DOI: 10.3745/KIPSTD.2009.16-D.3.407

객체지향 메트릭 기반인 결함 예측 모형의 범용성에 관한 실험적 연구

김 태 연[†] · 김 윤 규[††] · 채 흥 석[†††]

요 약

검증과 확인을 통한 소프트웨어의 효율적인 관리를 지원하기 위하여 많은 연구들이 개발 초기 단계에 예측하기 위한 목적으로 연구를 하고 있다. 기존의 많은 연구들이 결함을 예측하기 위한 모형들을 제시했지만 기존의 연구에서는 결함 예측 모형을 다른 시스템에 범용적으로 적용이 가능한지에 대한 충분한 연구가 없었다. 또한 대부분의 결함 예측 모형은 모형 개발 당시와 같은 동일 시스템에서 예측력을 평가하였다. 그러므로 본 연구에서는 결함 예측 모형이 개발 당시와 다른 시스템에 범용적으로 적용될 수 있는지에 관하여 실험하였다. 실험은 3개의 실험 대상 시스템에 3개의 결함 예측 모형을 적용하여 예측력을 평가하였다. 실험 결과에서는 모형의 범용성에 대하여 찾을 수 없었다. 이는 모형의 개발 당시 시스템의 메트릭 분포가 실험 대상 시스템과 다르기 때문으로 분석된다. 따라서 결함 예측 모형을 타 시스템에도 적용할 수 있도록 결함 예측 능력의 범용성을 높이기 위한 추후 연구가 필요함을 확인하였다.

키워드 : 객체지향 메트릭, 결함 예측 모형, 범용성

An Experimental Study of Generality of Software Defects Prediction Models based on Object Oriented Metrics

Tae Yeon Kim[†] · Yun Kyu Kim[††] · Heung Seok Chae[†††]

ABSTRACT

To support an efficient management of software verification and validation activities, much research has been conducted to predict defects in early phase. And defect prediction models have been proposed to predict defects. But the generality of the models has not been experimentally studied for other software system. In other words, most of prediction models were applied only to the same system that had been used to build the prediction models themselves. Therefore, we performed an experiment to explore generality of major prediction models. In the experiment, we applied three defects prediction models to three different systems. As a result, we cannot find their generality of defect prediction capability. The cause is analyzed to result from a different metric distribution between the systems.

Keywords : Object-Oriented Metrics, Defects Prediction Model, Generality

1. 서 론

소프트웨어 개발 초기 단계에 한정된 자원을 효율적으로 관리하기 위하여 소프트웨어 메트릭을 이용한 결함 예측 연구가 진행되고 있다[1-8]. 결함 예측 연구는 디자인 단계의 산출물을 대상으로 메트릭을 측정하여 결함이 발생할 가능성이 높은 클래스를 판단하는데 사용할 수 있다. 그리고 소프트웨어 산출물을 대상으로 결함 유무를 판단하기 위해서는 결함 예측 인자로 소프트웨어 메트릭과 수학 통계적인 결함 예측 모형이 사용될 수 있다.

소프트웨어 메트릭을 이용하여 결함을 예측할 수 있는 이론적 배경은 소프트웨어의 구조적 속성들(Structural Properties)이 인지 복잡도(Cognitive Complexity)에 영향을 끼친다는 것이다[9]. 인지 복잡도는 개인이 개발, 테스트 활동 등에서 다루는 일에 대한 정신적 부담량으로 정의한다. 이는 소프트웨어의 구조적 복잡도가 높을수록 외부 품질요소인 유지보수성, 결함 등에 바람직하지 않은 영향을 미치게 되어 유지보수 비용을 높이거나 결함을 더 많이 발생하는 결과를 유발하게 된다. 또한, 연구자들은 결함 예측 모형을 구축하는데 주로 로지스틱 회귀분석을 사용하였다. 로지스틱 회귀분석을 이용하면 결함이 발생하기 쉬운 클래스와 그렇지 않

※ 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음.
† 준 회 원 : 부산대학교 컴퓨터공학과 석사
†† 준 회 원 : 부산대학교 컴퓨터공학과 석사과정
††† 정 회 원 : 부산대학교 컴퓨터공학과 조교수(교신저자)
논문접수 : 2009년 3월 13일
수 정 일 : 1차 2009년 4월 28일, 2차 2009년 5월 11일
심사완료 : 2009년 5월 11일

은 클래스를 분류할 수 있다.

결함 예측 연구 결과가 개발 현장에서 폭 넓게 활용되기 위해서는 결함 예측 모형이 범용성을 가져야 한다. 그러나 기존의 연구들은 대부분 소프트웨어에서 수집한 결함 및 메트릭 데이터를 바탕으로 결함 예측 모형을 구축한 후 이 모형을 동일한 소프트웨어에 적용하였다.

결함 예측 모형의 범용성을 확인하기 위하여 본 논문에서는 기존에 제안된 결함 예측 모형을 타 시스템에 적용하여 예측력을 비교하였다. 실험 대상 시스템은 대규모 시스템인 Eclipse 3.3, jEdit 4.0, NASA KC1으로 하였다. 또한, 결함 예측 모형은 최근 SCI 논문지에 발표된 모형 중 예측의 정확성이 60% 이상인 모형을 선정하여 총 8번 비교 실험하였다.

본 논문은 다음과 같이 구성 된다. 2절에서는 결함 예측 모형에서 사용되는 메트릭과 통계적 기법에 관하여 기술하고 결함 예측 모형을 평가하는 방법에 관하여 설명한다. 3절에서는 결함 예측 모형의 예측력을 평가하는 실험 절차 설명하고 다음으로 실험 대상 및 선정된 결함 예측 모형에 대하여 설명한다. 4절에서는 각 모형별로 예측력을 비교한 실험 결과를 기술하고 5절에서는 관련연구를 기술한다. 마지막으로 6절에서는 결론과 향후 연구 방향에 관하여 설명한다.

2. 연구배경

2절에서는 객체지향 메트릭, 결함 예측 모형 그리고 결함 예측 모형의 예측력 평가 방법에 관하여 기술한다. 객체지향 메트릭은 소프트웨어의 구조적 속성을 계산하여 클래스의 결함 여부를 평가하는 척도로 사용된다. 또한 결함 예측 모형은 통계적 기법과 메트릭 값을 이용하여 클래스의 결함 발생 여부를 판별한다.

2.1 객체지향 메트릭

소프트웨어의 품질에 대한 정량적인 측정 및 예측이 중요해지면서 다양한 산출물에 대하여 객체지향 메트릭을 사용하는 연구가 진행되고 있다. 객체지향 메트릭은 많은 문헌에서 Chidamber와 Kemerer가 제안한 CK 메트릭을 사용하였다[10-17]. 그리고 실험적 연구에서는 소프트웨어의 결함을 검증하거나 예측하기 위한 메트릭으로 CK 메트릭을 많이 사용하였다[1-8, 20]. 그러므로 본 연구에서는 CK 메트릭이 결함을 예측하기 위한 적절한 척도로 여기고 결함 예측 인자로 선정하였다. CK 메트릭은 시스템의 규모와 복잡도에 영향을 주는 객체지향 소프트웨어의 구조적 요소를 측정하는데 사용되며 다음과 같다[10].

- CBO(Coupling Between Object classes)
- DIT (Depth of Inheritance Tree of a class)
- LCOM(Lack of Cohesion on Methods)
- NOC(Number Of Children of a Class)
- RFC(Response For a Class)
- WMC(Weighted Methods Per Class)

총 6개의 CK 메트릭은 개별 클래스를 단위로 하여 상속, 커플링, 응집도 그리고 복잡도를 측정하기 위해 사용된다. 결함 예측 모형은 메트릭의 측정 결과값을 바탕으로 해당 클래스의 결함 발생 여부를 결정한다.

2.2 결함 예측 모형

기존 연구에서는 소프트웨어의 결함을 예측하는 통계적 기법으로 회귀분석이 사용하고 있다. 또한 Basili는 테스트 단계에서 개별 클래스에서 발견된 결함의 유무를 예측하는 연구를 하였고 소프트웨어 결함 예측을 위하여 로지스틱 회귀분석(Logistic Regression) 기법을 처음으로 적용하였다[18].

로지스틱 회귀분석은 최대 우도 추정(Maximum Likelihood Estimation)에 기반한 표준 통계 모델링 기법이다. 이 분석 기법은 소프트웨어 품질 분류 모형을 구축하는데 적합하다. 왜냐하면 회귀분석을 통하여 계산된 결과를 임계값을 기준으로 결함이 발생하기 쉬운 클래스와 결함이 발생하기 쉽지 않은 클래스로 분류할 수 있다. 그리고 종속 변수를 예측하기 위하여 사용하는 독립변수로는 객체지향 메트릭이 주로 사용된다. 로지스틱 회귀분석 수식의 일반적인 형태는 [수식 1]과 같다.

$$\pi(X_1, X_2, ..., , X_n) = \frac{e^{C_0+C_1X_1+...+C_nX_{n1}}}{1+e^{C_0+C_1X_1+...+C_nX_{n1}}}$$

[수식 1] 로지스틱 회귀모형

X_i 은 독립변수인 객체지향 메트릭의 측정값이며 본 연구에서는 CK 메트릭을 사용한다. 종속변수인 π 는 클래스에서 결함이 발견될 확률을 의미하며 메트릭 측정값을 사용하여 계산된 확률이다. 결함이 발견될 확률이 특정 임계값 이상이면 해당 클래스에는 결함이 있는 것으로 간주한다. π 값의 범위는 0부터 1까지 이다. C_i 은 X_i 의 회귀계수를 의미하며 C_i 값이 크면 클수록 해당 독립변수가 클래스에서 결함을 발견할 확률에 영향력이 높다.

로지스틱 회귀분석을 구축할 때 모든 관찰된 사건들은 통계적으로 독립적임을 가정한다. 즉 클래스에서 결함이 발견될 때 하나의 결함은 다른 결함이 발견되는데 아무런 영향을 미치지 않는다고 가정하는 것이다. 실제로 해당 클래스에서 결함이 하나 이상 발견되면 각각의 결함은 서로 독립적인 관찰을 통하여 발견되었음을 의미한다. 그러므로 실험에서는 위 가정을 받아드려 클래스에서 발견된 결함의 수를 이용하였다.

예를 들어 WMC 메트릭과 로지스틱 회귀모형 수식을 사용하여 결함 예측 모형을 구축했다고 가정하자. 위 수식을 통하여 π (WMC=10) = 0.7 라는 결과를 도출하였다면 이는 WMC 측정값이 10인 해당 클래스에서 실제로 결함이 존재할 확률이 70%임을 의미한다.

2.3 예측 모형의 예측력 평가 방법

결함 예측 모형의 예측력 평가 방법에는 정밀성과 정확성 그리고 완전성 등이 있다[2-4]. 정밀성은 전체 클래스 중에서 예측 모형에 의하여 정확히 분류된 클래스가 차지하는 비율이다. 정확성은 예측 모형에 의하여 결함으로 분류된 전체 클래스 중에서 실제로도 결함인 클래스가 차지하는 비율이다. 완전성은 실제로 존재하는 총 결함 중에서 예측 모형에서 결함으로 분류한 클래스에 존재하는 결함의 비율이다.

<표 1>은 개별 클래스를 예측 모형에 의해 결함 유무로 분류된 클래스와 실제 결함 유무를 함께 표시하였다. 분류표의 열은 결함 예측 모형에 의해 클래스의 결함을 예측하고 해당 결과에 따라 결함이 있는 클래스와 없는 클래스로 분류한다. 분류표의 행은 실제로 결함이 있는 클래스를 기준으로 분류한다. A1~A4까지 영역은 분류된 클래스들의 수이고 B1 과 B2는 해당 영역에 분류된 클래스들이 가지고 있는 실제 결함의 총합이다. 그러므로 정밀성과 정확성, 완전성은 다음과 같이 계산 할 수 있다.

위의 3가지 예측력 평가 척도는 값이 높을수록 결함 예측 모형의 예측력이 좋은 것으로 평가한다. 예를 들어 결함 예측 모형의 정밀성이 60%로 계산된 경우에는 100개의 클래스 중 60개의 클래스를 결함이 있거나 없는 클래스로 예측하였음을 나타낸다. 다음으로 정확성이 60%로 계산된 경우에는 100개의 결함이 있는 클래스에서 60개의 클래스를 결함이 있다고 예측 하였음을 나타낸다. 마지막으로 완전성이 60%로 계산된 경우에는 100개의 결함 중 60개의 결함을 예측 하였음을 나타낸다.

하지만 결함 예측 모형의 예측력 평가시 임계값(X)에 따라 모형의 예측력이 다르게 평가 될 수 있다. 그래서 기존의 연구에서는 정확성과 완전성 사이의 절충점을 임계값으로 하였다[2-4]. 이는 정확성과 완전성 사이에 절충관계(trade-off)가 있기 때문이다. 본 실험에서도 각 모형의 예측력을 평가하기 위하여 절충점을 모형의 임계값으로 결정하여 예측력을 평가하였다.

〈표 1〉 적합도 분류표

		결함 예측 모형에 의한 예측 결과	
		결함 무 ($\pi < X$)	결함 유 ($\pi \geq X$)
실제	결함 무	A1	A2
	결함 유	A3(B1)	A4(B2)

- 정밀성(Precision): (A1+A4) / (A1+A2+A3+A4)
- 정확성(Correctness): A4 / (A2+A4)
- 완전성(Completeness): B2 / (B1+B2)

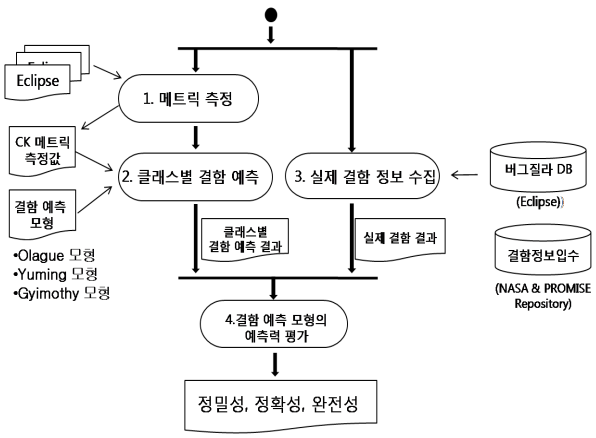
3. 실험환경

본 절에서는 모형의 비교 실험에 필요한 절차와 실험 환경에 대하여 기술한다. 3.1절에서는 결함 예측 모형의 예측력을 평가하기 위한 실험 절차를 설명한다. 3.2절에서는 실험 대상 시스템의 특성에 대해서 설명하고 3.3절에서는 기존의 결함 예측 모형과 특징을 설명한다.

3.1 실험 절차

(그림 1)의 타원형은 예측력 평가를 위한 세부 절차를 설명하고 있다. 그리고 가로형 막대는 개별 세부 절차가 독립적으로 수행됨을 의미한다. 평가 절차는 크게 두 부분으로 나뉜다. 첫 번째 단계에서는 예측력을 계산하기 위해서 기본적인 자료를 수집하거나 측정한다. 두 번째 단계에서는 모형이 예측한 클래스를 분류하고 예측력을 평가한다. 각 세부 절차별로 수행과정은 다음과 같다.

1. 실험 대상 시스템의 메트릭 측정: 해당 절차에서는 실험 대상인 Eclipse 3.3 소스 파일을 CK 메트릭으로 측정하는 것이 목적이다. 이후 단계에서 측정된 메트릭 결과값은 결함 예측 모형이 실험 대상의 결함을 예측하는데 사용된다. 본 실험에서 메트릭 측정은 Together 2007으로 하였고 기존의 다른 모형 연구에서는 자체 개발한 메트릭 측정도구를 사용하거나 상용 도구로 측정하였다. Olague의 연구[2]에서는 SSML tool chain[24] 도구를 사용하여 소스 코드 상태인 Mozilla Rhino 시스템을 CK, MOOD 그리고 QMOOD 메트릭으로 측정했다. 그리고 Gymoithy 연구[4]와 Marcus[1] 연구에서는 공개된 도구인 Columbus[25] 사용하여 메트릭을 측정하였다.
2. 결함 예측 모형에 의한 클래스별 결함 예측: 해당 절차에서는 결함 예측 모형에 의해서 실험대상의 클래스별 결함 유무를 예측한 결과값을 도출하는 것이 목적이다. 그러므로 결함 예측을 위해서 CK 메트릭 측정값과 결함 예측 모형이 필요하다. 실험에서 사용된 예측 모형은 Olague[2], Yuming[3] 그리고 Gyimothy[4] 모형이고 각 모형별로 클래스의 결함을 판별하기 위한 임계값이 있다. 실험 대상 클래스에서 결함이 있을 확률은 수식 1과 측정된 메트릭 값에 의해서 계산된다. 다음으로 결함 예측 모형의 임계값을 기준으로 클래스의 결함 유무가 판별된다. 즉 결함 예측 모형은 개별 클래스에서 임계값이 결함 확률과 같거나 큰 경우 결함이 있는 것으로 예측하고 임계값보다 결함 확률이 작으면 결함이 없는 것으로 예측한다.



(그림 1) 결함 예측 모형의 예측력 평가 절차

3. 실험 대상 시스템의 실제 결함 정보 수집: 해당 절차는 1,2번 절차와는 독립적으로 수행되며 실험대상의 실제 결함 유무에 대한 정보를 수집하는 것이 목적이다. 본 실험에서는 실제 결함 정보는 이클립스 웹사이트에서 제공하는 버그 추적 시스템인 버그질라에서 수집하였다. 그 외 결함정보는 NASA[26]와 PROMISE[27]에서 제공하는 파일로 입수하였다.
4. 결함 예측 모형의 예측력 평가: 해당 절차는 결함 예측 모형에 의해서 예측된 클래스와 실제 결함 있는 클래스의 정보를 바탕으로 결함 예측 모형의 예측력을 평가하는 것이 목적이다. 예측력의 평가 기준은 정밀성, 정확성 그리고 완전성이다.

3.2 실험 대상 시스템

실험 대상 시스템 선정은 실험 데이터의 정확성과 실제 결함 정보의 수집이 용이함을 기준으로 하였다. <표 2>는 실험 대상 선정 기준에 의해 수집된 대규모 공개 소프트웨어 시스템이다. 실험 대상 시스템은 4만 라인 이상의 대규모 시스템이고 결함이 있는 클래스의 비율은 15%~55%로 다양하게 수집되었다.

이클립스(Eclipse)는 자바 기반의 오픈 소스 프로젝트로서 실험에서는 이클립스 버전 3.3의 패키지 중 외부 라이브러리를 제외한 org.eclipse 패키지의 하위 패키지에 속한 소스 파일을 이용하였다. 그리고 해당 시스템의 결함 정보를 수집하기 위하여 이클립스가 사용하는 버그 추적 시스템(Bug Tracking System)인 버그질라(bugzilla)를 이용하였다. 버그 정보 수집을 위하여 버그질라 웹 페이지 정보를 수집하는 로봇을 개발하였다. 총 수집된 버그의 수는 4,133개이며 버그를 발생시킨 결함 클래스의 수는 1,725개였다.

NASA KC1은 오픈 소스 프로젝트와는 별도로 NASA의 소프트웨어 공학 연구소에서 NASA Metrics Data Program (MDP)을 통하여 제공되는 공개용 데이터 집합이다. 해당 시스템은 C++로 구현되었고 데이터에는 소스코드를 직접 제공하는 대신에 개별 클래스의 메트릭 측정값과 실제 결함 정보를 함께 제공하고 있다. 그리고 NASA KC1은 Yuming의 연구에서도 결함 예측 모형을 개발 시에 활용하였다[3].

jEdit는 오픈 소스 프로젝트로 JAVA 언어로 개발된 문서 편집기이다. 실험에서 사용된 jEdit의 버전은 4-0-final이며 메트릭 측정값 및 오류 정보는 PROMISE(PRedictOr Models In Software Engineering) 저장소를 통하여 공개되어 있다. 그리고 jEdit는 Watanabe의 결함 예측 모형의 구축에 사용된 시스템이다[19].

3.3 실험 대상 결함 예측 모형

많은 연구들이 소프트웨어 코드의 품질과 객체지향 메트릭 간의 관련성에 대해서 논의 하고 있다[1-8, 20]. 예측 모형의 정확한 비교 실험을 위하여 본 실험에서 사용한 예측 모형은 다음과 같은 기준으로 선정되었다.

- 결함 예측 모형의 예측력이 60% 이상인가?
- 4만 라인 이상의 대규모 시스템을 바탕으로 예측 모형이 개발 되었는가?
- 예측 모형이 실험을 통해 재검증 가능한 CK 메트릭으로 구성 되었는가?

<표 3>은 결함 예측 모형의 비교 실험을 위하여 위의 3가지 기준으로 선정한 모형을 요약하고 있다.

<표 3>에서 보면 각 행은 결함 예측 모형 선정 기준에 부합하는 모형이며 각 열은 예측 모형에 대한 주요 특징이다. 열의 예측 모형 수식은 로지스틱 회귀 분석식으로 표현되었다. 개별 클래스가 결함이 있을 확률(π)은 해당 모형 수식을 [수식 1]에 대입하여 구할 수 있다. 그리고 열의 정확성과 완전성은 예측 모형의 예측력을 나타내며 전체적으로 60%이상이다. 또한, 모든 모형에서 CBO와 WMC 메트릭을 사용하고 있기 때문에 해당 메트릭은 다른 메트릭보다 결함 있는 클래스를 찾는 것에 좀더 유의한 것으로 분석된다. 각 결함 예측 모형의 세부 특징은 다음과 같다.

Olague 연구에서는 자바언어로 개발된 오픈 소스 시스템

〈표 2〉 실험 대상 시스템의 규모 및 결함 정보

실험 대상 시스템	총 클래스	결함 클래스	결함 수	결함 클래스 비율	소스 라인(KLOC)
Eclipse 3.3	11,505	1,725	4,133	15%	1,455
NASA KC1	145	58	633	40%	40
jEdit 4.0	369	204	N/A	55%	160

〈표 3〉 실험 대상 결함 예측 모형

예측 모형(년도)	예측 모형 수식	정확성	완전성	구축 시스템	시스템 규모	개발 언어
Olague [2] (2007)	-2.98 + 0.06*CBO + 0.61*DIT -0.12*LCOM + 0.47*NOC + 0.02*WMC	82%	N/A	Mozilla Rhino 14R3	58.7 K 라인	Java
Yuming [3] (2006)	-0.431 + 0.224*CBO + 0.004*LOC -0.011*LCOM -1.063*NOC -0.019*RFC +0.031*WMC	61.4%	74.6%	NASA KC1	40 K 라인	C++
Gyimothy [4] (2005)	-0.229 + 0.631*CBO + 0.31*DIT + 0.461*LOC +0.246*WMC	72.6%	65.2%	Mozilla v.1.6	1 M 라인	C++

인 Mozilla Rhino 14R3를 대상으로 예측 모형을 만들었다. 개발된 예측 모형의 실제 예측력 비교 실험은 6개의 다른 버전인 Mozilla Rhino 시스템과 CK, MOOD, QMOOD 메트릭 모음으로 예측률에 관한 비교 연구를 수행 하였다. 그 결과 CK와 QMOOD 메트릭 모음을 이용한 결함 예측 모형이 우수하게 나왔다. 본 실험에서는 제시된 결함 예측 모형 중에서 예측력이 가장 높은 Mozilla Rhino 14R3를 선정했고 해당 시스템의 실제 결함 비율은 알 수 없었다.

Yuming 연구에서는 NASA KC1 데이터 집합을 실험 대상으로 하였다. 그리고 결함의 심각도를 두 단계로 분류하여 심각도별 예측 모형을 만들었다. 로지스틱 회귀분석 기법으로 만들어진 예측 모형은 높은 심각도 보다 낮은 심각도에 예측률이 더 높았다. 그리고 예측 모형은 클래스의 결함 비율이 42.6%인 시스템에서 개발 되었다.

Gyimothy 연구는 대형 오픈 소스 소프트웨어인 Mozilla 1.6을 대상으로 예측 모형을 개발하였다. 그리고 예측 모형을 개발하기 위한 기법으로 로지스틱 회귀분석과 함께 머신 러닝 기법을 적용하여 예측력을 높이고자 하는 시도를 하였다. 그 결과 로지스틱 회귀분석으로 개발된 예측 모형은 정확성이 머신 러닝으로 개발된 모형에 비하여 높았으나 완전성은 낮았다. 그리고 예측 모형은 클래스의 결함 비율이 39.3%인 시스템에서 개발 되었다.

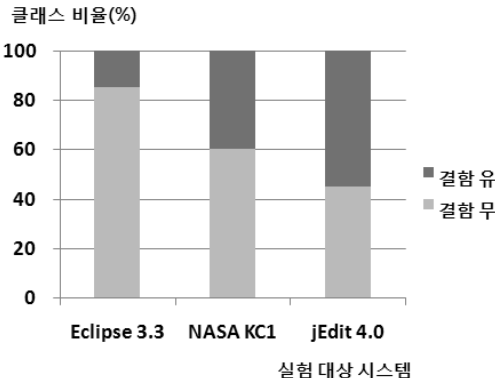
4. 실험결과

본 절에서는 결함 예측 모형을 서로 다른 시스템에 적용한 후 예측력의 결과를 기술한다. 우선 4.1에서는 실험 대상 결함 분포 및 메트릭 측정 결과값의 기술 통계량(Descriptive Statistics)을 설명한다. 다음으로 4.2에서는 결함 예측 모형의 비교 결과를 요약 및 상세화하여 분석한다.

4.1 실험 결과 데이터 기술 통계량

(그림 2)는 실험 대상 시스템의 결함 분포를 나타낸다. Eclipse는 전체 11,505개의 클래스 중에서 85%의 클래스가 결함을 포함하고 있지 않다. NASA의 KC1 데이터와 jEdit는 결함을 포함하지 않은 클래스의 수가 60%이하이다. 반면에 Eclipse는 다른 시스템에 비하여 결함의 농도가 낮다. 이는 버그 추적 시스템을 통하여 수집된 버그가 개발단계에서 수집된 버그가 아닌 배포 후에 수집된 버그이기 때문으로 판단된다[20]. 또한 소프트웨어 개발단계에서는 테스팅, 코드 검사 등의 품질관리 활동을 통하여 결함을 최대한 검출하였기 때문에 제품이 사용자에게 배포되고 난 이후에 보고되는 버그의 수는 개발단계에 비하여 적었던 것으로 판단된다.

<표 4>는 실험대상 시스템을 CK 메트릭으로 측정한 기



(그림 2) 실험 대상 시스템의 결함 분포

〈표 4〉 실험 대상 시스템의 기술 통계량

	메트릭	평균	표준편차	최소값	Q1	중간값	Q3	최대값
Eclipse 3.3	WMC	9.6	20.7	0	2	5	11	1023
	LCOM	31.8	38.2	0	0	4	71	100
	RFC	48.6	64	0	16	25	52	1213
	DIT	2.2	1.7	0	1	2	3	11
	CBO	14.8	19.3	0	3	8	19	231
	NOC	1.1	5.5	0	0	0	0	210
	LOC	126.4	288	1	16	45	130	7740
NASA KC1	WMC	17.4	17.5	0.0	8.0	12.0	22.0	100.0
	LCOM	68.7	36.9	0.0	56.5	84.0	96.0	100.0
	RFC	34.4	36.2	0.0	10.0	28.0	44.5	222.0
	DIT	2.0	1.3	1.0	1.0	2.0	2.5	7.0
	CBO	8.3	6.4	0.0	3.0	8.0	14.0	24.0
	NOC	0.2	0.7	0.0	0.0	0.0	0.0	5.0
	LOC	211.2	345.6	0.0	8.0	108.0	235.5	2313.0
jEdit 4.0	WMC	11.7	26.5	0.0	3.0	5.0	11.0	351.0
	LCOM	48.9	33.7	0.0	0.0	59.0	77.0	100.0
	RFC	170.0	268.6	0.0	20.0	32.0	71.0	862.0
	DIT	2.4	2.0	0.0	1.0	2.0	3.0	8.0
	CBO	13.5	14.3	0.0	4.0	9.0	18.5	125.0
	NOC	0.7	3.0	0.0	0.0	0.0	0.0	38.0
	LOC	225.3	478.1	3.0	38.5	94.0	216.5	5317.0

술 통계량 정보이다. 먼저 NOC와 DIT를 살펴보면 다른 메트릭 값에 비하여 상대적으로 낮은 값을 가진다. 이는 여러 연구에서와 같은 결과를 나타내며 위 메트릭의 변동성이 크지 않은 점은 Kemerer의 논문에서도 인정하고 있다[12]. 그러나 Eclipse의 LCOM의 경우에 전체 값 중 50%는 0이고 나머지 50%는 1에서 500,701까지 매우 큰 변동성을 가진다. 여러 연구에서 LCOM의 정의에 문제가 있음을 지적하였다[15, 21, 22]. 그러므로 결함 예측 모형은 개발 당시와 다른 메트릭 분포를 가진 시스템에 대하여 적용시에 원 예측력과는 다른 결과가 나올 수 있음을 유추 할 수 있다.

4.2 예측 모형의 평가

<표 5>는 선정된 결함 예측 모형을 3개의 개별 시스템에 적용한 결과이다. 첫번째 열은 실험에서 사용된 결함 예측 모형을 나타낸다. 두번째 열에서는 실험 대상 시스템을 기술하였으며 나머지 열에서는 정밀성, 정확성, 완전성의 순서로 결함 예측 모형의 예측력을 기술하였다. 그리고 실험대상 시스템과 원 논문에서 주장된 결함 예측력을 비교하기 위하여 함께 기술하였다.

<표 5>의 실험 결과는 모형의 예측력이 주장된 예측력과 유사한 경우(C0), 주장된 예측력보다 현저히 낮은 경우(C1), 그리고 예측력의 해석이 불가능한 경우(C2)로 분류하였다. 이는 <표 5>의 비고란에 각각 C0, C1, C2로 표시하였다.

- 주장된 예측력과 유사한 경우(C0): 총 8번의 실험 중에서 하나의 실험(YJ-5)만이 결함 예측 모형의 주장된 예측력과 실험 결과가 유사하였다. Yuming 모형인 경우 주장된 예측력이 약 61%에서 74%이었고 실험 결과 65%의 예측력을 보였다.
- 주장된 예측력보다 현저히 낮은 경우(C1): 총 8번의 실험 중 하나의 실험(OE-1)에서 예측력의 해석은 가능했지만 주장된 예측력보다 현저히 낮은 정확성을 보였다. Olague모형의 주장된 정확성은 82%인 것에 비하여 실험 결과에서 정확성은 17.4%로 현저히 낮았다(Olague 모형 연구에서는 정밀성과 완전성은 논문에서 제시하고 있지 않았다).
- 예측력의 해석이 불가능한 경우(C2): 총 8번의 실험 중에서 6개의 실험이 예측력의 해석이 불가능한 C2 유형이었다. C2유형의 모든 실험에서 임계값은 0이거나 1에 매우 근사하였다.
  - ■ 임계값이 0에 근사한 경우
    2개의 실험(ON-2, OJ-3)에서 임계치는 0에 매우 근사하였다. 따라서 2개의 실험에서 정확성과 완전성 간의 절충점을 찾지 못하였다. 해당 실험 결과의 상세한 분석은 4.2.3절에 하였다.
  - ■ 임계값이 1에 근사한 경우
    4개의 실험(YE4-4, GE-6, GN-7, GJ-8)에서 임계치는1에 매우 근사하였다. 실험 YE4-4에서는 정확성과 완전성의 절충점을 찾지 못했으며 특히 실험 GE-6, GN-7그리고 GJ-8에서는 정확성과 완전성간의 예측율의 차이가 더욱 컸다. 해당 실험 결과의 상세한 분석은 4.2.3절에 하였다.

실험 결과를 요약하면 3개의 결함 예측 모형 대상으로 3개의 서로 다른 시스템에 결함 예측을 위한 실험을 하였다. 총 8번의 실험 중에서 6개의 실험(C2유형)은 예측력의 해석이 불가능했고 하나의 실험(C1 유형)에서는 주장된 정확성보다 현저히 낮은 정확성을 보였다. 비록 나머지 하나의 실험(C0 유형)에서 주장된 예측력과 유사한 결과를 보였지만 전반적인 실험 결과를 요약하면 결함 예측 모형을 개발 당시와 다른 시스템에 적용하는 것은 범용적이지 않은 것으로 분석된다.

〈표 5〉 결함 예측 모형별 실험 대상의 예측력 결과

결함예측모형	실험대상시스템	정밀성	정확성	완전성	임계값	실험번호	비고
Olague	주장된 예측력	자료없음	82%	자료없음	자료없음	-	-
	Eclipse 3.3	71.9%	17.4%	17.4%	0.528	OE-1	C1
	NASA KC1	54.5%	38.2%	12.9%	0.001	ON-2	C2
	jEdit 4.0	39.0%	44.5%	41.7%	0.001	OJ-3	C2
Yuming	주장된 예측력	69.7%	61.4%	74.6%	0.859	-	-
	Eclipse 3.3	81.0%	34.6%	33.9%	0.999	YE-4	C2
	NASA KC1†	모형의 개발 당시 시스템이기 때문에 실험하지 않음					
	jEdit 4.0	61.5%	65.2%	65.2%	0.424	YJ-5	C0
Gyimothy	주장된 예측력	69.6%	72.6%	65.2%	0.5	-	-
	Eclipse 3.3	85.3	69.8	9.2	1.0	GE-6	C2
	NASA KC1	48.3%	43.6%	100%	0.999	GN-7	C2
	jEdit 4.0	56.4%	56.3%	93.6%	0.999	GJ-8	C2

† : Yuming의 모형은 NASA KC1 시스템을 대상으로 구축된 모형이므로 예측력으로 표시하지 않음

4.2.1 주장된 예측력과 유사한 경우(C0)

<표 6>은 Yuming 모형으로 jEdit 4.0 시스템에 결함을 예측한 실험(YJ-5)의 상세한 결과이다. 해당 모형에서는 임계값은 0.424로 선정되었고 정확성과 완전성은 65.2%이었다. 이는 기존 논문에서 주장된 예측력과 유사한 값이다. Yuming 모형에서는 165개의 클래스가 결함이 없는 것으로 예측이 되었고 204개의 클래스는 결함이 있는 것으로 예측되었다. 그리고 그 중에서 65.2%(정확성), 즉 133개의 클래스가 실제로 결함을 가진 클래스였다. 이는 Yuming 모형이 jEdit 시스템을 대상으로 유의한 수준에서 결함을 예측하고 있는 것으로 판단할 수 있다.

〈표 6〉 Yuming 모형으로 jEdit 4.0 시스템에 결함을 예측한 분류표

실험 번호 : YJ-5		결함 예측 모형	
		결함 무 ($\pi < 0.424$)	결함 유 ($\pi \geq 0.424$)
실제	결함 무	94	71
	결함 유	71(71)	133(133)

4.2.2 주장된 예측력보다 현저히 낮은 경우(C1)

<표 7>은 Olague 모형으로 Eclipse 3.3 시스템에 결함을 예측한 실험(OE-1)의 상세한 결과이다. 해당 모형에서는 임계값을 0.528로 선정되었고 정확성은 17.4% 이었다. 이는 기존 논문에서 주장한 82%의 정확성보다 매우 낮은 값이다. 그리고 대상시스템의 결함 밀도가 15%인 것을 고려하면 무작위로 결함 있는 클래스를 찾는 정확성과 해당 모형의 정확성 간에 거의 차이가 없기 때문에 해당 모형은 예측력이 부족한 것으로 판단된다.

〈표 7〉 Olague 모형으로 Eclipse 3.3 시스템에 결함을 예측한 분류표

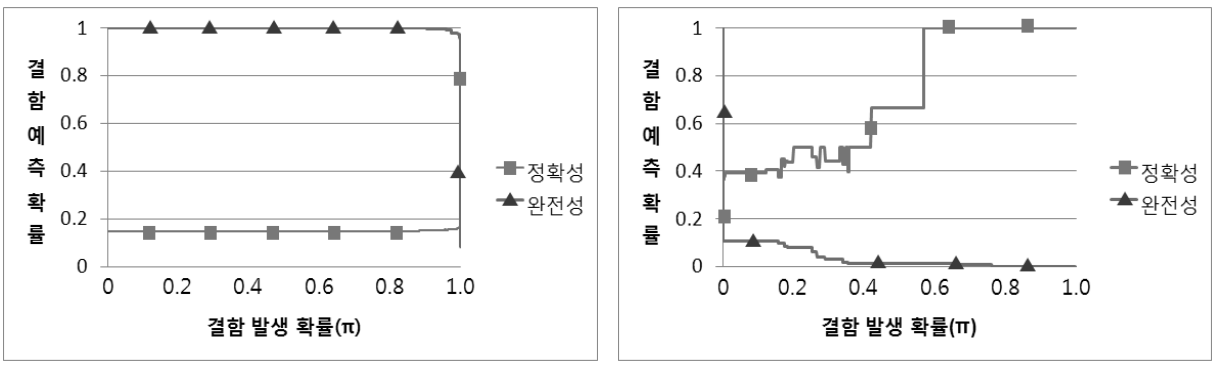
실험 번호 : OE-1		결함 예측 모형	
		결함 무 ($\pi < 0.528$)	결함 유 ($\pi \geq 0.528$)
실제	결함 무	8,384	1,396
	결함 유	1,431(3,414)	294(719)

4.2.3 예측력의 해석이 불가능한 경우(C2)

결함 예측 모형의 예측력을 해석하지 못한 경우는 모두 임계값이 극단적인 값이기 때문에 예측력 평가가 불가능 했다. 그림 3은 결함 예측 모형이 예측하지 못한 시스템의 임계값을 찾는 그래프이다. (그림 3-a)에서는 임계값이 0.999 이상인 실험번호 YE-4, GE-6, GN-7그리고 GJ-8인 경우의 그래프이다. 결함 예측 모형은 개별 클래스의 결함 발생 확률이 0.999 이상이면 결함이 있는 것으로 분류하고 낮으면 결함이 없는 것으로 분류한다. 하지만 임계값이 1에 매우 근사 되어 있기 때문에 사실상 거의 모든 클래스를 오류가 없는 것으로 분류하게 된다. 그러므로 정확성은 낮게 계산이 되고 완전성은 100%에 가깝게 된다. 결함 예측 모형의 절충점을 찾지 못한 이유는 로지스틱 회귀분석을 통한 회귀함수의 함수 값이 극단 값에 집중되어 있기 때문이다. 이는 실험 대상 시스템의 메트릭 값이 모형 개발 당시의 메트릭 값 분포보다 크기 때문에 로지스틱 회귀 모형 수식의 값이 1에 가깝게 계산된 것으로 분석된다.

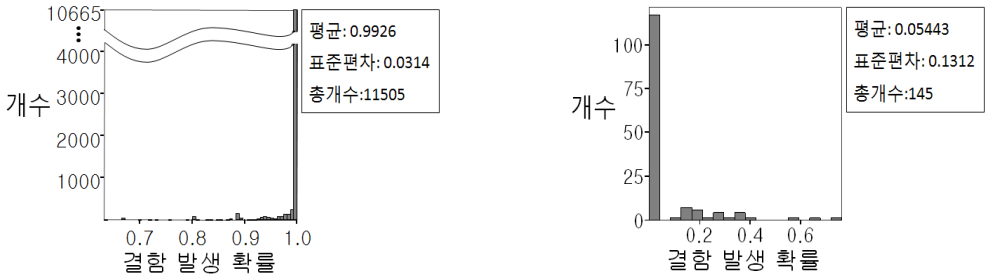
(그림 3-b)에서는 (그림 3-a)와 대조적으로 임계값이 0에 가깝게 근사 되어 결함 예측 모형의 적용이 불가능한 그래프이다. 대부분의 클래스들의 결함 발생확률이 0.001 이하의 값을 가지므로 절충점이 0에서 0.001 사이의 값에서 결정되게 된다. 이는 결함 예측 모형을 타 시스템에 적용시에 절충점 결정이 쉽지 않음을 의미한다. 그 이유는 절충점 값의 매우 작은 변화에도 결함 예측 모형의 예측력이 크게 달라지게 되기 때문이다.

(그림 4)는 결함 예측 모형이 결함 발생확률을 예측한 클래스의 확률 분포도이다. (그림 4)의 (a)와(b) 그래프는 (그림 3)의 (a)와 (b) 그래프와 대응된다. 예를 들어 실험번호 GE-6의 결함 예측 결과는 예측이 불가능하였다. 그 원인은 대부분의 클래스의 결함 발생 확률이 1에 가깝게 계산된 것으로 분석되었다. 그 결과 (그림 3) (a)와 같이 정확성과 완전성간의 절충점을 찾을 수 없으며 그림 4 (a)에서 클래스별 결함 발생 확률 분포가 실제로 1에 대부분 편향 되어 있음을 확인 할 수 있다. (그림 3) (a)와 (b)의 나머지 실험들도 (그림 4)의 (a), (b) 그래프와 매우 유사하여 생략하였다.



(a) 임계값이 1인 경우(YE-4, GE-6, GN-7, GJ-8)　　(b) 임계값이 0인 경우(ON-2, OJ-3)

(그림 3) 정확성과 완전성 간의 절충점 그래프



(a) 임계값이 1인 경우(GE-6) (b) 임계값이 0인 경우(ON-2)

(그림 4) 결함 발생 확률 분포도

5. 관련 연구

본 절에서는 결함 예측 모형을 타 시스템에 적용한 관련 연구를 소개한다. 해당 연구들은 모형의 적용 사례에 따라서 세 부분으로 나누어 소개한다. 우선 버전이 다른 제품군에 결함 예측 모형을 적용한 연구를 소개하고 다음으로 동일한 개발 조직에서 서로 다른 시스템을 대상으로 결함 예측 모형을 적용한 연구를 소개한다. 마지막으로 서로 다른 개발 언어간의 결함 예측 모형을 적용한 연구를 소개한다.

Shatnawi[20]의 연구에서는 동일 프로젝트의 서로 다른 버전에 대하여 결함 예측 모형의 예측력을 적용하였다. Eclipse 2.0에서 개발된 결함 예측 모형은 Eclipse 2.1과 3.0 버전에 결함을 예측하였고 그 결과 동일 제품군에는 결함 예측 모형을 적용하는 것이 유의함을 보였다. 또한 Gyimothy[4]의 연구는 Mozilla 1.6을 대상으로 로지스틱 회귀분석 방법으로 개발된 결함 예측 모형을 Mozilla 1.0 이후 버전에 적용하였으며 Shatnawi의 연구와 동일하게 결함 예측이 가능하다고 주장하였다. 그러나 앞선 두 연구는 결함 예측 모형의 예측력을 동일한 프로젝트의 버전만 다른 실험 대상으로 평가하였고 해당 모형을 타 시스템에 적용이 가능한지에 대한 고려가 없었다. 본 실험에서는 개발된 모형을 타 시스템에 적용한 경우 예측력이 현저히 떨어짐을 보였다.

다음으로 Briand[23]의 연구에서는 동일한 팀에서 개발한 두 개의 서로 다른 시스템을 대상으로 결함 예측력을 평가하였다. 실험 결과는 타 시스템에 예측 모형을 사용하는 것이 어렵다고 분석되었다. 그 이유는 개발 팀의 숙련도와 개발 시스템의 분석 방법 및 전략에 따라서 소프트웨어의 특성이 다르게 나타나며 이는 결과적으로 결함 예측에 영향을 준다고 주장 하고 있다.

마지막으로 Watanabe[19]의 연구에서는 결함 예측 모형이 서로 다른 언어로 개발된 시스템에 적용이 가능한지 비교 실험하였다. 결함 예측 모형의 예측력을 정확히 비교하기 위해서 실험 대상의 도메인과 시스템의 규모는 비슷하게 선정한 후 개발 언어만이 다르도록 실험을 준비 하였다. 그리고 예측할 실험 대상 시스템의 메트릭 값을 보정 후 결함 예측 모형을 적용하였다. 실험 결과를 요약하면 시스템간의 도메인과 크기가 비슷하면 결함 예측 모형의 재사용이 가능하다는 것이다. 하지만 실험은 오직 두 개의 C++, JAVA 프로젝트로 수행했기 때문에 일반화하기에는 부족함이 있다. 또한 연구에서 주장하는 예측력을 유지하기 위해서는 메트릭 값의 보정이 필요하지만 이는 모형의 구축 시스템에 메트릭 분포 및 측정값을 고려해야 하기 때문에 현실적으로 적용이 어렵다.

6. 결론 및 향후 연구

객체지향 설계 기법을 이용하여 개발된 소프트웨어의 품질을 높이기 위한 방법 중에는 소프트웨어 메트릭을 사용한 결함 예측 모형이 있다. 본 논문의 목적은 기존의 결함 예측 모형들을 범용적인 예측 모형으로 사용이 가능한지 확인하고 실제로 대규모 프로젝트를 대상으로 결함 예측의 정확성을 비교하기 위함에 있다.

비교 실험은 객체지향 메트릭과 로지스틱 회귀분석을 이용하여 서로 다른 3개의 시스템으로부터 구축된 결함 예측 모형을 Eclipse 3.3, NASA KC1, jEdit 4.0에 적용하여 예측력을 검증하였다. 8번의 실험 중에서 6개의 실험은 모형의 예측력을 해석하는 것이 불가능했다. 해석이 불가능한 이유는 결함 발생 확률의 분포가 특정 지점에 집중적으로 위치하여 결함 예측 모형의 임계값을 결정할 수가 없었기 때문이다. 그리고 나머지 실험 중에서 Olague 모형을 Eclipse 3.3에 적용한 경우 모형의 임계값은 결정할 수 있었지만 결함 예측 모형의 정확성은 현저히 낮았다. 그 이유는 Olague 모형이 개발된 당시 시스템의 메트릭 분포와 예측 대상 시스템의 메트릭 분포가 다르기 때문으로 판단된다. 비록 Yuming 모형을 jEdit 4.0에 적용한 실험인 경우 주장된 예측력과 유사한 결과를 보였지만 전반적인 실험 결과를 요약하면 결함 예측 모형을 개발 당시와 다른 시스템에 적용하는 것은 범용적이지 않은 것으로 분석된다.

본 논문에서는 결함 예측 모형이 개발된 당시 시스템과 다른 시스템에 적용했을 경우에 결함 예측 모형의 예측력이 일반성을 유지하는지 비교 실험하였다. 그리고 모형의 예측력이 일반성을 유지하지 못하는 이유에 대하여 실험 대상 시스템의 메트릭 분포가 원인인 것으로 분석하였다.

향후 연구에서는 결함 예측 모형의 범용성을 확보하기 위하여 다음과 같이 두 가지 방법을 사용하여 연구 할 것이다. 첫째 예측 인자로 사용된 메트릭의 분포를 분석하여 모형이

개발된 당시 시스템과 유사한 메트릭 분포를 가진 시스템에 예측하는 것이 타당한지 분석할 것이다. 둘째 모형과 예측 대상 시스템의 결함 밀도 혹은 개별 클래스가 가진 결함의 분포도를 조사하여 유사한 밀도나 분포도를 가지는 경우 제한적으로 예측하는 것이 타당한지 분석할 것이다.

참 고 문 헌

[1] A. Marcus, D. Poshyvanyk and R. Ferenc, "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems," IEEE Transactions on Software Engineering, Vol.34, No.2, pp.287-300, 2008.

[2] H.M. Olague, L.H. Etzkorn, S. Gholston and S. Quattlebaum, "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," IEEE Transactions on Software Engineering, Vol.33, No.6, pp.402, 2007.

[3] Y. Zhou and H. Leung, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," IEEE Transactions on Software Engineering, Vol.32, No.10, pp.771-789, 2006.

[4] T. Gyimóthy, R. Ferenc and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions on Software Engineering, Vol.31, No.10, pp.897-910, 2005.

[5] R. Subramanyam, M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," IEEE Transactions on Software Engineering, Vol.29, No.4, pp.297-310, 2003.

[6] G. Succi, W. Pedrycz, M. Stefanovic, and J. Miller "Pratical Assessment of the Models ofr Identification of Defect-Prone Classes in Object-Oriented Commercial Systems Using Design Metrics," Journal of Systems and Software , Vol.65, No.1, pp. 1-12, 2003.

[7] K. El Emam, S. Benlarbi, N. Goel and S.N. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics," IEEE Transactions on Software Engineering, Vol.27, No.7, pp.630-650, 2001.

[8] K. El Emam, W. Melo and J.C. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," Journal of Systems and Software, Vol.56, No.1, pp.63-75, 2001.

[9] S. Cant, D. Jeffery and B. Henderson-Sellers, "A Conceptual Model of Cognitive Complexity of Elements of the Programming Process," Information and Software Technology, Vol.37, No.7, pp.351-362, 1995.

[10] S.R. Chidamber, C.F. Kemerer and C. Mit, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, Vol.20, No.6, pp.476-493, 1994.

[11] F.B. Abreu, M. Goulao and R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems," Proceedings, 5th International Conference. Software Quality, pp.44-57, 1995.

[12] A.L. Baroni and F.B. Abreu, "Formalizing Object-Oriented Design Metrics upon the UML Meta-Model," Proceedings, Brazilian Symposium on Software Engineering, Gramado-RS, Brazil, 2002.

[13] K. El-Emam, "Object-Oriented Metrics: A Review of Theory and Practice," Advances in Software Engineering: Topics in Comprehension, pp.23-50, 2002.

[14] B. Henderson-Sellers, "Object-Oriented Metrics: Measures of Complexity," Upper Saddle River, N.J.: Prentice Hall, 1996.

[15] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," Journal of Systems and Software, Vol.23, No.2, pp.111-122, 1993.

[16] J.K. Mark Lorenz, "Object-Oriented Software Metrics: a practical guide," Prentice Hall, 1994.

[17] L.C. Briand, J.W. Daly and J. Wüst, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," Empirical Software Engineering, Vol.3, No.1, pp.65-117, 1998.

[18] V.R. Basili, L.C. Briand, W.L. Melo, "A Validation of Object-Oriented Design Metrics as Qualityindicators," IEEE Transactions on Software Engineering, Vol.22, No.10, pp.751-761, 1996.

[19] S. Watanabe, H. Kaiya and K. Kaijiri, "Adapting a Fault Prediction Model to Allow Inter Languagereuse," in Proceedings of the 4th international workshop on Predictor models in software engineering, pp.19-24, 2008.

[20] R. Shatnawi and W. Li, "The Effectiveness of Software Metrics in Identifying Error-Prone Classes in Post-Release Software Evolution Process," Journal of Systems and Software, Vol.81, No.11, pp.1868-1882, 2008.

[21] L.C. Briand, S. Morasca and V.R. Basili, "Property-Based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol.22, No.1, pp.68-86, 1996.

[22] M. Hitz and B. Montazeri, "Chidamber and Kemerer's Metrics Suite: a Measurement Theory Perspective," IEEE Transactions on Software Engineering, Vol.22, No.4, pp.267-271, 1996.

[23] L.C. Briand, W.L. Melo and J. Wüst, "Assessing the applicability of fault-proneness models across Object-Oriented software projects," IEEE Transactions on Software Engineering, Vol.28, No.7, pp.706-720, July, 2002.

[24] S. Quattlebaum, "A Comparison of the Results of Object-Oriented Metrics in C++ and Java," MS Thesis, Univ. of Alabama in Huntsville, 2004.

[25] Columbus, FrontEndART.ltd, http://www.frontendart.com/columbus.php

[26] NASA IV&V FACILITY, Metrics Data Program, http://mdp.ivv.nasa.gov/repository.html

[27] PROMISE., PROMISE data sets, http://promisedata.org

김 태 연

e-mail : tykim@pusan.ac.kr
1998년~2001년 ㈜세안IT 병원사업부 사원
2003년~2005년 ㈜에이치원 병원사업부 대리
2007년 부산대학교 정보컴퓨터공학(학사)
2009년 부산대학교 컴퓨터공학과(석사)
2009년 2월~현 재 (주) LG 전자 MC 연구소 개발연구원
관심분야 : 데이터 모델 설계, 대용량 데이터 솔루션, 객체지향 설계, 소프트웨어 테스팅, 소프트웨어 메트릭

김 윤 규

e-mail : kim_yunkyu@pusan.ac.kr
2008년 부산대학교 정보컴퓨터공학(학사)
2008년~현 재 부산대학교 컴퓨터공학과 석사과정
관심분야 : 소프트웨어 아키텍처, 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭

채 흥 석

e-mail : hschae@pusan.ac.kr
1994년 서울대학교 원자핵공학(학사)
1996년 한국과학기술원 전산학(석사)
2000년 한국과학기술원 전산학(박사)
2000년~2003년 ㈜ 동양시스템즈 기술연구소 선임연구원
2003년~2004년 한국과학기술원 전산학과 초빙교수
2004년~현 재 부산대학교 컴퓨터공학과 조교수
관심분야 : 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭, 소프트웨어 유지보수, 미들웨어 설계, 프로덕트 라인 공학