KIPS Tr. Software and Data Eng.
Vol.4, No.6 pp.247~252 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.6.247

Validation Technique for Class Name Postfixes Based on the Machine Learning of Class Properties

Hongseok Lee[†] · Junha Lee[††] · Illo Lee[†††] · Soojin Park[††††] · Sooyong Park[†††††]

ABSTRACT

As software has gotten bigger in magnitude and the complexity of software has been increased, the maintenance has gained in-creasing attention for its significant impact on the cost. Identifiers have an impact on more than 90 percent of the readability which accounts for a majority portion of the maintenance activities. For this reason, the existing works focus on domain-specific features based on identifiers. However, their approaches have a limitation when either a class name does not reflect the intention of its context or a class naming is incorrect. Therefore, this paper suggests a series of class name validation process by extracting properties of classes, building learning model by applying a decision tree technique of machine learning, and generating a validation report containing the list of recommendable postfixes of classes to be validated. To evaluate this, four open source projects are selected and indicators such as precision, recall, and ROC curve present the value of this work when it comes to five specific postfixes including functional information on class names.

Keywords : Software Maintenance, Readability, Class Name, Machine Learning

클래스 특성 기계학습에 기반한 클래스 이름의 접미사 검증 기법

이 홍 석[†] · 이 준 하[††] · 이 일 로[†††] · 박 수 진[††††] · 박 수 용[†††††]

요 약

소프트웨어의 규모가 커지고 복잡성이 증가함에 따라 소프트웨어의 유지보수가 보다 중요해지고 있으며 유지보수성에 많은 영향을 미치는 요인 중 하나는 소스코드 가독성이다. 가독성의 90% 이상 영향을 끼치는 요인은 소스코드에서 사용되는 식별자들의 이름이며 이를 위한 기존 연구들에서는 클래스의 식별자로 사용된 어휘를 이용하여 식별자의 이름을 검증한다. 하지만 대부분의 관련 연구는 그 특성상 개체의 도메인 관련 특성만을 고려하게 되며 클래스 내의 어휘가 적절하지 못한 경우 적용할 수 있는 범위가 한정적이라는 한계점이 있다. 본 논문에서는 클래스의 특성을 추출하여 의사결정트리 기법을 통해 기계학습을 시킨 후 클래스 역할 모델을 생성하며 이를 이용하여 이름을 검증할 대상 클래스의 역할에 해당하는 접미사를 추천하게 되어 클래스 이름 검증 보고서를 생성한다. 본 연구 기법의 효용성을 검증하기 위해 4개의 오픈소스 프로젝트에 대하여 본 연구 기법을 적용하였고 클래스 역할 정보를 담고 있는 5개의 접미사에 대해 정확도와 재현율, ROC 곡선과 같은 지표를 제시하였다.

키워드 : 소프트웨어 유지보수, 가독성, 클래스 이름, 기계학습

1. 서 론

소프트웨어 산업의 급격한 발전에 따라 오늘날의 소프트웨어는 과거에 비하여 규모가 커지고 복잡성이 증가하였다. 이러한 배경으로 소프트웨어의 유지보수 비용 또한 증가하고 있다[1]. 전체 소프트웨어 개발 생명주기 중 유지보수 단계에서 소요되는 비용은 소프트웨어 개발에 드는 총비용의 약 67%를 차지하고 있다[2, 3]. 따라서 소프트웨어 개발 비용을 줄이기 위해서는 유지보수 비용을 절감하는 것이 필수적이며 소프트웨어 유지보수에 있어 가독성이 높은 소스코드는 매우 중요하다. 코드의 가독성이란 코드가 의도하는 동작이나 알고리즘 등을 얼마나 쉽게 이해하도록 작성되었는지를 뜻하는 것으로, 가독성이 높다는 것은 코드가 이해하기 쉽게 잘 작성되었음을 의미하며 유지보수성 또한 높음을 의미한다. 이러한 가독성에 영향을 끼치는 요인으로는

※ 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No.2012M3C4A7033348)
† 준 회 원 : 서강대학교 융합소프트웨어연구소 연구원
†† 준 회 원 : Kettering University(General Motors Institute) 박사후과정
††† 비 회 원 : 서강대학교 컴퓨터공학과 박사과정
†††† 종신회원 : 서강대학교 서강미래기술연구원 교수
††††† 정 회 원 : 서강대학교 컴퓨터공학과 교수
Manuscript Received : December 24, 2014
First Revision : March 27, 2015
Accepted : April 7, 2015
* Corresponding Author : Sooyong Park(sypark@sogang.ac.kr)

크게 이름 짓기, 들여쓰기와 같은 코드의 포맷과 코드에 부연된 주석 등의 코딩 스타일 등이 있으며, 이 중에서도 이름 짓기, 다시 말해 식별자의 이름이 가독성에 무려 90%의 영향을 끼친다[4]. 가독성을 향상시킬 수 있는 좋은 이름의 특성으로는 도메인에 관련된 논의를 할 수 있는 어휘를 제공해야 하며 시스템이 어떻게 동작할지에 대한 미묘한 부분까지도 설명할 수 있어야 한다는 점이 있다[5]. 하지만 여러 사람에 의하여 소프트웨어 개발 및 유지보수가 이루어지는 환경에서는 종종 클래스의 이름에 시스템의 동작을 알 수 있게 해주는 정보가 누락되는 경우가 발생한다. 이러한 정보의 누락은 가독성 및 유지보수성의 저하를 초래하여 결국 유지보수 비용 상승을 야기시킨다. 따라서 이러한 문제를 해결하는 클래스의 의도를 잘 반영한, 좋은 이름을 판별하거나 제안하기 위한 연구들이 있다[6-8]. 대부분의 기존 연구에서는 클래스의 식별자로 사용된 어휘를 이용하여 클래스의 이름을 검증한다. 하지만 이러한 대부분의 관련 연구는 식별자에 사용된 어휘를 이용한, 즉 도메인 관련 특성만을 고려한 접근법이며 클래스 내의 어휘가 소스코드의 의도를 적절히 반영하지 못하거나 잘못 작성된 경우 적용할 수 있는 범위가 한정적이라는 한계점이 있다.

이에 따라 본 논문에서는 식별자 이외의 클래스에서 이용할 수 있는 정보로부터 해당 클래스의 시스템 내부적 역할을 식별하고 클래스 이름에 시스템이 어떻게 동작할지에 대한 정보가 누락되었는지를 판단한다. 또한 정보가 누락된 경우 클래스 이름에 식별된 시스템의 역할에 대한 정보를 클래스 이름의 접미사에 포함시킬 수 있도록 제안함으로써 기존 관련 연구의 한계를 해결하고자 한다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 소프트웨어의 유지보수성 향상을 위해 소스코드 내부 개체의 이름을 연구한 사례를 소개하고 비교한다. 3절에서는 본 연구에서 제안하는 클래스 특성을 이용한 클래스의 이름을 검증하는 기법에 대해 설명하고, 4절에서는 본 논문의 방법을 적용하여 수행한 실험의 결과를 보이며 논문에 사용된 기법들의 적합성을 검증한다. 마지막으로 5절에서는 결론 및 향후 연구를 다룬다.

2. 관련 연구

소프트웨어의 유지보수성 향상을 위해 소스코드 내부 개체의 이름을 토대로 한 연구들이 있다.

Simon Butler 등의 연구[6]에서 프로그램 이해도의 증진과 유지보수의 향상을 위해 Naming Convention의 관점에서 클래스 이름을 분석하였다. 클래스 이름이 Naming Convention을 준수하는가를 검증하기 위하여 60개의 오픈소스 프로젝트로부터 120,000개의 클래스 이름을 분석한 결과, 전체 클래스의 개수 중 73%가 [명사], 12%가 [형용사][명사]의 형태로 작성되어있다는 결과를 얻었다. 하지만 해당 연구에서는 클래스 이름이 어떠한 형태로 작성되었는지만을 분석하였을 뿐 클래스 이름이 유지보수의 향상을 위하여 잘 작성된 결과물인지에 대한 검증은 누락되어있어, 이를 기존 프로젝트 이해도의 증진과 유지보수의 향상을 위한 방안으로 적용하기에는 한계점이 따른다.

한편, Jeremy Singer 등[7]은 클래스의 역할에 따른 패턴을 분류한 연구인 Micro Pattern[15]과 클래스의 이름 사이의 관계를 분석한다. 미리 정의된 클래스의 구조적 패턴에 따라 그에 해당하는 클래스 이름과의 관계를 분석한 연구로 Comparable, Assert, Exception 이상 3가지의 이름을 가지는 클래스들이 각각 Micro Pattern에서 정의한 패턴 중 어떤 패턴에 해당하는지를 예를 들어 설명하며 클래스의 패턴에 따라 적절한 이름을 검증할 수 있음을 설명하였지만 실제로 이 연구만으로 클래스의 이름을 검증할 수는 없으며 그 가능성만을 보여주었다.

또한 Samir Gupta 등[8]의 연구에서는 클래스에 사용된 식별자들의 품사를 태깅하여 해당 연구에서 제안하는 4가지의 분류로 메소드 이름의 구성요소를 분석하였다. 20개의 오픈소스 프로젝트에서 11,546개의 식별자 분석을 바탕으로 하

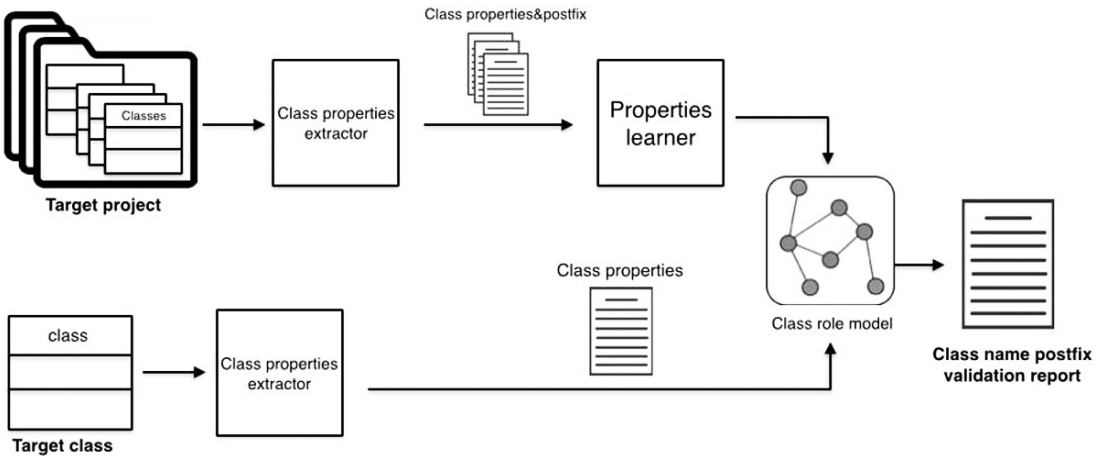


Fig. 1. Overall Validation Process for Class Name Postfix

여 작성된 메소드의 이름을 Leading Verb, Reactive Names, Implicit Action Verb, Non-Leading Action Verb 이상 4가지로 분류하였다. 그러나 이 연구에서는 메소드 이름의 구조를 분석 및 분류하는 데 그쳤고 시스템 내의 메소드 이름만을 연구하여 유지보수성에 큰 영향을 미치는 클래스의 이름을 고려하지 않았다는 제한점이 존재한다.

3. 클래스 특성을 이용한 클래스 이름의 접미사 검증 기법

본 절에서는 학습 대상이 되는 기존 프로젝트의 클래스 이름에 반영된 시스템 역할 특성을 기반으로 검증 대상 프로젝트의 클래스 이름의 접미사를 검증하는 기법에 대해 논한다. 먼저 학습의 대상이 되는 프로젝트로부터 클래스의 역할을 판독할 클래스 특성들을 추출하는 기법을 설명하고, 다음으로 추출된 클래스의 특성들을 기반으로 클래스 이름에 작성된 시스템 내의 클래스 역할을 판별하기 위한 모델을 생성하는 학습 과정에 대하여 기술한다. 마지막으로 생성된 클래스 역할 모델을 이용하여 검증 대상 클래스 이름의 접미사를 검증하는 기법에 대하여 자세히 기술한다.

Fig. 1는 본 연구에서 제안하는 클래스 이름 검증의 전체 프로세스를 나타낸다. 위 그림은 학습 대상 프로젝트의 클래스들을 입력으로 이용하여 검증 대상의 '클래스 이름 검증 보고서'를 생성하기 위한 일련의 과정을 개략적으로 보여준다.

3.1 클래스 특성 추출

오라클에서 제공하는 문서[13]에 따르면 클래스는 클래스의 이름 외에도 접근 제한자, 상속 관계, 반환값의 타입 등의 요소들을 포함한다. 클래스를 정의하기 위해서는 필수 혹은 선별적으로 요소들을 순서에 맞게 배치해야 한다. 클래스에 포함되는 필드 및 메소드 또한 미리 약속된 문법에 따라 요소들을 배치한다.

Table 1. Class Properties

Prop. #	Property
1	isAbstract
2	isInterface
3	isImplemented
4	nonPrimitiveMethodRatio
5	exceptionMethodRatio
6	hasChild
7	isEnum
8	collectionsFieldRatio
9	nonPrimitiveFieldRatio

이는 다시 말해 기존 연구에서 사용하였던 가변적 식별자 외에 클래스의 특성을 반영하는 선언 요소를 구성하는 예약어들을 이용하여 클래스의 특성을 추출할 수 있음을 말하며, 위 Table 1은 클래스 선언 요소들로부터 알 수 있는 클래스의 특성들이다.

다음 Table 2는 JHotDraw프로젝트의 DefaultDOMFactory 클래스로부터 특성들을 추출한 결과이다. 추출된 9개의 클래스 특성과 하나의 접미사로 구성된 집합은 클래스 역할 모델을 생성하기 위한 기계학습의 인스턴스로 사용된다. 이때 접미사는 클래스 이름의 마지막 단어를 말하며 시스템 내 클래스의 역할에 대한 정보는 대부분 클래스 이름의 접미사에 담겨있으므로 접미사를 추출하고 검증하게 된다.

Table 2. JHotDraw DefaultDOMFactory Class Example

Prop. #	1	2	3	4	5	6	7	8	9	Postfix
Value	0	0	1	0.56	0.0	0	0	1.0	1.0	Factory

3.2 클래스 역할 모델 생성

본 연구에서는 주어진 데이터의 종류를 알고 있고 클래스의 이름을 검증하기 위한 역할 모델을 생성하기 때문에 분류분석(교사 학습)을 사용하였다. 분류분석 기법에는 의사결정트리 기법, 나이브 베이즈 분류 기법, 베이지안 네트워크, 뉴럴 네트워크 등 다양한 학습 기법들이 존재하며 본 연구에서는 의사결정트리 기법 중 하나인 C4.5[14]를 사용한다. C4.5의 알고리즘은 의사결정트리 기법인 ID3 기법에 기초하여 ID3 기법의 범주형 속성에 대해서만 트리를 생성하는 한계 등을 보완한 기법이다. 다음 Algorithm 1은 C4.5의 알고리즘을 수식화한 것이다.

Input: an attribute-valued dataset D
1: Tree = {}
2: **if** D is "pure" OR other stopping criteria met **then**
3: terminate
4: **end if**
5: **for all** attribute $a \in D$ **do**
6: Compute information-theoretic criteria if we split on a
7: **end for**
8: a_{best} = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests a_{best} in the root
10: D_v = Induced sub-datasets from D based on a_{best}
11: **for all** D_v **do**
12: $Tree_v$ = C4.5(D_v)
13: Attach $Tree_v$ to the corresponding branch of Tree
14: **end for**
15: **return** Tree

Algorithm 1. C4.5 Algorithm

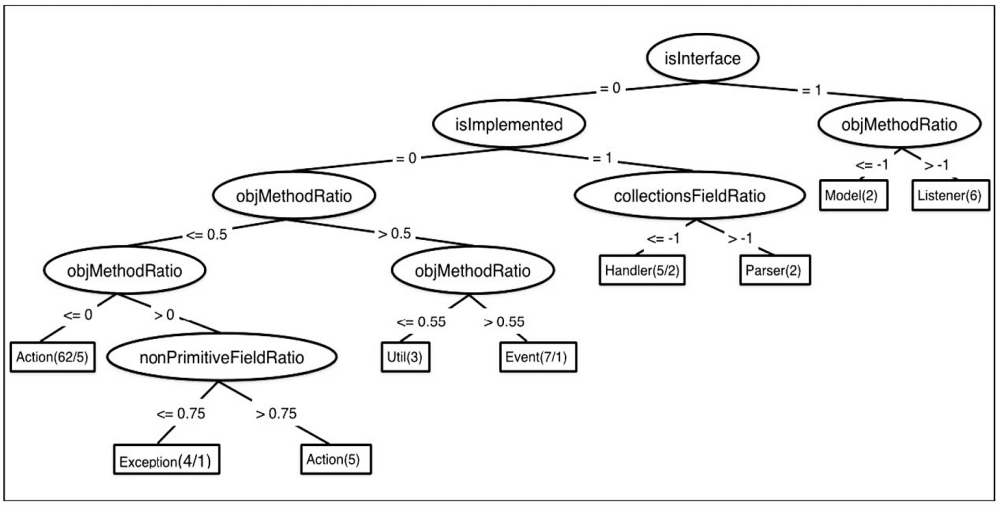


Fig. 2. Class Role Model Example

```
Report Summary :
    project name: JMeter
    filter : not applied
    target class : all
    the number of classes : 1013

=============================================================================================================
class name                                                  | recommend postfix | warning | prob
=============================================================================================================

org.apache.jmeter.extractor.BSFPostProcessor                |  Impl             |    +    | 0.9
org.apache.jmeter.protocol.http.control.AuthManager         |  Listener         |    +    | 0.75
org.apache.jmeter.engine.JMeterEngineException              |  Exception        |    -    | 0.929
org.apache.jmeter.visualizers.RequestView                   |  Impl             |    +    | 0.909
```

Fig. 3. Class Name Validation Report Example

본 연구에서 의사결정트리를 채택한 이유는 다음과 같다. 의사결정트리를 통한 데이터의 분석 결과는 트리구조로 표현되기 때문에 분석이 용이하여 본 연구에서 특성 학습 및 역할 판독기를 통해 생성되는 클래스 역할 모델을 사용자가 분석하기가 보다 쉽다. 또한 범주형과 수치형 데이터를 모두 학습시켜 분석자에게 유의미한 수치를 함께 제공하기 위하여 의사결정트리 기법들 중 C4.5 알고리즘을 사용하였다.

C4.5 알고리즘을 통해 학습 대상 프로젝트 클래스들의 특성들과 접미사들에 대한 의사결정트리가 산출되며 이는 검증 대상 클래스의 특성에 해당하는 접미사를 추천하게 되는 클래스 역할 모델이다. Fig. 2는 Java오픈소스 프로젝트인 JHotDraw를 클래스 특성 및 역할 판독기의 입력으로 생성된 클래스 역할 모델의 예이다.

3.3 클래스 역할 모델을 이용한 클래스 이름의 접미사 검증

3.2에서 생성된 클래스 역할 모델을 통해 검증 대상 프로젝트 클래스들의 이름을 검증하게 된다. 클래스 역할 모델에 의해 검증 대상 클래스의 역할에 따른 접미사를 추천하게 되며 추천된 접미사와 검증 대상 클래스 이름의 접미사를 비교하여 검증 보고서를 생성한다. 다음 Fig. 3는 본 논문에서 제안하는 기법의 최종 산출물이며 Java 오픈소스 프로젝트인 JMeter의 클래스 이름들을 검증한 검증 보고서이다.

검증 보고서의 *recommend postfix*는 검증 대상 클래스의 역할에 해당하는 본 기법을 통한 추천 접미사이며 *warning*은 추천된 접미사와 검증 대상 클래스 이름의 접미사가 다른 경우 '+'기호를 통해 검증자에게 클래스 이름의 접미사를 다시 고려해야 할 필요가 있음을 알린다. *prob*항목은 의사결정트리를 통해 추천 접미사가 검증 대상 클래스의 접미사로 분류될 확률값을 보인다. Fig. 3의 검증 첫 항목을 보면 BSFPostProcessor 클래스는 본 논문의 기법을 통해 접미사로 Impl이 추천되었으며 본래 클래스 이름의 접미사와 다르므로 *warning*은 '+' 표시되었으며, 0.9의 확률로 추천되었음을 알 수 있다. 실제로 BSFPostProcessor클래스는 DOMFactory라는 인터페이스를 구현하고 있는 클래스이므로 이는 바르게 추천되었다고 할 수 있다.

4. 검 증

본 연구에서는 연구 기법의 효용성을 검증하기 위해 Java 오픈소스 프로젝트인 ArgoUML[10], JMeter[12], jEdit[11], JHotDraw[9]에 대하여 변이 테스팅(Mutation Testing) 기법을 사용하여 검증을 진행하였다. 본 연구의 기법을 변이 테

Table 3. Postfix List

Postfix	Listener	Factory	Manager	Handler	Exception	Action	Event	Parser	Util	Model	Impl

스팅 하기 위해 먼저 검증 대상 프로젝트 클래스들 중 클래스 이름의 접미사에 클래스 역할에 대한 정보를 담고 있는 것들을 검증 대상으로 한다. 그다음 각각의 검증 대상 클래스들의 이름에서 접미사를 삭제하여 클래스 이름의 접미사에 클래스의 역할 정보가 빠져있는 변이 클래스로 만든다. 이렇게 변이된 클래스는 이름이 잘못 지어진 클래스의 역할을 하게 되며 본 연구 기법으로 찾아내야 할 대상이 된다. 결론적으로 본 검증은 이름이 변이된 클래스들의 집합을 정답 집합으로 놓고 본 연구의 기법을 통해 정답 집합의 변이된 클래스들을 얼마나 잘 찾아내는가를 통해 이루어지게 된다. 이때 본 검증에서 가정은 접미사를 제거하여 변이 클래스를 만든 정답 집합의 구성요소인 클래스들이 시스템 내부적인 역할에 대한 정보를 실제로 접미사를 통해 잘 반영하고 있었다고 두는 것인데 검증에 사용된 오픈소스 프로젝트들은 여러 유사 연구에서도 사례 연구를 위해 사용되었으며 객체지향 학습을 위한 용도로도 사용될 만큼 코드의 구조와 완성도가 높다고 알려졌다. 이상 4개의 프로젝트들에 대하여 변이테스트를 진행하기 위해 클래스 역할에 대한 접미사를 가지고 있는 클래스들만을 대상으로 하여 진행하였으며 접미사들의 목록은 다음 Table 3의 11개이다.

그리고 본 연구에서는 검증을 위해 기계학습을 통한 연구에서 범용적으로 사용되는 검증 기법인 k 겹 교차 검증(k-fold cross validation)을 사용하며 한 프로젝트당 10번씩, 10-fold cross validation을 진행하여 평균값을 검증의 결과로 사용하였다.

검증에서 사용된 4개 각각의 프로젝트들에 대해 10-fold cross validation을 진행하여 얻은 결과는 여러 연구의 검증에서 일반적으로 사용하는 정확도(precision)와 재현율(recall), 그리고 ROC 곡선(Receiver Operating Characteristic Curve)을 통해 본 연구의 검증 지표로 활용된다. 검증 결과로 대상 클래스들의 이름에서 접미사를 제거하여 본 연구 기법을 적용했을 때 원래 포함하던 접미사를 얼마나 정확히 추천해 주는지를 보였으며 다음 Table 4는 검증에 사용된 프로젝트 중 3개 이상의 프로젝트에서 높은 빈도로 출현한 Action, Impl, Listener, Handler, Factory를 클래스 이름의 접미사로 가지는 클래스에 대한 정확도와 재현율, ROC Area 각각의 평균값과 최댓값, 최솟값, 표준편차값을 나타낸다.

Table 4. Evaluation Results

	Precision	Recall	ROC Area
Avg.	0.678	0.703	0.868
Max.	0.923	0.968	0.967
Min.	0	0	0.464
Std.	0.291	0.304	0.215

결과를 보면 평균정확도는 0.678, 평균재현율은 0.703, 평균 ROC Area는 0.868로 유의미한 결과를 보임을 알 수 있다. 하지만 표준편차값이 아주 낮지 않음을 알 수 있는데 이는 검증 대상 중 JHotDraw프로젝트에서 Factory, Impl의 접미사를 가지는 클래스가 각각 1건과 0건으로 정확도와 재현율이 0의 값을 가지게 되어 평균과 표준편차의 대푯값에 영향을 끼쳤음을 알 수 있다.

5. 결 론

소프트웨어의 규모가 커짐에 따라 유지보수에 드는 비용 또한 상승하여 그 중요성이 점점 더 부각되고 있다. 유지보수성에 큰 영향을 끼치는 가독성의 향상을 위한 연구들이 있지만 대부분의 기존 연구에서는 클래스의 식별자로 사용된 어휘를 이용한다. 하지만 이러한 대부분의 관련 연구는 도메인 관련 특성만을 고려한 접근법이며 클래스 내의 어휘가 소스코드의 의도를 적절히 반영하지 못하거나 잘못 작성된 경우 적용할 수 있는 범위가 한정적이라는 한계점이 있다.

본 논문에서는 기존 연구의 한계를 극복하기 위해 클래스의 특성을 이용하여 연구를 진행하였다. 클래스의 특성을 추출하여 의사결정트리 기법으로 기계학습을 시킨 후 클래스 역할 모델을 생성하였고 생성된 모델을 통해 이름을 검증할 대상 클래스의 역할에 해당하는 접미사를 추천하게 되며 연구 기법의 최종 산출물로 클래스 이름 검증 보고서를 생성한다.

본 연구 기법의 효용성을 검증하기 위해 JHotDraw, ArgoUML, JMeter, jEdit 등 4개의 오픈소스 프로젝트에 대하여 본 연구 기법을 적용하였고 클래스 역할 정보를 담고 있는 5개의 접미사에 대해 평균 0.678의 정확도와 0.703의 재현율, 0.868의 ROC Area의 수치로 검증을 수행하는 결과를 보였다.

추후 연구에서는 본 연구에서 활용한 클래스의 특성과 더불어 메소드의 특성 또한 고려하여 클래스 이름을 검증하는 것에 있어 정확도를 더 개선하고 접미사뿐만 아니라 접두사 및 어근에 대한 정보를 활용하여 그에 대해서도 검증이 가능하게 하여 클래스 역할에 대한 정보가 누락된 클래스를 검증하는 것에서 더 나아가 가독성이 높은 클래스 이름을 새롭게 제안할 수 있도록 연구할 예정이다.

References

[1] Y. Ren, T. Xing, and X. Chai, “Research on Software Maintenance Cost of Influence Factor Analysis and Estimation Method,” 3rd International Workshop on

Intelligent Systems and Applications, pp.1-4, 2011.
[2] M. Fowler, "Refactoring: Improving the Design of Existing Code," Addison-Wesley, 1999.
[3] B. Boehm, V. R. Basili, "Software Defect Reduction Top 10 List," *Computer*, Vol.34, No.1, pp.135-137, 2001.
[4] Robert C. Martin, "Clean Code: A Handbook of Agile Software Craftsmanship," Pearson Education, p.309, 2008.
[5] William C. Wake, "Refactoring Workbook," Pearson Education, p.39, 2003.
[6] Simon Butler, Michel Wermelinger, Yijun Yu, and Helen Sharp, "Mining Java Class Naming Conventions," 27th IEEE International Conference on Software Maintenance, pp.93-102, 2011.
[7] Jeremy Singer, Chris Kirkham, "Exploiting the Correspondence between Micro Patterns and Class Names," 8th IEEE International working conference on Source Code Analysis and Manipulation, pp.67-76, 2008.
[8] Samir Gupta, Sana Malik, Lori Pollock, and K. Vijay-Shanker, "Part-of-Speech Tagging of Program Identifiers for Improved Text-Based Software Engineering Tools," IEEE 21st International Conference on Program Comprehension(ICPC), pp.3-12, 2013.
[9] JHotDraw [Internet], http://www.jhotdraw.org.
[10] ArgoUML [Internet], http://argouml.tigris.org.
[11] jEdit [Internet], http://www.jedit.org.
[12] JMeter [Internet], http://jmeter.apache.org.
[13] ORACLE Java Documentation [Internet], https://docs.oracle.com/javase/tutorial/java/javaOO/classdecl.html.
[14] J. Ross Quinlan, "C4.5: Programs for machine learning," Morgan Kaufmann Publishers, 1993.
[15] Y. Gil, I. Maman. "Micro Patterns in Java Code," 20th annual ACM SIGPLAN conference on Object Oriented Programming Systems Languages and Applications, pp.97-116, 2005.

이 홍 석

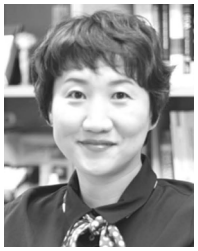
e-mail : acdc01421@gmail.com
2012년 건국대학교 의용메카트로닉스(학사)
2012년~2015년 서강대학교 컴퓨터공학과 (석사)
2012년~현 재 서강대학교 융합소프트웨어연구소 연구원
관심분야 : 요구공학, 소프트웨어 유지보수, 소프트웨어 아키텍처

이 준 하

e-mail : zuna.lee@gmail.com
2007년 단국대학교 컴퓨터과학과(학사)
2010년 서강대학교 컴퓨터공학과(석사)
2014년 서강대학교 컴퓨터공학과(박사)
2014년~현 재 Kettering University (General Motors Institute) 박사후과정
2015년~현 재 Magna Electronics in USA, System Engineer
관심분야 : 확장형 지능 로봇 소프트웨어 플랫폼, 자동차 소프트웨어

이 일 로

e-mail : 215-b@hanmail.net
2000년 전북대학교 전기전자제어공학부 (학사)
2003년 광주과학기술원 기전공학과(석사)
2003년~2008년 LG전자 DTV 연구소 선임연구원
2008년~현 재 국방기술품질원 기술기획본부 선임연구원
2013년~현 재 서강대학교 컴퓨터공학과 박사과정
관심분야 : 지능형 소프트웨어공학, 데이터마이닝, 자가적응형 소프트웨어



박 수 진

e-mail : psjdream@sogang.ac.kr
1995년 경북대학교 컴퓨터공학과(학사)
2000년 서강대학교 정보통신대학원(석사)
2008년 서강대학교 컴퓨터공학과(박사)
1995년~1999년 (주)LG-CNS 근무
2000년~2002년 한국래쇼날 소프트웨어 선임 컨설턴트
2010년~현 재 서강대학교 서강미래기술연구원 교수
관심분야 : 소프트웨어 재사용, 요구공학, 소프트웨어 아키텍처

박 수 용

e-mail : sypark@sogang.ac.kr
1986년 서강대학교 컴퓨터공학과(학사)
1988년 Florida State University, Computer and Information Science(석사)
1995년 George Mason University, Information Technology(박사)
1998년~현 재 서강대학교 컴퓨터공학과 교수
관심분야 : 요구공학, 동적 아키텍처