 작아진다고 주장하였다. Cartwright[5]는 DIT가 3일 때 시스템 유지보수에 긍정적인 결과를 가져온다고 주장한다.

Li and Henry[6]는 클래스 수준에서 결합도(coupling), 복잡성 및 규모를 측정하는 척도를 제시하였다. 척도는 다른 클래스를 유형으로 갖는 속성의 개수(DAC), 속성으로 사용된 서로 다른 클래스의 개수(DAC'), 로컬 메서드의 개수(NOM), 속성과 로컬 메서드의 수의 합계(SIZE2) 등이다.

Meol 외[7]에 의한 MOOD(Metrics for Object Oriented Design) 척도는 MOOD2 척도로 개선되었는데 이것은 클래스 다이어그램 수준뿐만 아니라 상속성 척도, 정보은닉 척도, 다형성 척도 등이 포함되었다. 여기에는 MHF(메서드 은닉 요소 : Method Hiding Factor), AHF (속성 은닉 요소 : Attribute Hiding Factor), MIF(메서드 상속 요소 : Method Inheritance Factor), AIF(속성 상속 요소 : Attribute Inheritance Factor), PF(다형성 요소 : Polymorphism Factor) 등이 있다. MHF는 모든 클래스에 정의된 메서드 중에서 은닉된(private이나 protected로 선언) 메서드와 시스템에서 정의된 모든 메서드의 비율을 계산한다. 즉, MHF가 높다는 것은 메

서드 은닉 정도가 높다는 것을 의미한다. AHF는 모든 클래스에서 정의된 모든 속성 중에서 은닉된(private 또는 protected) 속성과 시스템 전체의 속성과의 비율이다. MIF는 시스템의 모든 클래스에서 상속된 메서드의 개수와 총 메서드의 수와의 비율이다. AIF는 시스템의 모든 클래스에서 상속된 속성과 모든 클래스의 모든 속성의 개수와의 비율이다. PF는 실제 다형성의 수와 가능한 다형성의 최대 개수와의 비율이다. 실증연구[7]에서는 MHF와 MIF가 증가되면 오류 및 이를 수정하기 위한 노력의 심각성이 감소되어야 한다고 하고, 모든 속성은 은닉되는 것이 이상적이기 때문에 AHF는 100%가 되어야 좋다고 한다.

Briand 외[8]는 클래스간의 결합도 척도를 제안하였다. 실증연구 결과 결합도가 객체지향 시스템의 품질을 결정하는데 중요한 요소임을 주장하였으며, 다른 클래스를 호출하는 결합도 관계가 오류 발생 가능성과 관계가 크다고 주장하였다.

그밖에도 Lorenz and Kidd[9]는 클래스 규모 척도, 클래스 상속 척도, 클래스 내부 척도 등을 제안하였으며, Bansiya and Davis[10]은 클래스 수준에서 캡슐화(encapsulation), 결합도(coupling), 응집도(cohesion), 구성(composition), 상속성(inheritance)을 측정하는 척도를 제안하였다. Malhotra 외[11]는 오픈소스 소프트웨어를 이용하여 Chidamber and Kemerer[2]의 객체지향 척도와 소프트웨어 오류발생 가능성의 연관성을 검증하였다.

3. 객체지향 척도 분석

3.1 분석 방법

본 연구에서는 척도와 오류의 개수에 대한 분석을 실시하여 오류의 발생빈도와 객체지향 척도 간의 관계를 파악하였다. 오류와 척도와의 관계를 파악하기 위해서 본 연구에서는 로지스틱 회귀분석과 인공신경망을 이용하였다. 우선적으로 로지스틱 회귀분석을 이용하여 중요한 척도를 파악하였으며 추출된 척도를 로지스틱 회귀분석과 인공신경망에 각각 적용하여 어느 분석 방법이 오류 가능성에 대한 변별력이 높은지를 검증하였다

두 가지 방법 중에서 어떠한 것이 더 유용한 것인지를 검증하기 위하여 본 연구에서는 ROC 분석을 실시하였다. 이를 통하여 오류가 발생하였는데 오류가 있을 것이라고 예측한 비율(민감도), 오류가 없는데 없다고 정확하게 예측한 비율(특이도)을 비교 분석하여 로지스틱 회귀모형과 인공신경망 모형을 검증하였다. ROC 분석을 실시하기 위하여 클래스의 오류의 개수가 아니라 오류가 발생하였는지 아닌지를 가지고 판단하였다. 따라서 오류의 개수가1 이상이면 오류발생 오류의 개수가0이면 오류 없음으로 재 코딩하였다.

ROC(Receiver Operating Characteristics) 분석은 민감도를 y축에 1-특이도를 x축에 플롯 하여 모델의 예측성과를 판단하기 위한 분석방법이다. 민감도와 특이도는 모델의 정확도를 측정하기 위한 것이다. 민감도는 오류가 있을 것을 올바르게 예측한 비율이고, 특이도는 오류가 없는 것을 올바르게 없는 것으로 예측한 비율이다. 좋은 예측모델이 되려면 이 두 가지의 비율이 높아야 한다. 1-특이도는 오류가 없는데 있다고 예측한 비율이다. 이를 간략하게 설명하면 아래와 같다.

		실제 클래스	
		오류 있음	오류 없음
예측모델	오류예측	True Positive (TP)	False Positive (FP)
	오류 없음 예측	False Negative (FN)	True Negative (TN)
	열의 합	P	N

$fprate = \frac{FP}{N}$, 민감도 또는 $tprate = \frac{TP}{P}$,

특이도 $= \frac{TN}{FP+TN}$,

정밀도$(\mathrm{Pr}ecision) = \frac{TP}{TP+FP}$,

정확도$(accuracy) = \frac{TP+TN}{P+N}$

3.2 분석 결과

본 연구에서는 NASA의 KC1 데이터를 이용하였다[12]. 이 데이터는C++로 개발된 프로젝트에서 얻은 것이며 여기에는 총145개의 클래스가 있으며 오류가 있는 클래스는 60개이며 오류가 없는 클래스는 85개이다. 사용된 클래스 척도는 아래와 같다. 즉, 클래스내의 퍼블릭 데이터의 비율(PPD: Percent of Public Data), 객체간의 결합도(CBO: Coupling Between Objects), 상속의 수(DIT: Depth of Inheritance Tree), 응집력 있는 메서드의 부재(LCOM: Lack of Cohesion Methods), 하위 클래스의 수(NOC: Number of Children), 하위클래스 종속도

Table 1 Descriptive statistics of object-oriented metrics

Descriptive Statistics						
	N	Range	Minimum	Maximum	Mean	Std. Deviation
PPD	145	100	0	100	14.40	32.533
CBO	145	24	0	24	8.32	6.377
DIT	145	6	1	7	2.00	1.258
LCOM	145	100	0	100	68.72	36.889
NOC	145	5	0	5	.21	.699
DOC	145	1	0	1	.01	.117
FAN	145	3	0	3	.63	.695
RFC	145	222	0	222	34.38	36.203
WMC	145	100	0	100	17.42	17.449
LOC	145	61	0	61	11.83	12.11
Valid N (listwise)	145	-	-	-	-	-

Table 2 Correlation among metrics

Correlations											
		PPD	CBO	DIT	LCOM	NOC	DOC	FAN	RFC	WMC	LOC
PPD	Pearson Correlation	1	.046	-.071	.265**	.127	-.053	-.011	.057	.147	.036
	Sig. (2-tailed)	-	.580	.399	.001	.128	.530	.893	.494	.078	.668
	N	145	145	145	145	145	145	145	145	145	145
CBO	Pearson Correlation	.046	1	.470**	.257**	-.032	.180*	.460**	.387**	.245**	.678**
	Sig. (2-tailed)	.580	-	.000	.002	.698	.030	.000	.000	.003	.000
	N	145	145	145	145	145	145	145	145	145	145
DIT	Pearson Correlation	-.071	.470**	1	.257**	-.032	.000	.746**	.654**	.136	.550**
	Sig. (2-tailed)	.399	.000	-	.002	.706	1.000	.000	.000	.102	.000
	N	145	145	145	145	145	145	145	145	145	145
LCOM	Pearson Correlation	.265**	.257**	.257**	1	-.028	.101	.233**	.334**	.318**	.148
	Sig. (2-tailed)	.001	.002	.002	-	.735	.229	.005	.000	.000	.075
	N	145	145	145	145	145	145	145	145	145	145
NOC	Pearson Correlation	.127	-.032	-.032	-.028	1	.133	-.081	-.050	.036	-.001
	Sig. (2-tailed)	.128	.698	.706	.735	-	.110	.333	.554	.669	.989
	N	145	145	145	145	145	145	145	145	145	145
DOC	Pearson Correlation	-.053	.180*	.000	.101	.133	1	.062	.012	-.030	.034
	Sig. (2-tailed)	.530	.030	1.000	.229	.110	-	.456	.887	.720	.682
	N	145	145	145	145	145	145	145	145	145	145
FAN	Pearson Correlation	-.011	.460**	.746**	.233**	-.081	.062	1	.529**	.106	.413**
	Sig. (2-tailed)	.893	.000	.000	.005	.333	.456	-	.000	.204	.000
	N	145	145	145	145	145	145	145	145	145	145
RFC	Pearson Correlation	.057	.387**	.654**	.334**	-.050	.012	.529**	1	.628**	.301**
	Sig. (2-tailed)	.494	.000	.000	.000	.554	.887	.000	-	.000	.000
	N	145	145	145	145	145	145	145	145	145	145
WMC	Pearson Correlation	.147	.245**	.136	.318**	.036	-.030	.106	.628**	1	.092
	Sig. (2-tailed)	.078	.003	.102	.000	.669	.720	.204	.000	-	.271
	N	145	145	145	145	145	145	145	145	145	145
LOC	Pearson Correlation	.036	.678**	.550**	.148	-.001	.034	.413**	.301**	.092	1
	Sig. (2-tailed)	.668	.000	.000	.075	.989	.682	.000	.000	.271	-
	N	145	145	145	145	145	145	145	145	145	145

**. Correlation is significant at the 0.01 level (2-tailed)
*. Correlation is significant at the 0.05 level (2-tailed)
N = 145

Table 3 Univariate analysis of LR(Logistic Regression)

Metric	Coeff	SE	Sig.
PPD	0.01325	0.01843	0.0203
CBO	0.59503	0.08173	2.09E-11
DIT	0.7492	0.4822	0.1225
LCOM	0.02371	0.01646	0.1521
NOC	-1.3406	0.8665	0.124
DOC	-0.4507	5.2173	0.931
FAN	2.4719	0.8589	0.00448
RFC	0.04999	0.1653	0.00295
WMC	0.1292	0.03635	0.000516
LOC	0.28833	0.07094	7.91E-05

Table 4 Result of LR(Model 1)

Variables in the equation							
		B	S.E.	Wald	df	Sig.	Exp (B)
Step 1[a]	CBO	.207	.037	31.811	1	.000	1.231
	Constant	-2.167	.388	31.162	1	.000	.115
Step 2[b]	CBO	.306	.053	33.105	1	.000	1.358
	DIT	-.724	.221	10.772	1	.001	.485
	Constant	-1.584	.440	12.955	1	.000	.205

a. Variable(s) entered on step 1 : Coupling_between_objects.
b. Variable(s) entered on step 2 : Depth.

(DOC: Dependent on Child), 상위모듈에서 호출되는 수(FAN: FAN-IN), 클래스에서 구현된 메서드의 수(RFC: Response for Class), 클래스 내에서 구현된 메서드의 수(WMC: Weighted Methods per Class), 평균 라인수(LOC: Average Lines of Code)이다. 각 척도에 대한 기술 통계량이 <Table 1>에 제시되었다. 척도 간의 상관계수는 <Table 2>에 제시되어 있다. 척도 간의 상관관계를 살펴보면 CBO, DIT, LCOM, FAN, RFC, LOC는 다른 척도와 상관관계가 매우 높고, PPD, NOC는 다른 척도와 상관관계가 낮은 것으로 파악되었다. 특히, LOC 척도는 객체지향 척도는 아니지만 상관관계가 매우 높기 때문에 본 연구에서는 이를 포함하여 분석하였다.

개별 척도와 오류의 수에 대한 회귀분석 결과 <Table 3>과 같이 유의한 척도로는 PPD, CBO, FAN, RFC, WMC, LOC로 파악되었다. 특히 CBO와 LOC와 오류의 관련성이 매우 높은 것으로 파악되었다 파악된 척도를 이용하여 단계적 회귀모형을 실행하였다.

Table 5 Correctness of LR(Model 1)

Classification Table[a]					
Observed			Predicted		
			Fault		Percentage
			0	1	Correct (%)
Step 1	Fault	0	67	18	78.8
		1	25	35	58.3
	Overall Percentage		-	-	70.3
Step 2	Fault	0	68	17	80.0
		1	21	39	65.0
	Overall Percentage		-	-	73.8

a. The cut value is 0.500

Table 6 Result of LR(Model 2)

Variables in the equation						
	B	S.E.	Wald	df	Sig.	Exp (B)
CBO	.294	.065	20.633	1	.000	1.341
DIT	-1.313	.322	16.642	1	.000	.269
LCOM	-.017	.007	5.430	1	.020	.983
RFC	.027	.011	6.485	1	.011	1.027
LOC	.061	.027	5.034	1	.025	1.063
Constant	-.896	.523	2.931	1	.087	.408

본 연구에서는 객체지향 척도가 아닌 LOC를 사용하지 않고 오류판별을 하는 것과 설명력이 매우 높은 LOC를 포함하여 분석하는 것의 장단점을 고려하여 두 가지의 모형을 사용하였다. 즉, LOC를 포함하지 않고 객체지향 척도만을 이용한 모형을 모형1로 여기에 LOC를 포함한 모형을 모형2라고 하고 각각에 대하여 분석을 실시하였다. 모형 1의 로지스틱 회귀모형의 분석결과는 <Table 4>와 같다. 최종적으로 남은 척도는CBO와DIT로서 <Table 5>와 같이 분류 정확도는73.8%이다. 즉, 오류가 있는 클래스를 오류가 있다고 판단하고, 오류가 없는 클래스를 오류가 없다고 판단하는 비율이 73.8%이다.

모형 2의 로지스틱 회귀분석 결과는 <Table 6>과 같다. LOC가 포함된 경우에는 CBO, DIT, LCOM, RFC, LOC의 척도가 종속변수로 설정되었다. 이 경우에는 <Table 7>과 같이 분류 정확도가 77.2%이다. 모형1의 회귀분석 결과보다 분류정확도가 상승된 것을 알 수 있다.

인공신경망 분석에서는 학습한 데이터를 검증에 사용하지 않고 데이터를 7대 3으로 분리하여 70퍼센

Table 7 Correctness of LR(Model 2)

Classification Table[a]					
Observed			Predicted		
			Fault		Percentage
			0	1	Correct (%)
Step 1	Fault	0	67	18	78.8
		1	25	35	58.3
	Overall Percentage		-	-	70.3
Step 2	Fault	0	68	17	80.0
		1	21	39	65.0
	Overall Percentage		-	-	73.8
Step 3	Fault	0	70	15	82.4
		1	20	40	66.7
	Overall Percentage		-	-	75.9
Step 4	Fault	0	72	13	84.7
		1	19	41	68.3
	Overall Percentage		-	-	77.9
Step 5	Fault	0	71	14	83.5
		1	19	41	68.3
	Overall Percentage		-	-	77.2

a. The cut value is 0.500

Table 8 Correctness of ANN(Model 1)

Classification				
Sample	Observed	Predicted		
		0	1	Percentage Correct
Training	0	49	8	86.0%
	1	8	30	78.9%
	Overall Percentage	60.0%	40.0%	83.2%
Testing	0	15	2	88.2%
	1	2	7	77.8%
	Overall Percentage	65.4%	34.6%	84.6%
Dependent variable : Fault				

트는 학습에 이용하고 30퍼센트는 검증에 이용하였다. 인공신경망 모형에 대한 분석의 경우에도 LOC가 포함되지 않은 모형 1과 LOC가 포함된 모형 2에 대하여 실시하였다.

모형 1에 대한 인공신경망 분석 결과는 <Table 8>과 같다. 오류가 있는 클래스와 오류가 없는 클래스를 정확하게 예측한 정확도는 86.4%로 로지스틱 회귀모형에 비하여 높은 수치를 보이고 있다.

모형 2에 대한 인공신경망 분석 결과는 <Table 9>에 제시되어 있다. 본 연구에서는 임의로 데이터를 분류하여 학습과 검증에 사용하기 때문에 인공신경망 분석을 할 때마다 정확도 결과는 차이가 발생하고 있다. 분석결과 <Table 9>와 같이 검정 정확도는100%로 제시되어 있지만 이것은 검증에 사용된 클래스가9개에 불과하기 때문이다. 그러나 여러 번의 분석을 실시한 결과 정확도 수치는 일관성 있게 매우 높은 수치를 보이고 있다.

본 연구에서는 두 가지 모형에 대한 로지스틱 회귀모형과 인공신경망 분석 결과의 우수성을 검증하기 위하여 ROC 곡선을 이용하였다. ROC는 오류가 있는 클래스를 오류가 있다고 판단하는 민감도(sensitivity)와 오류가 없는 클래스를 오류가 없다고 판단하는 특이성(specificity)을 그래프로 표시한 것인데 y축에는 민감도를 x축에는 1-특이성을 표시한다. 따라서 곡선의 형태가 왼쪽 위 방향으로 많이 치우칠수록 우수한 것이고 곡선 아래 영역의 값이 클수록 우수한 것이다 두 가지 모형에 대한 두 가지 방법 각각에 대한 ROC 곡선은 <Fig. 1> ~ <Fig. 4>와 같다. 각 모형에 대한 결과 요약이 <Table 10>에 제시되어 있다. 결과를 보면 모델 2에 대한 인공신경망 모형이 가장 우수한 것임을 알 수 있다. 이모형에 대한 민감도, 특이도, 정확도 값은 검증 케이스가 많아지면 낮아지겠지만 여러 차례의 실험결과 가장 우수한 것으로 나타났다. AUC (Area Under the Curve)는 곡선 아래의 영역 값으로 민감도와 특이도를 합친 값이다. 이 값이 클수록 정확한 예측을 의미한다. AUC 값이 0.9~1.0이면 매우 우수,

Table 9 Correctness of ANN(Model 2)

Classification				
Sample	Observed	Predicted		
		0	1	Percentage Correct
Training	0	60	4	93.8%
	1	9	36	80.0%
	Overall Percentage	63.3%	36.7%	88.1%
Testing	0	8	0	100.0%
	1	0	0	0.0%
	Overall Percentage	100.0%	0.0%	100.0%
Dependent variable : Fault				

0.8~0.9이면 우수, 0.7~0.8이면 적정, 0.6~0.7이면 미흡, 0.5~0.6이면 무의미이다.

Table 10 Results of two models

	Logistic Regression		ANN	
	Model 1	Model 2	Model 1	Model 2
cut-off	0.41	0.42	0.5	0.5
sensitivity	76.4	78.9	88.2	100
specificity	69.6	74.5	77.8	100
correctness	73.8	77.2	84.6	100
AUC	0.837	0.872	0.929	0.969

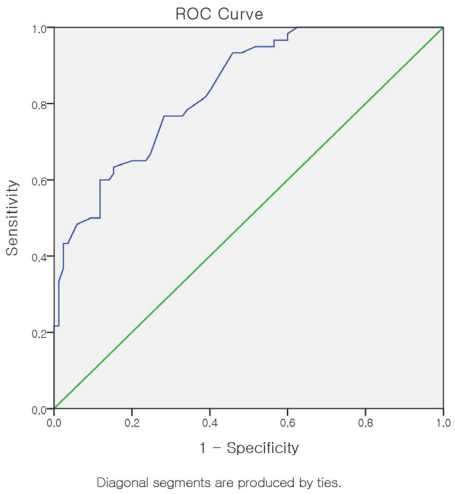

Fig. 1 ROC curve for LR(Model 1)

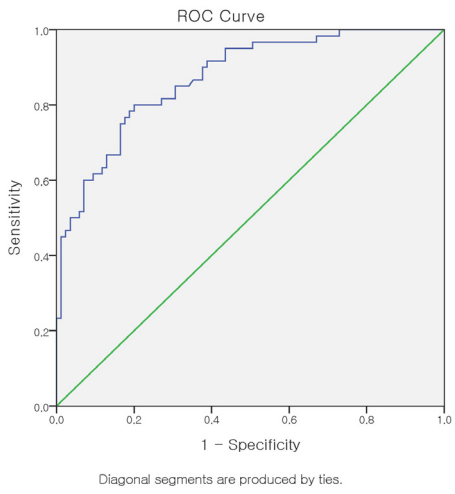

Fig. 2 ROC curve for LR(Model 2)

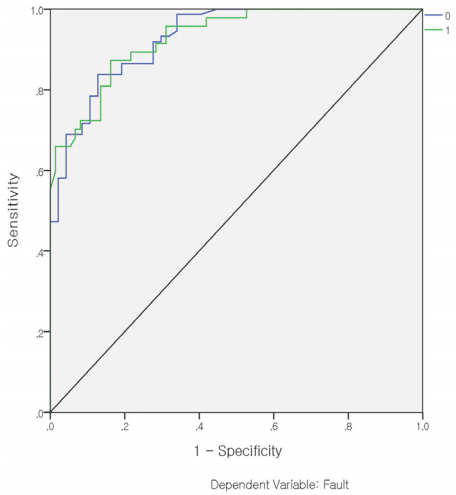

Fig. 3 ROC curve for ANN(Model 1)

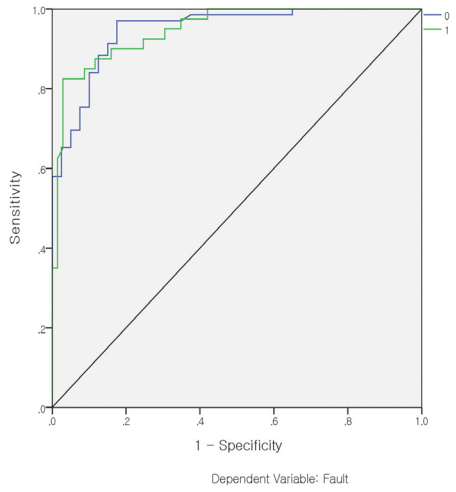

Fig. 4 ROC curve for ANN(Model 2)

4. 결 론

본 연구에서는 소프트웨어의 오류 발생에 따른 비용과 위험을 방지하기 위하여 소프트웨어 분석 및 설계 단계에서 고려해야할 품질 척도를 검증하였다. 소프트웨어 오류 및 작동 불량 등이 자주 발생하게 될 수 있는 객체지향 척도는 객체와 객체간의 연관성(CBO), 상속성의 깊이(DIT), 객체 내 메서드의 속성값 사용 비율(LCOM), 객체 내에서 구현된 메서드의 수(RFC)가 있으며 객체지향 척도는 아니지만 오류에

관련성이 큰 척도는 라인의 수(LOC)가 있었다. 객체지향 척도로만 구성된 모형과 라인수가 포함된 모형을 모두 사용하였다.

로지스틱 회귀분석과 인공신경망 분석을 이용하여 각각의 모형을 ROC 곡선을 이용하여 분석한 결과 라인수가 포함된 모형을 인공신경만 모델을 이용하여 분석한 것이 가장 우수한 결과를 보였다. 라인수 척도를 제외한 경우에도 인공신경만 모델이 더 우수하였다. 다만 사용된 데이터가 한국의 것이 아니며 산업표준으로 적용되고 있는 UML(Unified Modeling Language) 기반의 데이터 값이 모두 제공되지 못하였다.

향후 연구로는 객체지향 설계에서 가장 많이 사용하는 유스케이스 분석의 결과로 도출되는 유스케이스 다이어그램을 객체지향 척도 분석에 이용하는 것이다. 유스케이스 다이어 그램은 사용자 요구사항 도출의 주요 결과물이고 이를 이용하여 클래스를 설계하기 때문에 유스케이스 다이어그램 척도는 오류 및 신뢰도에 대한 척도로 매우 유용할 것이다.

References

[1] Boehm, B. W. (1981). “Software engineering economics”. Prentice-Hall.

[2] Chidamber, S. R. and Kemerer, C. F. (1994). “A metrics suite for object oriented design”. Software Engineering, IEEE Transactions on, Vol. 20, No. 6, pp. 476-493.

[3] Daly, J., Brooks, A., Miller, J., Roper, M. and Wood, M. (1995). “An empirical study evaluating depth of inheritance on the maintainability of object-oriented software”. Empirical Software Engineering: An International Journal. Vol. 1, No. 2, pp. 109-132.

[4] Briand, L. C., Morasca, S. and Basili, V. R. (1996). “Property-based software engineering measurement”. Software Engineering, IEEE Transactions on, Vol. 22, No. 1, pp. 68-86.

[5] Cartwright, M. (1998). “An empirical view of inheritance”. Information and Software Technology, Vol. 40, No. 14, pp. 795-799.

[6] Li, W. and Henry, S. (1993). “Object-oriented metrics that predict maintainability”. Journal of systems and software, Vol. 23, No. 2, pp. 111-122.

[7] Melo, W. (1996). “Evaluating the impact of object-oriented design on software quality”. In Software Metrics Symposium, 1996, Proceedings of the 3rd International, pp. 90-99.

[8] Briand, L., Devanbu, P. and Melo, W. (1997). “An investigation into coupling measures for C++”. In Proceedings of the 19th international conference on Software engineering, pp. 412-421.

[9] Lorenz, M. and Kidd, J. (1994). “Object-oriented software metrics: a practical guide”. Prentice-Hall.

[10] Bansiya, J. and Davis, C. G. (2002). “A hierarchical model for object-oriented design quality assessment”. Software Engineering, IEEE Transactions on, Vol. 28, No. 1, pp. 4-17.

[11] Malhotra, R., Kaur, A. and Singh, Y. (2010). “Empirical Validation of Object-Oriented Metrics to Predict Fault Proneness Using Open Source Software”. Software Quality Professional Magazine, Vol. 13, No. 1.

[12] Shirabad, J. S. and Menzies, T. J. (2005). “The PROMISE repository of software engineering databases”. School of Information Technology and Engineering, University of Ottawa, Canada, 24.