The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 6, pp.145-150, Dec. 31, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.6.145
JIIBC 2017-6-19

회귀 분석을 사용한 소스 코드 가독성 메트릭 분석

Metric Analysis of Source Code Readability using Regression Analysis

최상철*, 김순태***, 이정휴*, 유희경**

Sangchul Choi*, Suntae Kim*, Jeong-Hyu Lee* and Hee-Hyung Yoo****

요 약 소프트웨어 유지보수는 소프트웨어 생명주기에서 전체 비용의 많은 부분을 차지하고 있다. 소프트웨어를 유지보수하기 위해서는 코드를 읽는 것이 필수적이고, 이는 유지보수 활동에서 가장 많은 시간이 소요되는 활동이다. 코드의 가독성은 사람이 소스코드를 이해하는데 드는 어려움의 정도를 측정하는 메트릭이다. 코드의 가독성이 좋을수록 사람이 소스코드를 이해하기 쉬워진다. 본 논문에서는 소스코드의 가독성을 이분법적으로 판단하는 기존의 연구보다 향상된 기법을 사용하여, 개발 중인 소스코드의 가독성 정도를 정량적으로 측정할 수 있는 새로운 소스코드 가독성 메트릭을 제안한다. 이를 평가하기 위해, 우리는 설문조사를 수행하고 가독성을 가장 잘 설명하는 척도를 찾을 수 있도록 회귀 분석 기법을 사용해 그 결과를 분석할 것이다.

Abstract Software maintenance accounts for a large portion of the software life cycle cost. In the software maintenance phase, comprehending the legacy source code is inevitable, which takes most of the time. Source code readability is a metric of the extent of code readers' difficulty of code comprehension based on the source code itself. The better the code is readable, the easier it is for code readers to comprehend the source code. This paper proposes novel source code readability metric to quantitative measure the extent of current source code under development, which is more enhanced measurement method than previous research that dichotomously judges whether the source code was readable or not. As an evaluation, we carried out a survey and analyzed them with Regression Analysis to find best parameters of the metric.

Key Words : code readability, metric, software maintenance

Ⅰ. 서 론

소프트웨어 유지보수는 소프트웨어 생명주기에서 전체 비용의 많은 부분을 차지하고 있다[1]. 여기에서 소스코드를 읽는 활동은 소프트웨어 유지보수에서도 가장 많은 시간이 소요되는 활동이다[2]. 소스코드의 가독성은 프로그램 코드를 사람이 얼마나 쉽게 읽을 수 있는지를 나타내는 것으로 정의된다. 따라서, 소스코드의 가독성은 코드를 읽는 시간에 큰 영향을 미치는 요소이고 이에 따라 소프트웨어 유지보수의 매우 중요한 요소 중 하나이다[2].

소스코드 가독성 연구는 이전에도 많이 있었다. 기존

*정회원, 전북대학교 소프트웨어공학과
**정회원, 강원대학교 컴퓨터공학과
***정회원, 전북대학교 소프트웨어공학과, CAIIT
접수일자: 2017년 9월 7일, 수정완료: 2017년 10월 25일
게재확정일자: 2017년 12월 8일

Received: 7 September, 2017 / Revised: 25 October, 2017 / Accepted: 8 December, 2017
***Corresponding Author: stkim@jbnu.ac.kr
Dept. of Software Engineering, Chonbuk National University, Korea

의 연구들은 기본적으로 샘플 코드 선정, 속성 추출, 설문조사, 통계 분석의 순으로 소스코드의 가독성을 판별하기 위한 연구를 진행했다. Buse는 소스코드로부터 얻을 수 있는 여러 정적인 측면들을 메트릭으로 정의하여 가독성을 측정하고자 했다. 여기엔 LOC(Lines of Code), 코드 내 주석의 수, 식별자의 수, 공백의 수 등이 포함되었다[2]. 또 다른 연구자 Halstead는 코드의 연산자 및 피연산자들의 수를 기반으로 한 여러 메트릭을 정의하여 이를 바탕으로 소스코드의 가독성을 측정하고자 했다[3]. 또, Posnett은 기존의 Buse와 Halstead가 정의한 메트릭 중 가장 코드를 잘 설명하는 메트릭을 선정하여 소스코드의 가독성을 측정하기 위한 메트릭을 축소하기 위한 시도를 하기도 했다[4]. 앞서 언급한 기존의 연구들 및 그 외 많은 소스코드 가독성 관련 연구들은 소스코드가 읽을 만한지, 아닌지를 따지는 이분법적인 분류를 하는 것을 주된 목표로 하고 있다. 따라서 이를 통해 코드의 가독성이 얼마나 되는지를 정량적으로 알 방법이 없다.

본 논문은 소스코드의 가독성을 정량적으로 측정하는 모델을 제안하고자 한다. 이를 위해 본 논문에서는 소스코드의 가독성 메트릭 측정을 위한 척도를 정의하고 설문조사 및 통계적 방법을 통해 이를 정제하고 검증할 것이다. 만들어진 모델을 통해 소스코드의 가독성을 정량적으로 측정할 수 있다면 특정 소스코드의 가독성이 얼마나 되는지를 객관적으로 볼 수 있는 기준이 될 것이다. 본 연구의 기여는 크게 두 가지로 나눌 수 있다. 첫째, 소스코드 가독성 메트릭 측정을 위한 척도를 제안한다. 둘째, 메트릭을 사용해 소스코드 가독성을 정량화한다.

본 논문은 아래와 같이 구성된다. 2장에서는 앞서 언급한 Buse, Halstead, Posnett의 연구를 설명하고, 3장에서는 소스코드의 가독성을 측정하기 위한 메트릭을 제안한다. 4장에서는 설문조사를 통해 메트릭을 정제하고 여러 통계적인 방법을 통해 메트릭이 소스코드의 가독성을 얼마나 잘 설명하는지를 검증하며, 5장에서는 결론을 소개하고 본 논문을 마무리한다.

II. 관련 연구

1. Buse의 모델

Buse는 소스코드의 정적인 측면들을 이용한 가독성 측정 메트릭을 제안하고 이를 통한 소프트웨어 품질과의 관련성을 주장했다[2]. Buse가 사용한 척도들은 식별자의 길이, Indentation, 한 라인의 길이, 주석의 수 등 소스코드에서 정적으로 측정할 수 있는 요소들로 이루어져 있다. Buse는 척도를 정의한 후 총 100개의 샘플 코드를 선정한 후 120명의 사람에게 가독성을 1~5점으로 점수를 매기는 설문조사를 시행했다. 1점은 소스코드의 가독성이 매우 떨어진다는 뜻이고 5점은 소스코드의 가독성이 매우 뛰어나다는 뜻이었다. 설문조사를 마친 후 각 척도에 대한 분석을 통해 어떤 척도가 가독성에 긍정 혹은 부정적인 영향을 미치는지를 판별했다.

2. Halstead의 모델

Halstead는 소스코드의 복잡성을 계산하기 위한 방법을 소개했다[3]. Halstead는 연산자의 종류(n_1)와 총수(N_1), 피연산자의 종류(n_2)와 총수(N_2)를 통해 여러 척도를 정의했다. 척도의 목록은 다음과 같다.

- Program Length(N): $N = N_1 + N_2$
- Program Vocabulary(n): n = n1 + n2
- Volume(V): $V = N\log_2 n$
- Difficulty(D): $D = \frac{n_1}{2}\frac{N_2}{n_2}$
- Effort(E): E = DV

3. Posnett의 모델

Posnett은 가독성 측정을 위한 척도의 수를 최소한으로 줄일 방법을 제시했다[4]. Posnett은 Buse의 모델과 Halstead의 모델에서 가독성을 가장 잘 설명하는 척도를 선택하여 제안했다. 선택된 척도는 각각 LOC와 Volume이었는데, 이 외에도 소스코드에 대한 엔트로피를 계산하여 총 3가지의 척도로 소스코드의 가독성을 추정하는 모델을 제안했다.

III. 소스코드 가독성 메트릭 측정을 위한 척도

본 논문에서는 소스코드의 가독성을 측정하기 위해 표 1 같은 척도를 선정하였다. 해당 척도는 어느 언어에서도 사용할 수 있으나, 본 논문에서는 필자가 주로 사용하는 Java 언어를 기준으로 하여 척도를 설명할 것이다.

표 1. 가독성 메트릭 측정을 위한 척도
Table 1. Measures for measurement of readability metric

척도	설명
LOC	소스코드의 라인 수
numOfMethodInvocation	소스코드 내에서 호출한 메서드의 수
numOfBranch	소스코드 내에서 사용된 if, switch 등 분기문의 수
numOfLoops	소스코드 내에서 사용된 for, while 등 반복문의 수
numOfAssignment	소스코드 내에서 변수에 값을 할당하는 연산의 수
numOfComments	소스코드 내의 주석의 수
numOfBlankLines	소스코드 내의 공백 라인의 수
numOfStringLiteral	소스코드 내에서 사용된 고정 스트링의 수
numOfArithmaticOperators	소스코드 내에서 사용된 기본 사칙 연산자의 수
numOfLogicalOperators	소스코드 내에서 사용된 and, or 등 논리 연산자의 수
numOfBitOperators	소스코드 내에서 사용된 비트 단위 연산자의 수
AverageOfVariableName Length	소스코드 내 변수 이름의 평균 길이
AverageLineLength	소스코드 내 각 라인의 평균 길이
maxNestedControl	소스코드 내에서 사용된 제어문이 제어문 내에서 중첩되어 사용되었을 때, 중첩된 수 중 가장 큰 수
ProgramVolume	소스코드가 품고 있는 정보량[3]
entropy	코드의 어수선한 정도 혹은 복잡도[4]

표 1의 척도들은 경험적인 측면에서 코드의 가독성에 영향을 미칠 것으로 생각되는 요인들이 먼저 선택되었다. 그리고 관련 연구들을 조사하며 소스코드의 가독성을 잘 설명할 수 있을 것으로 생각되는 ProgramVolume과 entropy를 추가하게 되었다. 또, 표 1에서 정의된 척도는 메서드 단위의 가독성을 측정하기 위한 척도이다. 만약 클래스 단위의 가독성, 패키지 단위의 가독성, 프로젝트 단위의 가독성을 측정하고자 한다면, 위에서 언급한 척도 외의 또 다른 척도를 정의해야 할 필요성이 있다고 생각된다.

이와 같은 척도들은 그림 과 같은 형식을 통해 가독성을 정량화하게 될 것이다. 그림 1의 X(LOC)와 X(numOfMethodInvocation)은 다항식을 나타낸다. 이는 1차식 혹은 2, 3차식일 수도 있다. w_1과 w_2는 각 척도의 다항식에 대한 가중치를 의미한다.

$$\text{Readability} = w_1 \times X(LOC) + w_2 \times X(numOfMethodInvocation) + \dots$$

그림 1. 소스 코드 가독성을 측정하기 위한 식
Fig. 1. Equation for measuring source code readability

Ⅳ. 실험 및 결과

우리는 3절에서 정의한 소스코드 가독성 척도가 얼마나 소스코드의 가독성을 잘 표현하는지, 소스코드의 가독성을 표현하는데 가장 적절한 모델이 어떻게 정제되어야 할지를 알기 위해 그림 2와 같은 실험을 실행하였다.

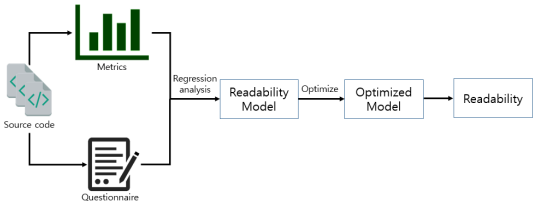


그림 2. 소스코드 가독성 모델 생성 및 정제를 위한 실험 과정
Fig. 2. Experiment progress for source code readability model generation and refinement

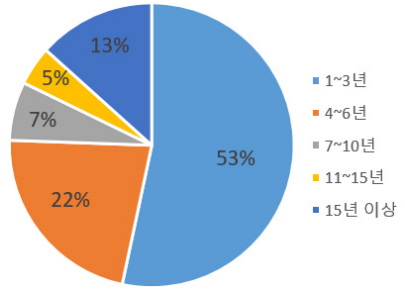
1. 실험 준비

앞에서 제시한 실험을 수행하기 위해 먼저 설문을 위한 소스코드를 준비했다. 이를 위한 소스코드는 오픈 소스 라이브러리인 antlr4 4.1.2, base-one 1.8.1.3, cglib 3.2.0, dbcp 4.2.2, fileupload 1.4, groovy 2.5.0, httpclient 3.0, javacc 6.1.0에서 추출했다.

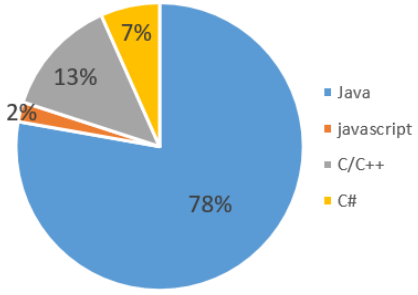
소스코드를 추출하기 위해 각 라이브러리의 메서드에 대해 표 1에서 정의한 척도를 측정했다. 그리고 측정된 척도를 바탕으로 각 척도의 수치 분포가 가능한 균등하도록 총 60개의 소스코드를 선택했다. 각 소스코드의 길이는 최소 13라인에서 최장 96라인이었다.

2. 설문조사

설문조사는 구글폼(Google form)을 사용해 시행했다. 설문조사에는 대학교 학생, 대학원 석·박사 과정 연구원, 회사원 등이며 총 45명이 참여했다. 각 참여자의 경력 및 사용 언어는 그림 3과 같다. 각 참여자는 익명으로 구글폼에 접근하여 한 명당 20개씩의 소스코드에 대한 가독성 점수를 매겼다. 가독성 점수는 1~5점으로 1점은 소스코드를 이해하기 매우 어려움, 5점은 소스코드를 이해하기 매우 쉬움을 의미한다.



(a) 경력 분포



(b) 사용 언어 분포

그림 3. 설문조사 참여자들의 경력 및 사용 언어 분포
Fig. 3. Career and language usage distribution of annotators

3. 가독성 모델 생성 및 정제 결과

우리는 설문조사 결과를 분석해 소스코드의 가독성 점수를 측정하는 모델을 만들어내기 위해 회귀 분석을 사용했다. 회귀 분석은 변수 간의 관계를 추정하기 위한 통계적인 방법의 집합이다[5]. 우리는 이중 다중 선형 회귀 분석과 비선형 회귀 분석 방법을 사용했다. 다중 선형 회귀 분석은 종속변수와 둘 이상의 독립변수 간의 관계를 추정하는 방법이다[6]. 여기서 종속변수란 독립변수의 변화 때문에 변하는 결과이고 독립변수는 결과에 영향을 미치는 원인이다. 다른 방법인 비선형 회귀 분석 방법은 종속변수와 독립변수 간의 관계를 제곱 승, 로짓변환 등을 사용해 추정하는 방법이다[7].

먼저, 다중 선형 회귀 분석을 사용해 가독성과 표 1에서 정의한 척도 간의 관계를 추정했다. 그림 4는 다중 선형 회귀 분석을 사용해 도출된 결과이다. 그림에서 Estimate 항목은 각 항을 1차 다항식으로 정리했을 때의 상수이다. 그리고 중요한 Pr은 p-value로 해당 항목이 분석자의 주장을 지지하기에 유의미한 항목인지를 알 수 있는 값이다. 일반적으로 p-value가 0.05 이하일 때 유의미하다고 한다. 다음으로 주목해야 할 항목은 Multiple R-squared이다. Multiple R-squared는 모델의 설명력을 의미하는데, 설명력이란, 만들어진 모델이 데이터를 얼마나 잘 표현하는지를 의미한다. 이를 통해 모든 척도를 사용한 모델의 설명력은 0.7717로 77.17%라는 것을 알 수 있다. 설명력이 100%에 가까울수록 모델이 데이터를 잘 설명한다는 의미이다. 다음으로, 만들어진 모델을 최적화하기 위해 단계선택(Step-wise) 기법을 적용했다. 단계선택 기법은 모델에 사용된 독립변수들의 조합을 통해 설명력에 큰 영향을 미치지 않는 독립변수들을 제거한 모델을 제공한다. 그림 5는 단계 선택 기법을 사용해 최적화한 모델의 결과를 보여준다. 해당 모델의 설명력은 74.59%로 전체 척도를 사용했을 때보다 2.58%가 낮아졌으나, 그보다 반 이상 적은 7개의 척도를 사용해 얻은 설명력이므로 가독성의 측정이 더 빠르다는 장점이 있다.

```
Residuals:
     Min       1Q   Median       3Q      Max
-1.07130 -0.27882  0.03493  0.37126  0.83169

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                   7.442e+00  9.634e-01   7.725 1.17e-09 ***
LOC                          -2.288e-02  1.156e-02  -1.979  0.05428 .
numOfMethodInvocation        -1.479e-02  1.116e-02  -1.325  0.19226
numOfBranch                   9.152e-03  4.270e-02   0.214  0.83132
numOfLoops                    4.657e-02  7.676e-02   0.607  0.54720
numOfAssignment               1.409e-02  2.394e-02   0.588  0.55928
numOfComments                 5.089e-02  2.503e-02   2.033  0.04821 *
numOfBlankLines               4.302e-02  2.774e-02   1.551  0.12826
numOfStringLiteral            3.473e-02  1.971e-02   1.762  0.08522 .
numOfArithmatifcOperators    -1.118e-02  3.236e-02  -0.345  0.73145
numOfLogicalOperators         1.069e-01  1.029e-01   1.038  0.30490
numOfBitOperators            -1.034e+00  7.053e-01  -1.466  0.14981
AverageOfVariableNameLength -4.113e-02  1.379e-01  -0.298  0.76700
AverageLineLength             2.805e-05  6.392e-03   0.004  0.99652
maxNestedControl             -1.704e-01  6.125e-02  -2.781  0.00801 **
programVolume                -1.049e-03  4.775e-04  -2.196  0.03351 *
entropy                      -5.435e-01  1.623e-01  -3.349  0.00170 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5492 on 43 degrees of freedom
Multiple R-squared:  0.7717,    Adjusted R-squared:  0.6867
F-statistic: 9.084 on 16 and 43 DF,  p-value: 3.937e-09
```

그림 4. 전체 메트릭을 사용한 다중 선형 회귀 분석 결과
Fig. 4. Multiple linear regression result using whole metric

```
Residuals:
     Min       1Q   Median       3Q      Max
-1.04081 -0.34491  0.01728  0.41666  0.81618

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      7.4219960  0.5558859  13.352  < 2e-16 ***
LOC             -0.0203190  0.0066990  -3.033  0.00377 **
numOfComments    0.0399795  0.0228265   1.751  0.08576 .
numOfBlankLines  0.0369742  0.0224644   1.646  0.10582
numOfBitOperators -0.7550834  0.5506277  -1.371  0.17617
maxNestedControl -0.1525998  0.0451605  -3.379  0.00139 **
programVolume    -0.0005915  0.0001940  -3.049  0.00361 **
entropy          -0.6106970  0.1280578  -4.769 1.54e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5269 on 52 degrees of freedom
Multiple R-squared:  0.7459,    Adjusted R-squared:  0.7117
F-statistic: 21.81 on 7 and 52 DF,  p-value: 2.051e-13
```

그림 5. 단계 선택 기법을 통해 최적화된 모델
Fig. 5. Optimized model using step-wise method

다음으로, 우리는 비선형 회귀 분석을 시행했다. 분석을 시행하기 위해 3차 다항식을 사용했는데, 그 이유는

각 척도 별 데이터를 시각화했을 때 보이는 분포가 3차 식을 따르고 있었고, 2차 식의 경우 그래프가 종 모양이기 때문에, 한쪽 끝으로 갈수록 값이 너무 급격하게 증감하기 때문에 3차 식을 선택했다. 그림 6, 7은 3차 식을 사용한 비선형 회귀 분석의 결과와 이를 최적화한 모델을 보여준다. 그림에서 보는 바와 같이 둘 다 90%를 넘어가는 설명력을 보여준다. 다만, 비선형 회귀 분석의 경우 데이터에 최적화되는 경향이 있어 데이터에 존재하지 않는 패턴의 경우 가독성이 제대로 측정되지 않을 가능성이 있다.

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5572 on 13 degrees of freedom
Multiple R-squared:  0.929,     Adjusted R-squared:  0.6776
F-statistic: 3.695 on 46 and 13 DF,  p-value: 0.006712
```

그림 6. 전체 메트릭을 사용한 비선형 회귀 분석 결과
Fig. 6. Non-linear regression result using whole metric

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.507 on 17 degrees of freedom
Multiple R-squared:  0.9231,    Adjusted R-squared:  0.7331
F-statistic: 4.858 on 42 and 17 DF,  p-value: 0.0004656
```

그림 7. 최적화된 비선형 회귀 모델 결과
Fig. 7. Result of optimized non-linear regression model

위와 같은 이유로 본 논문에서는 다중 선형 회귀 분석을 통해 만들어진 모델을 선택했다. 그림 8은 다중 선형 회귀 분석을 통해 만들어진 가독성 모델이다. 그 이유는 이를 통해 만들어진 모델이 비선형 회귀 분석을 통해 만들어진 모델에 비해 비교적 데이터에 최적화되지 않기 때문이다.

```
Readability = (-0.020)*LOC + 0.040*numOfComments
            + 0.037*numOfBlankLines + (-0.755)*numOfBitwiseOperators
            + (-0.153)*maxNestedControl + (-0.001)*programVolume
            + (-0.611)*entropy + 7.442
```

그림 8. 가독성 점수 측정 모델
Fig. 8. Model for measuring readability score

V. 결 론

본 논문에서는 소스코드의 가독성 점수를 측정하기 위한 모델을 제안했다. 제안하는 기법은 기존의 가독성 추정 기법들의 이분법적인 분류에서 벗어나 가독성의 점수를 측정하는 방법을 제안했다. 이를 통해, 소스코드의 가독성을 정량적인 점수를 통해 알 수 있다는 장점을 가지게 되었다.

본 논문에서는 설문조사와 회귀 분석을 통해 소스코드 가독성 측정을 위한 모델을 제안했다. 설문조사에는 대학생, 대학원 석·박사 과정 연구원, 회사원 등 45명이 참여했다. 설문조사 참여자들은 각각 20개의 소스코드에 대해 1~5점 척도의 가독성 점수를 채점했다. 이후, 설문조사 결과를 바탕으로 회귀분석을 사용하여 소스코드의 가독성을 추정하였다. 사용된 분석방법은 다중 회귀 분석과 비선형 회귀 분석 방법이었다. 두 방법 모두를 사용해 회귀 분석을 진행하였고 최적화 알고리즘을 적용하였다. 이를 통해 총 4가지의 모델을 만들어 냈고, 이 중 데이터에 덜 최적화된 모델인 다중 회귀 모델을 선택하였다. 해당 모델은 총 7가지의 척도를 사용하고, 설명력은 74.59%를 기록했다.

향후 연구에서는 만들어진 가독성 측정 모델을 적용했을 때 어떠한 변화가 나타날지를 연구할 것이다. 또, 추가적인 설문조사를 통해 모델이 더 일반화될 수 있도록 할 것이다.

References

[1] B. Boehm and V. R. Basili. Software defect reduction top 10 list. IEEE Computer, Vol. 34, No. 1, pp. 135-137, Jan 2001.
DOI: https://doi.org/10.1109/2.962984

[2] Raymond P.L. Buse nad Westley R. Weimer, "A Metric for Software Readability", IEEE Transactions on Software Engineering, Vol. 36, No. 4, pp. 546-558, Aug 2010.
DOI: https://doi.org/10.1109/TSE.2010.2

[3] M. Halstead, "Elements of software science", Elsevier New York, 1977.

[4] D. Posnett, A. Hindle, and P. Devanbu, "A simpler model of software readability", The 8th working conference on Mining software repositories(MSR) vol. 11, pp. 73 - 82, May 2011.
DOI: http://doi.acm.org/10.1145/1985441.1985454

[5] Rodney Ramcharan, "Regressions: Why Are Economists Obessessed with Them?", Finance

Development, vol. 43, No. 1, March 2006.
DOI: http://www.imf.org/external/pubs/ft/fandd/2006/03/basics.htm.

[6] David A. Freedman, “Statistical Models: Theory and Practice”, Cambridge University Press, 2009

[7] Seung-Ho Choi, Ho-Sang Sun, “A Nonlinear Regression Analysis Method for Frame Erasure Concealment in VoIP Network”, The institute of internet, broadcasting and communication, Vol. 9, No. 5, pp. 129-132, Oct 2009.
DOI: http://www.earticle.net/article.aspx?sn=113182.

저자 소개

최 상 철(준회원)

- 2016년 : 전북대학교 학사
- 2016.2~ : 전북대학교 석사재학
- E-Mail: 114477aa@jbnu.ac.kr

김 순 태(정회원)

- 2014~: 전북대학교 소프트웨어공학과 부교수
- 2010: 서강대학교 공학박사
- 2007: 서강대학교 공학석사
- E-Mail: stkim@jbnu.ac.kr

이 정 휴(정회원)

- 1993년~ : 전북대학교 소프트웨어공학과 교수
- 1993년: 전북대학교 공학박사
- 1986년: 전북대학교 공학석사
- E-Mail: jhlee25@jbnu.ac.kr

유 희 경(정회원)

- 1992년~ : 강원대학교 컴퓨터공학과 교수
- 1995년: 동국대학교 이학박사
- 1985년: 동국대학교 이학석사
- E-Mail: khyoo@kangwon.ac.kr