미비사항:

12J5-022R

Vol.2, No.10 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2013.2.10.001

Classifying a Strength of Dependency between classes by using Software Metrics and Machine Learning in Object-Oriented System

Sungkyun Jung[†] · Jaegyoon Ahn[††] · Yunku Yeu[†††] · Sanghyun Park[††††]

ABSTRACT

Object oriented design brought up improvement of productivity and software quality by adopting some concepts such as inheritance and encapsulation. However, both the number of software's classes and object couplings are increasing as the software volume is becoming larger. The object coupling between classes is closely related with software complexity, and high complexity causes decreasing software quality. In order to solve the object coupling issue, IT-field researchers adopt a component based development and software quality metrics. The component based development requires explicit representation of dependencies between classes and the software quality metrics evaluates quality of software. As part of the research, we intend to gain a basic data that will be used on decomposing software. We focused on properties of the linkage between classes rather than previous studies evaluated and accumulated the qualities of individual classes. Our method exploits machine learning technique to analyze the properties of linkage and predict the strength of dependency between classes, as a new perspective on analyzing software property.

Keywords : Object Oriented Design, Object Oriented Metrics, Strength of Coupling, Machine Learning, Software Quality, Software Clustering

기계학습과 품질 메트릭을 활용한 객체간 링크결합강도 분류에 관한 연구

정 성 균[†] · 안 재 균[††] · 여 윤 구[†††] · 박 상 현[††††]

요 약

객체지향 설계는 상속 및 은닉과 같은 개념이 도입되어 소프트웨어 개발 생산성 및 품질 향상을 가져다 주었다. 하지만 소프트웨어의 크기가 커지게 되면 이를 구성하는 객체의 수가 증가하고 이에 비례하여 상속 또는 호출과 같은 객체간 결합관계가 증가한다. 또한 이러한 객체간 결합관계는 객체지향 소프트웨어의 복잡도와 밀접한 관계를 갖고 있는데 다수의 결합관계는 소프트웨어의 복잡도를 높이어 결국에는 소프트웨어 품질저하로 이어지게 된다. 그래서 소프트웨어 개발 분야에서는 컴포넌트 기반의 설계와 같은 방법을 통하여 객체간 결합관계를 명확히 함으로써 소프트웨어의 품질을 높이려는 노력이 진행되고 있다. 또한 객체 품질 메트릭을 정의, 산출하여 소프트웨어의 품질을 측정하고 이를 활용하여 높은 품질의 소프트웨어가 될 수 있는 방법들을 찾는 연구가 함께 진행되고 있다. 이러한 연구의 일환으로 본 연구는 컴포넌트와 같은 시스템 분해 관점에서 객체 상호간 결합링크 속성의 분석을 통하여 서브시스템 분해를 위한 기초자료를 구축하고자 한다. 이전까지의 연구들이 개별객체를 평가하고 수치화하여 이를 누적하는 방식이었다면 이번 연구는 소프트웨어 복잡도와 밀접한 관계가 있는 객체간 상호간의 링크결합관계를 분석 대상으로 선정하고 객체간 링크의 속성분석 및 결합강도 예측에 기계학습을 활용한 새로운 관점에서의 소프트웨어 분석 방법을 제안한다.

키워드 : 객체지향, 객체지향 메트릭, 객체결합강도, 기계학습, 소프트웨어 품질, 소프트웨어 클러스터링

1. 서 론

소프트웨어 시스템을 보다 작은 서브 시스템으로 분해하는 주제는 소프트웨어의 패러다임의 변화에 상관없이 소프트웨어 관련분야의 주요연구 대상이다. 소프트웨어 시스템을 좋은 그룹으로 나누었을 때에 해당 소프트웨어의 유연성과 이해도 향상 그리고 개발 시간의 단축과 관련 있다는 것은 오래 전부터 소프트웨어 분야에서는 알려진 내용이다[1].

※ 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2012R1A2A1A01010775).
† 정 회 원 : Enpix 기술지원팀장
†† 준 회 원 : UCLA, Dept. of Integrative Biology and Physiology, Postdoctoral Scholar
††† 준 회 원 : 연세대학교 컴퓨터과학과 박사과정
†††† 종신회원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2012년 12월 24일
수 정 일 : 1차 2013년 4월 2일
심사완료 : 2013년 6월 20일
* Corresponding Author : Sanghyun Park(sanghyun@cs.yonsei.ac.kr)

객체지향 설계에서도 객체간 상호관계의 증가와 소프트웨어 복잡도의 상관관계 분석과 같은 연구들을 기초로 하여 객체간의 좋은 그룹을 생성하기 위해 컴포넌트 기반 방법론 같은 방법들이 제안되었다. 또한 객체지향 및 컴포넌트 기반 개발방법론이 금융권과 같은 대형 SI 소프트웨어 개발까지 적용범위가 확대되어 복잡해진 소프트웨어 시스템을 서브시스템으로의 분해하는 기준에 대한 관심과 중요성이 높아지고 있다. 이를 위하여 소프트웨어 클러스터링과 같은 연구를 통하여 대형 시스템을 서브시스템으로 분해하는 기준을 찾기 위한 다양한 연구가 시도되었다[2][3].

이와 더불어 대형화되는 소프트웨어 시스템들로 인하여 소프트웨어의 품질 역시 주요 토픽이 되었으며 소프트웨어의 품질을 측정하기 위한 다양한 메트릭(Metrics)의 연구가 진행 되었다[4][5][6]. 일반적으로 소프트웨어의 품질 척도는 모듈간의 낮은 결합도와 높은 응집도을 갖는 객체의 설계를 목표로 한다. 이러한 결합도와 응집도는 소프트웨어 시스템을 서브 시스템으로 분해하는 방법으로도 유용하게 활용된다. 단일 객체 관점의 결합도/응집도는 모듈이나 객체 자체 품질 향상에 집중하지만 서브 시스템 분해관점에서는 서브시스템이나 객체 그룹 품질 향상에 초점을 맞춘다. 객체 그룹의 품질을 높이기 위해서는 객체나 모듈간의 결합 강도가 높은 링크를 연결하고 결합 강도가 낮은 링크를 제거하는 방법으로 객체 그룹의 응집도를 높이고 결합도를 줄여서 객체 그룹의 품질을 높일 수 있다. 따라서 이러한 분류 처리에 주요 척도가 되는 객체간 링크의 속성에 대한 다양한 관점에서의 연구가 필요하다. 예로서 객체간의 정보로는 구현관계, 상속관계, 호출관계, 종속관계 등의 객체간 링크유형을 살펴보면 상속관계가 호출관계에 비해 상대적으로 높은 응집도를 갖는다. 이렇게 객체 상호간 링크특성들의 분석을 통하여 연결된 객체들간에 상대적 결합강도를 비교 분석할 수 있다. 하지만 객체지향 설계에서 객체간 결합 강도는 상속관계와 호출관계 같이 명확하게 구분되는 것만 있는 것이 아니기 때문에 객체간 링크 속성에 대한 다각도의 분석을 필요로 한다. 객체간 결합 강도 예측에는 또 다른 고려사항이 있다. 그것은 객체지향 소프트웨어를 설계할 때에는 문제영역에 적합한 객체 설계 패턴이 적용되는 특징이 있다는 점이다. 서로 다른 문제영역의 소프트웨어를 획일적인 기준으로 예측하는 것은 적절하지 못한 예측방법이 될 것이다. 따라서 각 소프트웨어 특성에 적합한 예측방법이 제공되어야 한다. 객체지향 소프트웨어 분야에서의 서브시스템 분해에 관한 연구는 소프트웨어 클러스터링 분야[2]에서도 진행되었지만 이전 연구들도 개별 소프트웨어에 동일한 기준을 모듈 분해의 지표로 삼기 때문에 서로 다른 객체 디자인 패턴의 적용에 따른 차이를 수용하는 부분에 한계가 있었으며 객체의 속성과 객체간의 링크 특성을 고려한 다면적인 연구는 찾아 보기 힘들다. 그래서 이번 논문은 객체의 속성과 객체 상호간 관계 속성의 연관관계 분석을 각 소프트웨어별로 수행하는 방식으로 각 소프트웨어의 특성이 반영된 객체간 링크 결합강도 분류 메소드를 제안한다.

본 제안 메소드는 객체의 특성을 내포하고 있어서 객체간 링크 결합강도 예측에 유용하게 쓰일 수 있는 객체지향 설계 품질 메트릭을 우선 선별하였다. 이와 함께 객체간 링크의 특성을 나타내는 객체 상속, 구현, 호출과 같은 객체간 관계정보와 각 객체의 추상클래스, 인터페이스, 일반클래스라는 객체속성 정보를 함께 사용한다. 결합강도 예측을 위해서 SMO, C4.5, NaiveBayes 3가지 기계학습 방법들을 사용하였으며 3가지 기계학습방법의 예측율 비교를 통하여 객체간 링크 결합강도 분류에 효과적인 기계학습 방식에 대해서 알아본다. 예측 실험은 다양한 분야의Java open source 소프트웨어들을 대상으로 하여 특정 소프트웨어 도메인에 편중되지 않도록 하였다. 그리고 소프트웨어의 기계학습 실험 데이터를 통합하여 본 논문에서 제안한 동일한 메소드로 객체간 결합강도 예측 실험을 추가적으로 진행하여 본 논문의 제안 메소드에 객체간 링크 결합강도 분류 예측방법으로써의 범용적 활용 가능성을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 객체 속성 정의를 위한 소프트웨어 객체지향 설계 메트릭과 적용된 기계학습들을 살펴본다. 그리고 객체 관계 추출을 위해 적용된 객체의 동적 바인딩 정보 추출방식에 대해서 간략히 소개한다. 3장에서는 본 논문에서 제안하고 있는 기계학습을 활용한 객체간 결합강도 분류 메소드에 대해서 구체적으로 설명한다. 4장에서는 실험 환경과 실험 대상인 오픈 소스 프로젝트들과 실험에 사용된 소프트웨어 그리고 실험 결과를 기술한다. 5장에서는 논문의 결과 및 향후 연구과제와 향후 연구 발전 방향에 대한 논의를 기술한다. 마지막으로 참고문헌에 대해서 기술한다.

2. 관련 연구

2.1 객체지향 설계 메트릭

소프트웨어 메트릭은 일반적으로 프로젝트 계획과 평가 관점에서 소프트웨어 품질 관리 영역에서 유용하게 쓰인다. 특히 이러한 관점은 객체지향 접근방식에서는 더욱 중요하다. 소프트웨어 메트릭에 관한 다양한 방식들이 제안되고 있으며 특히 measurement theory에 입각하여 제시된 방식이 많이 인용되었다[6][7][8][9]. 대표적인 객체지향 설계 메트릭은 다음의 6가지 이며, 이는 객체지향 설계 메트릭 연구에는 자주 인용되고 있는 것들 이다[4][10]. WMC (Weighted methods Per Class)는 클래스 내에 메소드의 수에 따른 정적 복잡도를 표현한다. DIT(Depth of Inheritance Tree)는 해당 클래스의 상속의 깊이다. NOC(Number of children)는 직접 상속된 sub class의 수이다. CBO(Coupling between objects) 는 상속관계를 제외한 다른 클래스와의 연계에 대한 수치다. RFC(Response for a class)는 객체의 모든 메소드와 그 메소드에 의해 호출된 메소드들의 세트의 수이다. LCOM(Lack of Cohesion in Methods)은 유사도가 제로인 메소드 쌍의 개수 이다.

2.2 활용 기계학습

본 연구에서 활용한 3가지 주요 기계 학습에 대해서 살펴본다.

SMO(Sequential minimal optimization algorithm for training SVM)은 Support Vector Machine으로부터 확장되었다. Support Vector Machine(SVM)은 1979년에 Vladimir Vapnik에 의해 제안 되었다[11]. SVM의 기본 개념은 긍정적 예제와 부정적 예제간의 최대 마진 값을 갖도록 분리하는 초 평면에 대한 정의다. 이때 초 평면이 선형의 경우의 마진값은 초 평면으로부터 가장 근접한 긍정적 예제와 부정적 예제간의 거리로 정의된다. 선형 SVM의 결과를 위한 계산식은 다음과 같다. $u = \overrightarrow{w} \bullet \overrightarrow{x} - b$

$\overrightarrow{w}$는 초 평면에 대한 노멀 벡터이고 $\overrightarrow{x}$는 입력벡터이며 긍정적 예제와 부정적 예제간의 분해를 위한 초 평면은 u=0인 평면이다[12][13].

C4.5는 ID3알고리즘의 단점들을 보완하여 발표된 Decision Tree알고리즘이다. C4.5는 ID3의 주요 알고리즘인 엔트로피 산출과 Information Gain을 수용하고 있다[14].

엔트로피는 주어진 데이터 집합내의 혼잡도를 의미하는데 이는 주어진 데이터 집합 내 레코드들의 클래스 종류 차이 정도가 엔트로피 수치로 나타난다. 결정 트리 분류 알고리즘에서는 엔트로피가 높은 상태에서 낮은 상태가 되도록 하여 트리를 구성한다.

Naive Bayesian 분류방법은 확률기반의 방식으로 정보분류에 전통적으로 많이 적용되는 분류 방법이다. Bayesian network의 하나의 형태로써 특히 단순화 측면에서 두 가지의 주요 가정을 따르고 있다. 첫 번째는 특정 클래스가 주어졌을 때 분류 대상을 표현하는 각 속성은 다른 속성들과 두 번째는 조건부 독립을 이룬다는 가정과 숨겨지거나 함축된 속성은 예측 처리에 영향을 주지 않는다는 가정이다. 이러한 가정을 바탕으로 하여 모델을 단순화 함으로써 여러 분야에서 다양한 문제 해결에 적용되어 왔다. 그리고 이 방식은 표현과 사용방식 측면에서 단순한 접근방식을 제공하며, 학습과 실험에서 정확한 예측을 위해 지도에 의한 학습 관점에서 설계되었다[15].

2.3 자바 패키지

패키지(Package)는 자바(Java) 프로그램의 분류를 위한 매우 중요한 메커니즘이다. 모든 자바 클래스(class)는 패키지의 일부이고 패키지는 패키지 트리 내에서 계층적 구조로 표현 된다. 이러한 이유로 패키지는 자바 프로그램의 서브(Sub) 모듈들로서 충분히 고려할 가치가 있다. 자바에서는 기능이나 설계관점에서 밀접한 연관이 있는 클래스를 동일한 패키지에 위치 시켜 클래스의 관리를 용이하게 하는데 이론적으로 패키지는 하나의 독립된 단위로 취급되기도 한다[16].

2.4 런타임 레벨의 객체분석

객체간 결합 정보의 추출과 분석에는 객체의 구조적인 특성과 정적인 코드 분석정보를 일반적으로 활용한다. 그러나 정적인 코드 분석방법은 객체지향의 특징인 다형성, 동적 바인딩에 의한 결합관계 분석의 어려움과 소프트웨어의 사용되지 않는 코드(Dead)들의 존재의 문제로 인하여 결합관계 분석에 한계가 있으며 이러한 한계로 인하여 런타임(Runtime)시의 객체간 실질적인 결합 관계를 정확하게 반영하지 못하는 문제가 있다. 이를 보완한 런타임 객체 레벨의 결합관계 분석 방식은 메소드의 실행 혹은 호출하는 객체의 변수, 반환되는 객체의 클래스들간의 종속강도를 상속, 다형성, 동적 바인딩으로 인한 호출 관계까지 세분하여 수치화 할 수 있기 때문에 정적인 소스 기반의 분석방식에 비해 정확한 객체간 결합정보를 제공한다[17].

2.5 SLOC을 기반 클래스 간의 의존성 강도 메트릭(RCBC)

SLOC(Source Line of Code)을 사용하여 클래스간 상대적인 결합 강도를 측정하는 방법이다[18]. 해당 측정 방법은 시스템 유지 보수과정에 한 클래스를 수정했을 때 리플 효과 [19]에 의해 수정된 클래스에 의존하는 클래스들도 함께 영향을 받아서 재검사를 해야 하며 그로 인한 비용이 높아지는 현상에 대한 관점에서 클래스간의 의존성 강도를 측정하였다.

메소드의 SLOC합계를 MLOC 이라고 정의하고 클래스간의 상대적인 의존성 강도(RCBC)를 아래와 같이 정의하였다.

$$RCBC(c1,c2) = \frac{MLOC(Relation(c1,c2))}{MLOC(All\,Relation)}$$

RCBC(c1, c2)는 클래스 c1이 시스템의 변화에 의해 재검사 유발될 가능성 중 클래스 c2에 의해 재검사가 유발될 가능성을 나타내는 상대적인 의존성 강도의 메트릭이며 각 객체간 결합 강도를 0 ~ 1사이의 수치로 표현할 수 있는 방식이다. 하지만 SLOC라는 하나의 속성만으로 호출 메소드와 피호출 메소드간 결합의 영향도를 정의하여 객체 결합이 가지고 있는 복잡한 객체간 상호 관계를 고려하지 못하였고, 메소드 변경에 대한 재검사 영향도라는 한정된 관점에서만 객체간 결합 강도를 해석하여 해당 소프트웨어에 문제영역 해결을 위한 설계 패턴과 같은 각 소프트웨어가 가지고 있는 특성을 반영하지 못하였다.

3. 제안 메소드

3.1 제안 메소드의 개요

본 논문에서 제안하는 메소드는 우선 소프트웨어내의 객체간의 결합강도를 예측하기 위하여 런타임 객체간 링크를 추출한다. 런타임 객체간 링크정보의 추출에는 Apache Byte Code Engineering Library(BCEL)[20]을 사용하였다. 추출된 런타임 객체의 링크 정보는 두 가지 학습 데이터로 가공된다. 먼저 자바 패키지의 특성을 근거로 하여 객체간 링크정

보를 강결합/ 약결합의 결합강도 구분해서 기계학습 예측 데이터로 활용 했다. 동일 패키지와 근거리 패키지에 위치하는 클래스간의 관계는 강결합으로 판단하고 그 외의 결합은 약결합으로 판단하는 방법이 적용되었다. 또한 추출된 런타임 객체 링크 정보는 객체지향 설계 메트릭을 산출하는데도 활용 하였다. 본 논문에서 사용한 대부분의 객체지향 설계 메트릭의 수치는 관련 객체간 링크 개수의 합계이기 때문에 객체간 호출 및 관계형태에 따라 해당 수치의 급격한 편차를 보인다. 이러한 수치는 기계학습에 좋지 않은 영향을 줄 수 있기 때문에 각 수치를 정규화하여 기계학습에 적합한 수치가 되도록 가공한다. 객체지향 설계 메트릭 수치와 정규화된 수치가 학습에 유효하다고 판단하여 두 수치 모두 학습 정보로 활용하였다. 그리고 본 논문에서는 각 객체간의 상호관계 관점에서의 분류를 목표로 하기 때문에 해당 메트릭 수치를 직접 사용하지 않고 두 객체의 수치의 차이와 평균을 학습자료로 활용하도록 설계하였다. 런타임 객체간 링크 정보가 마지막으로 쓰이는 부분은 객체간 링크의 특성을 나타내는 구분에 활용되었다. 각 객체간 링크 관계를 자바의 상속관계, 구현관계, 함수참조관계, 멤버변수 참조관계로 분류하여 기계학습 정보로 활용한다. 또한 각 링크의 시작 객체와 끝 객체의 유형을 추출하여 추가 학습 정보로 활용하고 있다. 이러한 단계를 통해 가공된 학습 데이터는 대표적인 기계학습 방법인 결정 트리 방식의 C4.5, SVM 기반의 SMO, 확률기반의 NavieBaye의 3가지 기계학습 방법으로 해당 소프트웨어의 강결합/ 약결합 분류 예측 실험을 수행할 수 있도록 하였다. 그리고 다수의 기계학습 속성으로 인한 기계학습 예측율 저하가 발생하는 것을 최소화하기 위하여 Relief 알고리즘을 활용하여 속성의 수를 줄였다. Relief알고리즘을 통하여 각 속성의 스코어를 산출하고 속성의 스코어 정보를 바탕으로 낮은 스코어의 속성을 제거하는 방법을 통하여 Overfitting으로 인한 예측율 저하를 최소화하였다.

3.2 런타임 레벨의 객체간 관계 추출기법

메트릭 데이터 수집을 위해서 BCEL을 사용 했다. BCEL은 binary java class file의 분석 및 생성 그리고 조작 위해서 사용되는 API이다. 모든 클래스들은 BCEL이 제공하는 객체들로 가공되어 제공 되는데 대상 클래스에 대한 메소드, 필드 그리고 바이트코드(byte code) 명령어등과 같은 객체의 다양한 정보를 포함하고 있다. 또한 바이너리(Binary)코드를 기반으로 하기 때문에 주석이나 사용되지 않는 코드는 모두 제거된 상태로 제공 되며, 상속과 같이 정적(static) 코드 분석을 통해서는 얻기 어려운 동적관계정보를 얻을 수 있게 된다[20].

3.3 자바 패키지 특성을 활용한 결합강도 기초 학습 데이터 생성

관련연구에서 이미 언급하였듯이 자바 프로그램에서 동일 패키지 내에 존재하는 클래스들은 Java 프로그램의 객체 설계 관점에서 매우 강한 결합 강도를 갖도록 설계된다[16]. 이를 바탕으로 추출된 링크가 동일한 패키지 내에 위치한 두 객체간의 링크인 경우 해당 링크의 결합강도를 강결합으로 분류하였다. 또한 자바 프로그램의 패키지가 계층 구조로 구성되는데 서로 다른 패키지에 존재하는 클래스와의 링크 결합강도 측정에는 패키지 계층간의 거리를 활용 하였다.

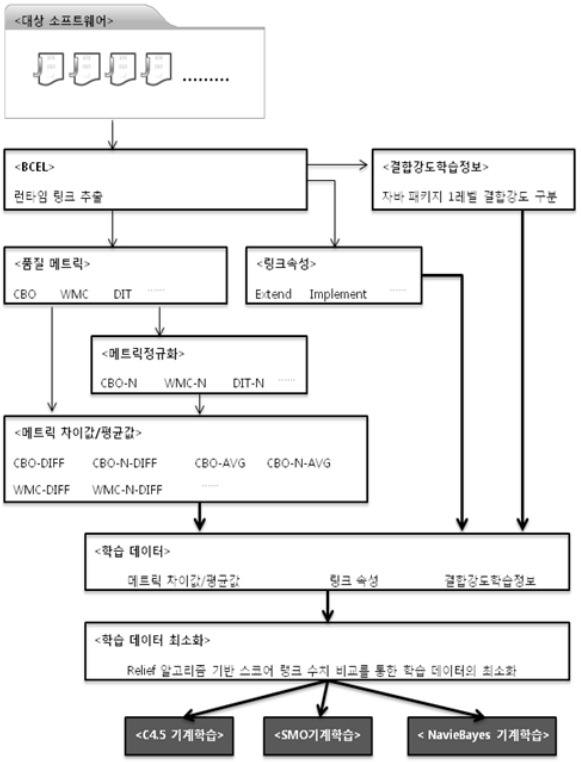


Fig. 1. Proposal method of classifying strength of dependency between objects

3.4 기계학습 항목 및 항목 속성

기계학습을 위한 항목과 항목이 가질 수 있는 허용값 및 데이터를 위해서 기계학습의 분류 구분을 결합강도(STRONG, WEAK)로 정의 하였으며 링크 유형(EXTEND, METHOD_VARIABLE, IMPLEMENT, MEMBER_VARIABLE), 시작 대상의 형태(CLASS, ABSTRACT, INTERFACE)와 끝 형태(CLASS, ABSTRACT, INTERFACE)를 항목으로 활용 하였다. 다양한 객체지향 설계 메트릭(WMC[4], DIT[4], NOC[4], CBO[4], RFC[4], LCOM[4], LCOM_[10], DAC_[7], ICH[6], ICP[6], IH_ICP[6], NIH_ICP[6], NOA[6], NOD[21])들의 두 객체간의 평균값, 차이값, 그리고 메트릭 수치를 정규화한 값의 평균값, 차이값을 기계 학습 항목으로 선정하여 실험을 진행하였다.

3.5 객체지향 설계 Metrics 정규화

객체지향 설계 Metrics 수치를 학습 데이터로 활용하는 것에 추가하여 객체지향 설계 Metrics의 정규화를 통하여 객체지향 설계 Metrics를 객체간 링크 결합강도 예측율 학

습에 적합하게 가공하여 학습 데이터로 활용한다. 객체지향 설계 Metrics 수치는 객체간 링크간에 발생하는 링크 또는 파라메터의 수에 비례하여 증가하기 때문에 수치간에 편차가 클 수 있다. 본 실험에서는 이러한 편차가 큰 수치를 가공하여 0과1사이의 수치로 정규화하고 있다. 본 연구에서 적용한 정규화 방법은 전체 데이터에서 해당 객체지향 설계 Metrics의 수치가 위치하는 지점을 수치화하는 방법을 적용하였다. 특정위치까지의 객체지향 설계 Metrics 데이터의 누적 개수를 전체 데이터 개수를 나누는 방식으로 0~1사이의 값을 갖도록 정규화 하였다.

3.6 객체지향 설계 메트릭 수치의 차이값과 평균값

객체지향 설계 메트릭 수치를 직접 기계학습 데이터로 활용하지 않고 객체간 상대적 비교를 위해서 연결된 객체의 객체지향 설계 메트릭 수치의 차이와 평균을 활용하는 방식을 적용하였다. 두 객체간 속성의 차이 비교를 위해 객체지향 설계 메트릭의 수치의 차이값을 산출하였으며, 객체지향 설계 메트릭 수치의 방향적 특징을 위해 평균값을 산출하여 실험을 진행하였다.

4. 실험 환경 및 결과

4.1 실험 소프트웨어 및 실험 환경

기계학습 소프트웨어로 Weka 3.7 & API[22]를 사용하였다. 그리고 객체간 링크 추출을 위해서 BCEL를 활용한 자바 어플리케이션을 작성하여 실험하였다. Weka에서 제공하는 API를 활용한 자바 어플리케이션을 개발하여 기계 학습(C4.5: J48, SMO, NavieBayes)을 활용한 결합강도 예측 부분의 실험을 진행 하였다. 학습 항목의 수에 따른 오버피팅(Overfitting) 및 최적 학습 항목 추출과 같은 랭크 관련 실험은 Weka Application(ReliefFAttributeEval[23] - Ranker)을 활용하여 진행하였다.

4.2 대상 소프트웨어 선별기준

자바 오픈 소스 프로젝트 가운데에서 여러 프로젝트에서 자주 사용되고 있으며 소프트웨어 버전이 일정 수준 이상으로 품질이 보장되는 소프트웨어를 실험 대상으로 선정하였다. 다양한 분야에 소프트웨어들을 실험 대상으로 하여 객체지향 설계의 도메인 종속적인 부분을 배제하였다. 또한 포함 클래스 수와 각 객체간 관계의 강/약 결합 분포도가 좋은 소프트웨어를 실험 대상으로 선정하였다.

4.3 클래스간 패키지 거리에 따른 결합강도 학습데이터 구성

본 논문에서는 자바 패키지의 특성을 기반으로 하여 결합강도 학습데이터를 생성하였다. Fig. 2에서처럼 동일 패키지 내에 클래스간의 링크와 직접 연결된 1레벨 상위 패키지와 1레벨 하위 패키지간의 링크를 강결합으로 분류 하였다.

1레벨 패키지 거리를 강결합으로 규정한 것은 결합강도가 높은 상속(extends)과 구현(implement) 링크의 객체간 패키지 거리에 관한 실험결과에 따른 것이다. 본 논문에서 채택한 객체간 패키지 거리 기반의 강결합 분류방식 검증을 위해 이번 논문의 실험 대상이 되는 소프트웨어 23개의 상속/구현관계를 갖는 객체간 패키지 거리를 조사하였다. Fig. 3에서와 같이 상속/구현 객체관계가 2레벨이상의 링크 거리보다 현저히 높은 비율로 1레벨 이내에 존재함을 알 수 있다. 상속/구현관계의 링크는 약 평균 75.67 %가 1레벨 거리 내에서 관계를 맺고 있었다. 이러한 실험 결과와 함께 자바 패키지가 가지고 있는 함축적인 객체패키지 방식에 근거하여 1레벨 이하 객체간 패키지 거리의 객체간 결합을 강결합으로 규정하였다

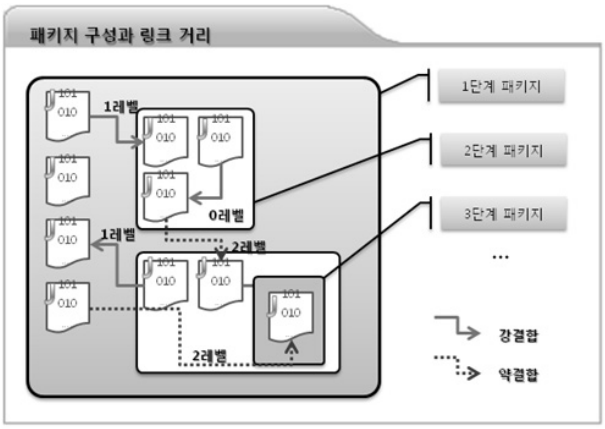


Fig. 2. Structure of java package and distance of link between objects

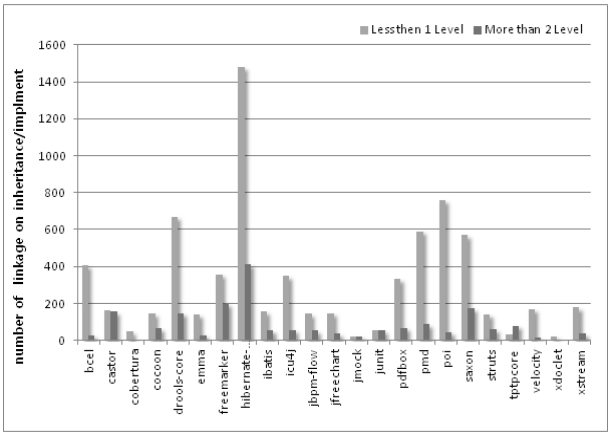


Fig. 3. Distance of inheritance/implementation by software

4.4 실험 대상 소프트웨어들의 기계학습별 예측율

본 메소드는 객체간 링크의 결합 강도 예측을 특정 소프트웨어가 가지고 있는 설계 패턴이 감안된 학습 데이터를 바탕으로 기계학습방법을 사용하여 강결합과 약결합으로 분류한다. 제안된 메소드가 객체간 링크 결합강도에 관하여 높은 수치의 예측율을 보인다면 객체와 객체간의 상대적 결합강도라는 소프트웨어의 모듈화 작업에 중요한 지표를 얻게 된다.

본 실험은 23개의 자바 오픈 소스 소프트웨어들을 대상으로 수행하였다. 각 소프트웨어의 링크 정보는 제안된 메소드를 통하여 객체지향 설계 메트릭으로 변환, 정규화 작업을 거쳐 실험 데이터로 가공된 이후에 C4.5, SMO, NaiveBayes 기계학습을 통하여 강결합/약결합으로 분류하는 실험을 진행하였다.

3개의 기계학습의 실험결과는 다음과 같다. C4.5의 경우 94.72 % ~ 76.96 % 예측율을 보였고, SMO는 92.64 % ~ 73.27 %, NaiveBayes는 86.50 % ~ 64.42 %의 수치를 보였다. 소프트웨어에 따라 약간의 예측율 차이를 보였으나 3가지 기계학습 모두에서 유의한 예측율을 보였다. 또한 기계학습별 평균 예측율과 표준편차에서도 C4.5의 경우 87.05 %(±0.042), SMO는 80.56 %(±0.049), NaiveBayes는 73.63 %(±0.056)의 수치를 보였다. 결합강도 측정 기계학습 방법별 예측율은 최대/최소값 및 평균, 표준편차 값을 기준으로 하여 C4.5 > SMO > NaiveBayes 순의 결합강도 예측율을 보였다. 본 연구가 제안하는 메소드에 의한 예측율은 객체간 링크의 결합강도에 대한 것 이다. 이러한 결합강도는 서브 시스템 모듈화 작업과 컴포넌트 구성을 위한 객체 그룹을 분류할 때 활용될 수 있다. 특히 객체간 링크를 활용하는 그래프 클러스터링 기법에서는 매우 효과적인 정보가 될 것으로 예상한다.

추가적으로 Table 1에서도 알 수 있듯이 각 기계학습간 상대적 예측율이 3가지 기계학습에서 거의 유사하게 나타났다. 이는 각 학습방법에 따라 예측율의 차이는 있으나 해당 소프트웨어를 분류하고 있는 지표가 개별 기계학습 방법 내에서는 유사할 수 있다는 추측이 가능하다. 이러한 분류 지표의 유사성에 관해서는 다음 절에서 제안 메소드의 범용적 활용 관점에서 랭크 스코어 비교 방법을 활용하여 다시 살펴볼 것이다.

4.5 객체간 링크의 특징과 기계학습 예측율간의 관계 분석

본 연구의 실험 결과를 살펴보면 결정 트리 방식의 C4.5이 SMO와 NaiveBayes 보다 최대/최소 및 평균, 표준 편차의 모든 면에서 좋은 성능을 보였다. 이것은 객체지향 설계를 하였을 때 객체간에 나타나는 관계가 가지는 특성에 의

Table 1. Prediction for strength of dependency between objects by machine learning with C4.5, SMO, NaiveBayes

소프트웨어	C4.5	SMO	NaiveBayes
bcel	94.72 %	92.64 %	86.50 %
castor	90.59 %	84.22 %	76.16 %
cobertura	88.59 %	84.03 %	80.23 %
cocoon	81.68 %	76.60 %	72.10 %
drools-core	88.81 %	78.24 %	71.86 %
emma	89.68 %	83.62 %	75.76 %
freemarker	92.58 %	87.80 %	77.02 %
hibernate-core	83.21 %	73.27 %	68.98 %
ibatis	88.73 %	79.91 %	74.07 %
icu4j	82.00 %	78.37 %	71.81 %
jbpm-flow	86.63 %	79.00 %	64.58 %
jfreechart	82.10 %	77.49 %	72.32 %
jmock	88.77 %	81.28 %	75.94 %
junit	76.96 %	70.85 %	64.42 %
pdfbox	87.06 %	80.09 %	76.90 %
pmd	92.23 %	88.00 %	85.56 %
poi	88.82 %	79.95 %	71.20 %
saxon	86.70 %	79.37 %	74.11 %
struts	85.15 %	77.53 %	65.36 %
tptpcore	90.04 %	83.71 %	75.50 %
velocity	86.41 %	78.06 %	71.20 %
xdoclet	88.07 %	83.03 %	72.02 %
xstream	82.53 %	75.86 %	69.80 %
평균	87.05 %	80.56 %	73.63 %
표준편차	0.0416	0.0491	0.0560

한 현상일 것이다. 객체지향 설계를 바탕으로 하여 품질 높은 소프트웨어 개발을 하려고 할 때에는 설계패턴들을 도입하게 된다. 이러한 설계패턴들은 일반적으로 “패턴 이름”, “문제점”, “해결책”, “결과”라는 4개의 주요 요소를 가지고 있다[24]. 자주 반복되거나 특정 도메인에서 자주 발견되는 객체설계상의 문제점에 대해서 반복되는 실험적 경험을 바탕으로 그 문제점에 가장 적합한 해결책을 제안하고 그것에 대한 장단점을 미리 알려주는 지식 전달 방법으로 활용된다. 이러한 설계 패턴을 기반으로 설계된 객체지향 소프트웨어는 적용 설계 패턴이 가지고 있는 특성을 대부분 따르게 된다. 설계 패턴들은 도메인의 문제해결을 위해서 독특한 형태의 객체관계를 형성하게 되는데 이러한 특징이 결정 트리 방식인 C4.5 기계학습에서 높은 예측율이 나타나는 현상을 보인 것으로 생각한다. 결정 트리는 학습데이터가 이산적이며 이분 분류일 때 좋은 성능을 보이는데 노드간 엔트로피 산출 및 점진적 분류 방식[25]이 객체지향 소프트웨어에 적용된 설계패턴의 특성에 따라 객체간 관계 분석에 잘 적응되어 상대적으로 높은 예측율을 보였다고 생각한다. NaiveBayes은 가우시안 분포를 가정한 확률기반 분류 방법[26]이다. 하지만 객체지향 설계는 도메인의 문제점을 해결하기 위해 도메인에 맞는 독특한 설계 기법을 사용하는 특성이 반영되어 객체지향 메트릭 수치의 분포도가 가우시안 분포와는 많은 차이를 보인다. 가우시안 분포를 가정한 확률기반의 NaiveBayes방식의 실험에서는 낮은 예측율를 보일 가능성이 높다. SVM 기반의 SMO 분류 방식은 설계패턴 특성에 따라서 일정한 형태로 설계된 객체지향 시스템 설계의 성질에 의해 유의한 예측율을 보인 것으로 분석된다. SMO 기계학습은 특정 설계패턴을 기반으로 한 소프트웨어의 적용 설계패턴에 영향을 받아 초평면이 산출될 것으로 예상되며 이러한 초평면에 의한 분류가 유의한 수준의 객체간 링크 결합강도 분류가 가능했던 것으로 판단된다.

4.6 통합 데이터의 기계학습별 예측율

범용적인 결합강도 분류 방식으로써의 활용 가능성을 확인하기 위해서 다수의 소프트웨어로부터 추출된 학습데이터들을 하나의 데이터로 통합하여 본 논문의 제안된 메소드로 결합강도를 분류하는 추가실험을 진행하였다. 통합된 데이터에서도 유의한 예측율을 보인다면 범용적인 결합강도의 분류 방식으로써 본 논문의 제안 메소드의 활용 가능성을 검증할 수 있기에 추가실험을 진행하였다. 검증을 위해서 23개 소프트웨어로부터 추출한 학습 데이터를 통합하여 동일한 메소드로 결합강도를 예측하는 실험을 진행하였다. 실험 결과는 Table 2과 같이 C4.5의 경우 83.64 %, SMO은 74.25 %, NaiveBayes은 51.92 % 예측율을 보였다. 예측율 비교 우위는 개별 소프트웨어와 동일하게 C4.5 > SMO > NaiveBayes 순서를 보였다. 그리고 통합 학습데이터의 예측율 수치가 개별 실험에 비해 조금 낮아졌으며 NaiveBayes의 경우에는 유의 범위에서 벗어났다.

이러한 예측율의 하락은 개별 소프트웨어가 직면한 문제상황의 해결을 위해 선택한 객체지향 설계 패턴의 차이로 인한 것으로 예측된다[24]. 하지만 C4.5과 SMO의 경우에는 높은 유의 예측율을 보였으며 이러한 두 방식의 유의한 실험결과를 근거하여 범용적인 결합강도 분류 방법으로써 본 제안 메소드가 활용될 수 있음을 유추할 수 있다.

Table 2. Prediction of machine learning about strength of dependency between objects based on summarized data

기계학습	예측율
C4.5	83.64 %
SMO	74.25 %
NaiveBayes	51.92 %

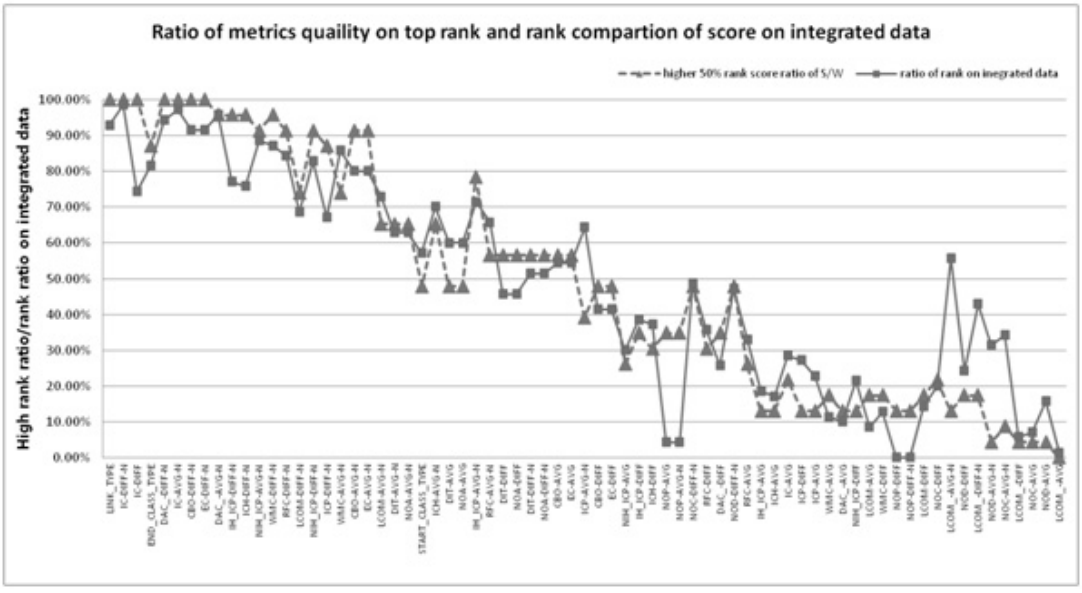


Fig. 4. Comparison between ratio of high rank on quality metrics and rank score on summarized data

추가적으로 범용적인 결합강도 분류에 관련한 데이터 비교를 진행하였다. Rank 알고리즘(ReliefFAttributeEval, Ranker - weka)을 활용하여 상위 스코어를 산출하여 각 속성들의 영향도를 확인 하였다. 개별 소프트웨어 별로 랭크 스코어가 상위 50 % 에 해당하는 객체지향 설계 메트릭을 선별하고 랭크 스코어가 상위 50 %이상 되는 소프트웨어의 개수를 확인해 본 결과에서 랭크 스코어가 높은 메트릭은 여러 소프트웨어에서도 유사하게 높은 랭크 스코어를 보였다. 이는 객체지향 설계 메트릭 가운데에 객체간 결합 강도 분류에 범용적으로 영향을 주고 있는 속성들의 존재를 확인할 수 있는 실험 결과이다. 이와 더불어 실험 소프트웨어의 통합 기계학습 데이터의 랭크 스코어와 개별 소프트웨어별 랭크 상위 50 % 객체지향 설계 메트릭의 소프트웨어 적용 비율 수치 비교를 통해 본 논문의 제안 메소드의 범용적 적용가능성을 보다 명확히 확인할 수 있다. 개별 소프트웨어에서 상위 50 % 랭크 스코어에 포함된 개수가 많은 학습항목이 통합 데이터의 랭크 스코어에서도 상위 스코어를 보여주고 있다. 개별 소프트웨어에서의 상위 랭크 비율과 통합 데이터에서의 랭크 스코어는 Fig. 4에서 알 수 있듯이 두 그래프간 0.92의 PEARSON 값을 갖는 매우 유사한 추이를 보인다. 이러한 실험 결과는 본 논문이 제안한 메소드와 학습 항목들이 범용적 결합강도 분석에도 활용될 수 있음을 보여주는 실험결과로 분석된다.

5. 결론 및 향후 연구

본 논문은 객체지향 소프트웨어의 서브시스템 분해를 위한 기초 데이터 구축을 목표로 하였다. 이를 위해 객체지향 소프트웨어에서 각 객체들간의 링크 결합강도를 측정하는 메소드를 제안하였다. 객체지향 설계 메트릭을 객체의 속성으로 규정하고 이러한 속성의 비교를 통하여 객체간 상호관계를 수치화한 데이터는 객체간 링크 결합강도 분류에 매우 효과적인 기준 데이터가 될 수 있음을 다양한 기계 학습을 활용한 결합강도 분류 실험으로 확인하였다. 또한 개별 소프트웨어 실험결과에서 상당히 높은 예측율을 보이는 점을 통하여 소프트웨어가 각자의 문제의 영역을 해결하기 위해서 유사한 설계 패턴을 적용하고 개별 소프트웨어 내에서의 객체간 결합은 비슷한 유형으로 적용되었음을 예측할 수 있었으며, 본 논문의 제안 메소드가 소프트웨어의 객체간 결합 강도 분류에 효과적인 메소드임을 확인 하였다. 또한 추가 실험을 통하여 범용적인 결합강도 분류 방식으로의 활용 가능성을 엿볼 수 있었다.

또한 본 연구에서 제안한 메소드와 측정된 객체간 결합강도의 강/약결합 예측데이터의 가치는 논문 서두에서 언급한 것처럼 소프트웨어 시스템의 서브 시스템 분해 연구에서 보다 확실히 나타날 것으로 생각한다. 예로써 제안된 메소드로 분류된 높은 수준의 예측율는 갖는 결합강도가 클러스터링 기법과 접목한다면 객체간의 그룹을 생성하는 처리에 가중치와 같은 데이터로 활용되어 더 효과적인 클러스터링이 되도록 할 수 있기 때문에 실험을 통해서 보여준 예측율과 결합강도 데이터는 이후 서브 시스템 분해 관점의 연구를 통해서 그 가치가 다시 평가될 것으로 생각한다. 그리고 본 논문은 결합강도의 분류의 수준을 강결합과 약결합의 수준으로만 분류하였지만 이를 보다 세분화하거나 수치화할 수 있다면 객체간 결합관련 정보의 질을 높일 수 있을 것이다. 그리고 실험 결과에 대한 각 기계학습의 세부 내용의 분석과 상위 랭크 스코어, 객체지향 설계 메트릭 수치들을 함께 비교 분석 통하여 객체간의 결합강도 분류의 원리를 유추할 수도 있을 것이다. 또한 결정 트리의 실험 데이터 수치나 경향 분석을 통하여 객체지향 메트릭과 결합강도간의 정성적 분석도 향후 연구로 남아 있다.

참 고 문 헌

[1] D. L. Parnas, “On the criteria to be used in decomposing systems into modules” Communications of the ACM, 1972, Vol.15, No.12, pp.1053-1058.

[2] S. Xanthos, “Clustering object-oriented software systems using spectral graph partitioning.” ACM Student Research Competition, 2005.

[3] P. Rousseeuw, A. Struyf, M. Hubert, “Clustering in an object-oriented environment.” Jaurnal of Statistical Software, 1996, pp.1-30.

[4] S. R. Chidamber, C. F. Kemerer, “Towards a Metrics Suite for Object Oriented design” Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA’91), October 1991, 26 (11) pp.197-211.

[5] D. P. Tegarden, S. D. Sheetz, D. E. Monarch, “A Software Complexity Model of Object-Oriented Systems” Decision Support Systems, 1995, Vol.13, No.3-4, pp.241-262.

[6] Y. S. Lee, B. S. Liang, S. F. Wu, F. J. Wang, “Measuring the Coupling and Cohesion of an Object- Oriented Program Based on Information Flow” International Conference on Software Quality, Maribor, Slovenia, 1995.

[7] W. Li, S. Henry, “Object-oriented metrics that predict maintainability.” Journal of Systems and Software, 1993, Vol.23, No.2, pp.111-122.

[8] M. H. Tang, M. H. Kao, M. H. Chen, “An Empirical Study on Object Oriented Metrics.” Sixth Int’l Software Metrics Symp, 1999, pp.242-249.

[9] J. K. Chhabra, V. Gupta, “Package Coupling Measurement in Object-oriented Software.” Journal of Computer Science and

Technology 2009, Vol.24, No.2, pp.273-283.

[10] S. R. Chidamber, C. F. Kemerer, "A metrics suite for object oriented design." IEEE Transactions on Software Engineering, 1994, Vol.20, No.6, pp.476-493.

[11] V. Vapnik, "Estimation of Dependences Based on Empirical Data"Springer-Verlag, 1982.

[12] V. Vapnik, "The Nature of Statistical Learning Theory." Springer-Verlag, 1995.

[13] J. C Platt, "Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods: support vector learning." MIT Press, Cambridge, MA, 1999.

[14] J. R. Quinlan, "C4.5: Programs for Machine Learning." Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[15] G. H. John, P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers." Proc. 11th Conf. on Uncertainty in Artificial Intelligence , 1995, pp.338-345.

[16] M. Fowler, "Reducing Coupling." IEEE Software July august, 2001, pp.102-105.

[17] A. Mtichell, J. F. Power, "Using object-level run-time metrics to study coupling between objects." ACM Symposium on Applied Computing. Santa Fe,New Mexico, USAObjects, 2005.

[18] J.M. Hwa , S.H. Lee , Y.R. Kwon, "A Coupling Metric for Measuring Strength of Dependency between Classes in Object-Oriented Systems." KIISE, Korea Computer Congress, 2007, Vol.34, No.1, pp.33-34.

[19] D. C Kung, J. Gao, P. Hsia, Y. Toyoshima, and C. Chen, "On Regression Testing of Object-oriented Software Maintenance" The Journal of Syst. And Software 1996, Vol.32, No.1, pp.21-40.

[20] D. Markus, "Byte code engineering library (BCEL), version 5.1, April 25 2004. http://jakarta.apache.org/bcel/".

[21] A. Lake, C. Cook, "Use of factor analysis to develop OOP software complexity metrics." Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.

[22] I. H. Witten, E. Frank, "Data Mining: Practical machine learning tools and techniques." Morgan Kaufmann, San Francisco. 2nd Edition, 2005.

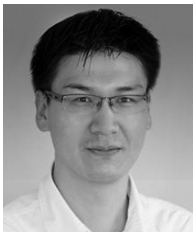
[23] K. Kira, L. Rendell, "A practical approach to feature selection.", Proc. the Ninth International Workshop on Machine Learning, Morgan Kaufmann Publishers Inc, 1992, pp.249-256.

[24] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Design." Addison-Wesley, Reading, MA 1995.

[25] T. Michell, "Machine Learning" Boston, McGraw-Hill, MA 1997.

[26] D. Heckerman, D. Geiger "Learning Bayesian networks: a unication for discrete and Gaussian domains." Proc. Eleventh Conf. on Uncertainty in Articial Intelligence, 1995, pp.274-284.

[27] E. Arisholm, L. C. Briand, A. Foyen, "Dynamic coupling measures for object-oriented software." IEEE Transactions on Software Engineering, 2004, Vol.30, No.8, pp.491 - 506.



정 성 균

e-mail : nvirus@enpix.co.kr
1998년 조선대학교 컴퓨터공학과(학사)
2013년 연세대 공학대학원 컴퓨터공학과 (석사)
1999년~현 재 Enpix 기술지원팀장
관심분야 : Software engineering & Machine learning

안 재 균

e-mail : jgahn@ucla.edu
2006년 연세대학교 컴퓨터과학과(학사)
2009년 연세대학교 컴퓨터과학과(석사)
2013년 연세대학교 컴퓨터과학과(박사)
2013년~현 재 UCLA, Dept. of Integrative Biology and Physiology, Postdoctoral Scholar
관심분야 : Bioinformatics, Data mining, Database system

여 윤 구

e-mail : yyk@cs.yonsei.ac.kr
2009년 연세대학교 컴퓨터과학과(학사)
2011년 연세대학교 컴퓨터과학과(석사)
2011년~현 재 연세대학교 컴퓨터과학과 박사과정
관심분야 : Bioinformatics, Data mining

박 상 현

e-mail : sanghyun@cs.yonsei.ac.kr
1989년 서울대학교 컴퓨터공학과(학사)
1991년 서울대학교 컴퓨터공학과(석사)
2001년 UCLA 컴퓨터과학과(공학박사)
1991년~1996년 대우통신 연구원
2001년~2002년 IBM T. J. Watson Research Center Post-Doctoral Fellow
2002년~2003년 포항공과대학교 컴퓨터공학과 조교수
2003년~2006년 연세대학교 컴퓨터과학과 조교수
2006년~2011년 연세대학교 컴퓨터과학과 부교수
2011년~현 재 연세대학교 컴퓨터과학과 교수
관심분야 : Database, Data mining, Bioinformatics, Index structure for Flash memory, SSD