퍼지 및 지능시스템학회 논문지 2001, Vol. 11, No. 2, pp. 141-147

허용적 러프집합에 의한 소프트웨어 분류

The Software Classification by the Tolerance Rough Set

김성애* · 최완규** · 이성주*
SungAe-Kim*, WanKyoo-Choi**, and SungJoo-Lee*

* 조선대학교 전자계산학과
** 광주대학교 컴퓨터전자통신공학부

요 약

소프트웨어의 측정값에 근거하여 소프트웨어 품질에 관한 의사결정을 할 때, 동치관계의 요구조건인 추이적(transitive) 특성이 항상 만족되는 것은 아니다. 순환수(cyclomatic number)가 거의 비슷한 프로그램에서, 하나는 '구조적인' 프로그램 범주에 속하고 또 다른 하나는 '비구조적인' 프로그램 범주에 속한다고 명확히 분류할 수 있는가하는 점이다.
따라서, 본 연구에서는 동치관계보다는 허용적 관계를 만족하는 허용적 러프집합에 근거한 소프트웨어 분류기준을 제시하고자 한다. 분류기준을 생성하기 위한 실험 데이터 집합을 수집하고, 집합 내의 각 원소에 관한 허용적 클래스들을 생성한 후, 각 허용적 클래스들의 중심값을 클러스터링하여 분류기준을 생성한다. 생성된 분류기준을 또 다른 실험 집합에 적용하여 비교 분석한 결과 생성된 분류기준이 타당함을 보여준다.

ABSTRACT

When we decide the software quality based on the software measurement, the transitive property which is a requirement for the equivalence relations is not always satisfied. Given two programs whose cyclomatic numbers are about the same, it is difficult for us to decide clearly that one is 'structural' and the other is 'non-structural'.
Therefore, we propose a rough set based method for software classification that employs a tolerance relation instead of an equivalence relation. We collect the experiment data and generate the tolerance classes for elements in the experiment data, and generate the classification ranges for the decision of the software quality by clustering the means of the tolerance classes. We also apply the generated classification ranges to another experiment data and show their validity by comparing and analyzing them.

Key Words : Software classification, Tolereance rough set

1. 서 론

전체 시스템 비용에서 소프트웨어 비용의 증가로 인하여, 소프트웨어의 복잡도 및 품질 측정에 대한 관심이 증가해왔고, 다수의 소프트웨어 척도들이 제안되었다. 소프트웨어 측정은 프로그래머에게 소프트웨어의 개발 및 유지보수를 위한 중요한 정보 즉, 유지보수성, 이해성, 복잡도 등을 제공한다[5].

소프트웨어 측정에서의 하나의 문제는 측정에 근거하여 의사결정을 할 때 실제로 어떤 기준에 의해 "유지보수가 용이하다"와 같은 의사결정을 할 수 있는가하는 것이다. 이 같은 의사결정은 "유지보수가 용이하다" 또는 "유지보수가 용이하지 않다"라는 언어적 변수를 정의하고 이런 언어적 변수로 분류될 수 있는 측정값의 범위를 결정한 후 임의의 소프트웨어에 대해서 측정값에 근거하여 어떤 부류의 언어적 변수에 속하는 가를 결정하는 것이다. 어떤 프로그램의 순환수가(cyclomatic number)가 20 이상이면 "구조가 필요 이상으로 복잡한 프로그램이다"[10]라고 한 것은 같은 개념을 적용하는 것이다.

여러 연구들[1, 3, 7, 8, 10, 16]이 실험을 통하여 소프트웨어 분류를 위한 기준들을 제시하였으나 이러한 분류 기준들은 분류에 관한 동치관계를 가정하고 있다. 그러나 소프트웨어 측정값에 근거하여 소프트웨어를 분류할 때 동치관계의 요구조건인 추이적 특성이 항상 만족되는 것은 아니다. 순환수(cyclomatic number)가 19와 20인 프로그램 A와 B에서, A는 "구조적인" 프로그램 범주에 속하고 B는 "구조가 필요 이상으로 복잡한" 프로그램 범주에 속한다고 명확히 분류할 수 있는가하는 문제이다.

따라서 본 연구에서는 소프트웨어 측정값에 근거하여 소프트웨어를 분류할 때, 동치관계보다는 추이적 특성이 없는 허용적(tolerance) 또는 유사(similarity) 관계를 만족하는 허용적 러프집합[2, 14, 15, 17]에 기반하며, 이를 데이터의 유사관계 표현에 적용하여 소프트웨어의 분류기준을 산출하고 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 러프집합 이론 및 허용적 러프집합에 대해서 간략히 기술하고, 3장에서는 소프트웨어 객체들의 허용적 클래스들을 얻기 위한 유사성 척도를 정의하고, 분류 기준을 얻기 위한 알고리즘을 제시한

접수일자 : 2000년 5월 31일
완료일자 : 2001년 3월 5일

다. 4장에서는 제안한 알고리즘을 두 개의 실험 집단에 적용하여 비교분석하고, 결론을 제시한다.

2. 관련 연구

2.1 러프집합(Rough set)

러프집합 이론[11, 12, 13]은 어떤 개념을 대상으로 하여 그 개념에 명확하게 속하는 것과 속할 가능성을 가지는 것을 집합을 이용해 나타내고 있다.

정보 객체들이 정보 테이블에서 각 정보를 나타내는 속성들에 의해 표현되며, 주어진 정보에 의해서 서로 구별할 수 있는 경우, 이들 정보 객체들이 구분 불가능한 동치관계에 있다고 정의하고, 이들은 동치관계에 의해 동치항(equivalence class)으로 나눌 수 있다.

정보 객체 x, y, z가 동치관계 R을 만족한다면 이들은 다음 세 가지 성질을 만족한다

1) reflexive(반사적) : xRx
2) symmetric(대칭적) : xRy→yRx (1)
3) transtive(추이적) : xRy∧yRz→xRz

하나의 정보 객체 집단이 동치관계에 의해 분류될 때, 하나의 동치항 내의 원소들의 집합을 원소집합(elementary set)이라고 정의하고, 이 원소집합에 의해 정의되는 집합공간을 근사 공간 A=(U, R)라고 정의한다. 여기에서 U는 정보객체의 전체 집합이고, R은 전체집합(U)에서 정의된 동치관계를 나타낸다. 근사 공간(approximation space)에서 하나의 결정에 대해 정보객체를 분류하는 경우, 동일한 원소집합 내에 있으면서도 서로 다른 현상을 나타내는 경우가 발생할 수 있다. 이렇게 결정상의 불일치를 나타내기 위해서 러프집합에서의 두 가지 근사(approximation)를 정의한다. 하나는 결정에 의해 나타내어지는 집합 X에 항상 포함되는 원소집합으로 정의되는 하한근사 R*(X)이고, 다른 하나는 개념 X와 일치하는 부분이 하나라도 존재하는 모든 원소집합으로 정의되는 상한근사 R*(X)로서, R(x)는 하나의 정보 객체 x가 속한 원소집합인 동치항을 나타낸다.

$$R_*(X)=\{x\in U \mid R(x)\subseteq X\}$$
$$R^*(X)=\{x\in U \mid R(x)\cap X \neq \varnothing\} \quad (2)$$

2.2 허용적 러프집합(Tolerance rough set)

데이터 분류에 관한 문제의 경우에서는 동치관계를 적용하여 데이터간의 유사관계를 나타내는 것은 불합리하다[18]. 데이터 x와 y가 동일한 언어적 범주에 속하고, 데이터 y와 z가 동일한 언어적 범주에 속할 때, 데이터 x와 z도 반드시 동일한 언어적 범주에 속한다고 할 수 없다. 즉, 데이터 분류의 경우, 추이적 성질이 항상 만족되는 것이 아니기 때문이다. 이러한 현상은 두 언어적 범주가 인접하는 경계영역에서 자주 발생한다. 그러므로 데이터 분류의 문제에서는 반사적(relfexive) 성질과 대칭적(symmetric) 성질을 만족하는 허용적 관계에 의해 데이터들간의 유사관계를 나타내야 한다[2].

소프트웨어 메트릭에 근거한 소프트웨어의 분류의 경계영역에서도 반드시 추이적 성질이 만족하는 것은 아니다. 일반적으로 "프로그램의 라인수가 30이하이면 적합하다"는 기준을 근거로 할 때, 29와 30라인으로 구성된 프로그램은 적합하고 31라인으로 이루어진 프로그램은 적합하지 않다라고 명확히 분류할 수 있는가라는 문제가 제기된다. 그러므로 소프트웨어의 분류의 경우에 허용적 관계에 의해서 소프트웨어들 간의 유사관계를 나타내는 것이 필요하다.

모든 속성에 관한 허용적 관계를 τ라 하면, 한 원소 $x\in U$와 모든 속성에 대해 허용적 관계에 있는 원소들의 집합을 $T(x)$라고 하면 이는 다음과 같이 정의된다[2, 18].

$$T(x)=\{y\in U \mid x\ \tau\ y\} \quad (3)$$

임의의 집합 $Y\subseteq U$와 모든 속성에 대해 허용적 관계에 있는 데이터에 관한 하한근사 $\tau_*(Y)$ 및 상한근사 $\tau^*(Y)$는 다음과 같이 정의된다.

$$\tau_*(Y)=\bigcap_{x\in U}\{T(x) \mid T(x)\subseteq Y\}$$
$$\tau^*(Y)=\bigcap_{x\in U}\{T(x) \mid T(x)\cap Y\neq\varnothing\} \quad (4)$$

3. 소프트웨어 분류기준 생성

본 연구에서는 소프트웨어 메트릭에 근거하여 소프트웨어를 분류할 때 k개의 범주로 분류하기 위하여 먼저 각 객체들에 대한 허용적 클래스(tolerance class)들을 정의하고, 각 허용적 클래스들을 클러스터링하여 k개의 그룹으로 분류하기 위한 분류기준을 정의한다.

3.1 소프트웨어 분류를 위한 허용적 클래스(Tolerance class)

일반적으로 허용적 관계는 두 원소간의 유사성 정도를 나타내는 유사성 척도(similarity measure)에 의해 표현된다[6]. 유사성 척도는 적용되는 문제에 따라 여러 가지로 정의될 수 있는데 일반적인 성질은 다음과 같다. 두개의 정보 객체가 갖는 속성값 사이의 유사척도를 s(x, y)라고 할 때, $s(x, y)\geq\alpha$이면 두 정보 객체 x와 y는 허용적 관계에 있다고 한다. 여기서 α는 적용되는 문제에 따라 결정되는 두 데이터간의 허용적 관계의 가부를 판단하는데 사용되는 유사척도의 임계치(threshold)이다[18].

데이터 분류와 같은 문제에서는 흔히 유사척도를 Hamming distance, Euclidian diatance 등과 같은 거리 함수를 사용하여 정의하지만, 메트릭 측정값에 근거하여 소프트웨어를 분류하기 위하여 그들을 적용하는 데에는 문제가 있다.

LOC값에 근거하여 소프트웨어를 분류할 때, 10라인과 20라인으로 구성된 소프트웨어 객체들간에는 분명한 차이를 보이지만 100라인과 110라인으로 구성된 프로그램 객체들간에는 분명한 차이를 보이지 않음을 알 수 있다.

따라서, 본 논문에서는 메트릭 측정값에 근거하여 프로그램 객체들을 분류할 때, 퍼지 소속함수를 이용하여 유사성 척도를 정의한다.

특정 측정 메트릭들의 측정값들의 전체집합 $U=\{x_1, x_2, \cdots, x_n\}$의 임의의 x_i와 다른 원소들인 x_j간의 유사성 척도를 다음과 같이 정의된다

$$s(x_i, x_j) = \mu(x_i, x_j)\wedge\mu(x_j, x_i)$$
$$\mu(x_i, x_j)=\frac{1}{1+(x_j-x_i)^2(x_{\max}+1-x_i)/x_{\max}} \quad (5)$$
$$x_i\in U,\ x_j\in U,\ j\neq i,\ j=1, 2, \cdots, n$$
$x_{\max}$: 미리 정의된 최대 값

LOC의 측정값의 최대 값이 100이라면(즉, x_{max} = 100), 측정값이 각각 10과 90인 객체와 다른 객체들 간의 유사성의 척도는 그림 1과 같이 나타난다.

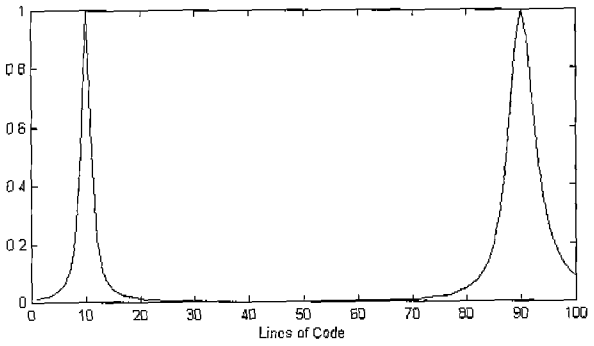

그림 1. LOC에 대한 $\mu(10)$과 $\mu(90)$의 소속함수
Fig. 1. The membership function for $\mu(10)$ and $\mu(90)$ of LOC

그림 1에서 LOC가 10인 객체와 유사한 객체들의 범위와 LOC가 90인 객체와 유사한 객체들의 범위가 다르며, LOC 값이 증가할수록 포함되는 범위가 증가한다는 것을 알 수 있다.

식 (5)의 유사성 척도에 근거하여 임의의 한 원소 $x_i \in U$의 허용적 클래스(tolerance class) $\tau(x_i)$를 식(6)과 같이 정의하며, 유사성 척도 $s(x_i, x_j)$는 반사성과 대칭성의 공리를 만족한다.

$$\tau(x_i) = \{x_j \mid s(x_i, x_j) > \alpha,\ x_i, x_j \in U,\ j \neq i,\ j = 1, 2, \cdots, n\} \cup \{x_i\} \quad (6)$$

3.2 Clustering

n개의 객체들에 대한 허용적 클래스들은 n개가 생성되는데, n개의 허용적 클래스들을 k개의 ($k \leq n$) 집합으로 묶어서 k개의 분류기준을 생성하기 위해서 n개의 허용적 클래스들에 클러스터링 알고리즘을 적용한다.

클러스터링은 일련의 유사성이 있는 대상 자료들을 하나의 그룹으로 구성될 수 있도록 다른 그룹들과 분리하는데, 이를 위하여 식(7)과 같이 각 허용적 클래스들의 중심값을 구한 후, 각 중심값들을 k-means 알고리즘을 사용하여 클러스터링한다.

$$C = \{c_1, c_2, \cdots, c_n\} \quad (7)$$

여기서, $c_i = \frac{\sum_{j=0}^{m} x_j}{m},\ x_j \in \tau(x_i)$

k-means 알고리즘은 MacQueen에 의해 제안된 알고리즘으로, 패턴을 k개의 클러스터로 나눈 후 클러스터에 포함되어 있는 패턴들의 평균으로 클러스터의 중심값을 계산하고 이 중심값과 각 패턴과의 거리를 계산한 후 거리가 가장 가까운 클러스터에 패턴을 포함시키는 방법으로 그 조건은 다음과 같다[21].

$$c_i \in m_j,\ ||c_i - z_j||^2 < ||c_i - z_k||^2 \quad (8)$$

$1 \leq i \leq n,\ 1 \leq k \leq m,\ j \neq k$
n: 객체들의 수, m: 클러스터의 수
z: 클러스터의 중심값

그리고 이 계산의 각 클러스터의 중심 값이 더 이상 변하지 않을 때까지 반복한다. 초기 k개의 클러스터의 중심값을 설정하는 방법에는 주어진 패턴에서 처음 k개의 패턴을 추출하여 중심값으로 하는 방법과 임의의 k개를 추출하여 중심값으로 하는 방법이 있는데 본 연구에서는 주어진 데이터의 최대값을 k등분한 값을 추출하여 클러스터의 초기 중심값으로 설정하였다. 예를 들어서, 최대값이 100이 고, k가 3이면, 초기 중심값은 100/3, 200/3, 300/3으로 설정하였다.

3.3 분류기준 생성 알고리즘

n개의 허용적 클래스들로부터 k개의 분류기준을 생성하는 과정은 그림 2와 같다.

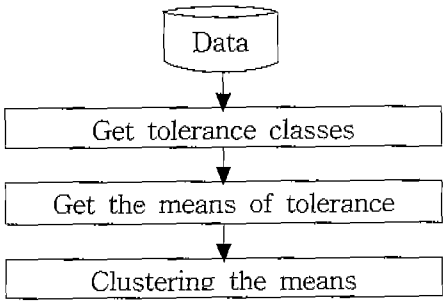

그림 2. 분류기준 생성 과정
Fig. 2. The generation process of the classification range

위의 과정을 통해 생성된 k개의 분류 기준들을 적용하여 임의의 소프트웨어 객체는 소프트웨어에 대한 언어적 분할인 G_i (여기서 i = 1, 2, ..., k)로 분류된다.

4. 실험 및 결과

실험을 위해서 본 연구는 Linux 커널 소스와 Ansi C 런타임 라이브러리에서 추출된 C언어로 작성된 18,404개의 모듈들을 대상으로 하였다. 실험 대상의 모듈들의 전체 라인수는 533,165이였다. 모집단의 크기가 20,000일 때, 95% 신뢰도 수준에서 허용오차 ±1%내에서의 적정 표본의 개수가 8,213개이므로[9], SPSS를 이용하여 무작위로 8,213개를 선택하여 분류기준 생성을 위한 실험 데이터 집합 T_A를 구성하였고, 실험 결과를 나머지에 실험 데이터 집합 T_B에 적용하여 T_A에서의 결과와 T_B에서의 결과를 비교하였다.

기능 중심으로 구성된 모듈의 품질 측정은 규모(size)와 난이도(difficulty)의 메트릭들을 수집한 정적 분석기에 의해 쉽게 측정될 수 있으며[8, 22], 기존의 소프트웨어 모듈의 품질 평가 방법들인 McCabe의 Cyclomatic Number, Halstead의 Software Science, Lines Of Code 등의 도입이 가능하다[22]. 따라서, 본 모델에서는 현업에서의 많은 연구와 실험을 통해서 그 타당성이 검증된 메트릭들인 LOC(lines of code), McCabe의 Cyclomatic Number, Halstead의 Volume, Difficulty, Effort에 관한 분류기준을 생성하였다.

4.1 정규모집단에 대한 검토

표 1은 8,213개의 모듈들로 구성된 T_A에 관한 측정값들에 대해서 SPSS를 이용하여 추출된 기술통계량(descriptive statics)을 보여주며, 그림 3은 LOC에 관한 누적 분포 그래프를 보여준다.

표 1. 실험집합 T_A의 기술 통계
Table 1. Descriptive statistics of T_A

	Mean	Std. Deviation	Skewness	Kurtosis
LOC	28.8034	20.5029	1.276	1.100
CYC	5.0097	3.8107	2.252	7.039
VOL	580.2505	562.9894	1.792	3.734
DIF	20.9839	14.8588	1.389	2.176
EFF	19119.9740	31019.0731	2.990	11.180

※ LOC: Lines of Code, CYC: Cyclomatic number, VOL: Volume, DIF: Difficulty, EFF: Effort

NTT에서의 4년에 걸쳐 실시된 Isoda의 실험[4]에서 작성된 100만 라인에 이르는 프로그램 모듈 중에서 48%정도가 50라인 이하의 모듈들로 구성되어있고, 모듈크기의 누적 분포가 그림 4와 같은 S자형을 이루고 있음을 보여주고 있다.

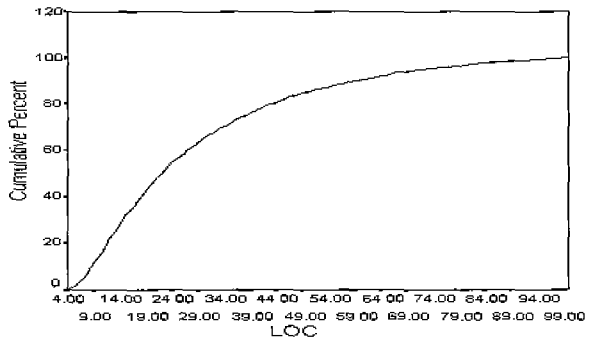

그림 3. LOC의 누적분포
Fig. 3. Cumulative distribution of LOC

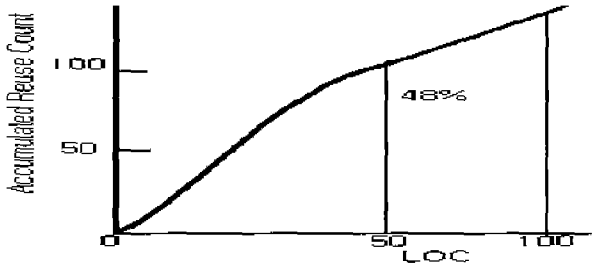

그림 4. LOC에 관한 누적 재사용 분포
Fig.4. Accumulated reuse distribution of LOC

그림 3의 TA에서의 LOC에 관한 누적 분포가 그림 4의 Isoda의 실험에서의 분포와 비슷하고, 또한 표 1에서 알 수 있듯이 각 척도들의 왜도(skewness)와 첨도(kurtosis)가 0에 근사한 작은 값들을 가진다는 것과 중심점의 극한 정리[19, 20]에 의해 실험에 사용된 데이터가 근사적으로 정규 분포가 된다는 것을 알 수 있다.

4.2 분류기준 결정

산업 현장에서의 실험적인 연구들[8]은 코드의 라인수(LOC)는 100이하, 복잡도(Cyclomatic Number)는 50이하, 난이도(Difficulty)는 100이하, 프로그램 노력도(Effort)는 300,000이하, 볼륨(Volume)은 10,000이하를 수용 가능한 값들의 범위로 제시하였다.

따라서 본 논문에서는 기존의 연구들의 수용하여 각 측정 메트릭들의 최대 값을 표 2와 같이 설정하였으며, 식 (6)에서의 α 값을 0.1로 설정하였다.

표 2. 메트릭들의 최대 값
Table 2. The max values of metrics

Metric	LOC	CYC	VOL	DIF	EFF
Max	100	50	10,000	100	300,000

식(5)와 (6)을 이용하여 T_A의 각 원소들에 대한 허용적 클래스를 구하고, 클러스터링 과정을 통해서 실험 데이터 집합 T_A에로부터 산출된 기능중심 모듈을 세 가지 범주(Low, Medium, High)로 분류하기 위한 소프트웨어 분류 기준은 표 3과 같다.

표 3. 산출된 분류 기준
Table 3. The generated classification ranges

분류 / 메트릭	Low	Medium	High
LOC	$0<\tilde{}\leq 29$	$24<\tilde{}\leq 57$	$50<\tilde{}\leq 100$
CYC	$0<\tilde{}\leq 12$	$7<\tilde{}\leq 19$	$14<\tilde{}\leq 50$
VOL	$0<\tilde{}\leq 568$	$564<\tilde{}\leq 1405$	$1403<\tilde{}\leq 10000$
DIF	$0<\tilde{}\leq 22$	$15<\tilde{}\leq 43$	$35<\tilde{}\leq 100$
EFF	$0<\tilde{}\leq 29734$	$29636<\tilde{}\leq 91667$	$91456<\tilde{}\leq 300000$

4.3 분류기준의 적용

표 3의 분류기준을 T_A의 8,213개의 모듈과 T_B의 10,191개의 모듈에 적용하고, LOC에 관한 분류 기준을 적용한 결과는 표 4와 표 5와 같다.

표 4. LOC에 의한 T_A에서의 분류
Table 4. The classification in T_A by LOC

분류	개수
Low	5222
Medium	3021
High	1280

표 2의 LOC에 관한 분류기준에 의해 집합 T_A와 T_B의 원소들을 분류할 때 두 집단의 특성을 비교하는데, 두 모평균의 차에 관한 추론을 통해서 분류된 데이터들의 평균이 통계적으로 유의한 차이가 있는가를 검토한다.

표 5. LOC에 의한 TB에서의 분류
Table 5. The classification in TB by LOC

분류	개수
Low	6435
Medium	3763
High	1641

μA, μB는 각각 T_A에서의 분류집단과 T_B에서 분류집단(Low, Medium, High)의 모평균이고, σ_A^2, σ_B^2는 각각 모분산이라 할 때, 두집단의 모평균의 차에 관한 추론 문제에서 일반적으로 σ_A^2, σ_B^2를 미리 알고 있는 경우보다는 모르는 경우가 많으므로 두 모집단이 정규분포를 따른다는 가정 이외에 두 모집단은 미지의 동일한 분산을 갖는다는 조건, 즉 $\sigma_A^2 = \sigma_B^2 = \sigma 2$이라는 가정이 필요하다[19].

따라서, 먼저 등분산 가정(H0 : $\sigma_A^2 = \sigma_B^2$)을 검정할 필요가 있으므로, T_A와 T_B에서 "Low", "Medium", "High"로 분류된 원소들의 유의수준 5% 수준에서 등분산 가정의 검정을 위해 Excel을 이용한 F-검정 통계량은 표 6~8과 같다. 각각의 경우 P(F<=f)>0.05이므로 유의하지 않은 것으로 나타났으므로 $\sigma_A^2 = \sigma_B^2$ 가정을 기각할 수 없다.

표 6. 분류집합 Low에 대한 F-검정 통계량
Table 6. F-statistics for classification Low

	Low of T_A	Low of T_B
평균	16.16357	16.22941
분산	42.82152	42.45972
관측수	5221	6434
자유도	5220	6433
F비	1.008521	
P(F<=f) 단측 검정	0.373342	
F기각치: 단측 검정	1.04423	

표 7. 분류집합 Medium에 대한 F-검정 통계량
Table 7. F-statistics for classification Medium

	Medium of T_A	Medium of T_B
평균	36.59011	36.54035
분산	88.33883	89.5411
관측수	2974	3705
자유도	2973	3704
F비	0.986573	
P(F<=f) 단측 검정	0.349457	
F기각치: 단측 검정	0.944224	

위의 결과에서 등분산에 대한 가정을 기각할 수 없으므로 등분산의 가정(H0 : $\sigma_A^2 = \sigma_B^2$)아래서 분류된 데이터들의 평균이 통계적으로 유의한 차이가 있는가(즉, H0 : $\mu_A = \mu_B$)를 검정하기 위한 유의수준 5% 수준에서 μ_A-μ_B에 대한 가설검정을 위한 T-검정 통계량은 표 9~11과 같다.

표 8. 분류집합 High에 대한 F-검정 통계량
Table 8. F-statistics for classification High

	High of T_A	High of T_B
평균	67.86865	67.9939
분산	185.7981	189.8011
관측수	1279	1640
자유도	1278	1639
F비	0.97891	
P(F<=f) 단측 검정	0.344035	
F기각치: 단측 검정	0.916579	

표 9. 분류집합 Low에 대한 T-검정 통계량
Table 9. T-statistics for classification Low

	Low of T_A	Low of T_B
평균	16.16357	16.22941
분산	42.82152	42.45972
관측수	5221	6434
공동(Pooled) 분산	42.62179	
가설 평균차	0	
자유도	11653	
t 통계량	-0.54139	
P(T<=t) 단측 검정	0.294125	
t 기각치 단측 검정	1.644985	
P(T<=t) 양측 검정	0.58825	
t 기각치 양측 검정	1.960166	

표 9~11에서 "t 통계량 < t 기각치 양측 검정"을 만족하고, "P(T<=t) 양측 검정>0.05"이므로 유의하지 않으므로 $\mu_A = \mu_B$ 가정을 기각할 수 없다.

따라서, 표 3의 분류기준을 통해서 두 집단을 분류할 때, 두 집단간에 차이가 없다는 결론을 내릴 수 있다. 즉, 표 3의 분류 기준을 T_A와 T_B에 적용하였을 때 동일한 특성을 보이므로, 표 3의 분류기준을 소프트웨어 분류를 위한 기준으로 사용할 수 있다.

표 10. 분류집합 Medium에 대한 F-검정 통계량
Table 10. T-statistics for classification Medium

	Medium of T_A	Medium of T_B
평균	36.59011	36.54035
분산	88.33883	89.5411
관측수	2974	3705
공동(Pooled) 분산	89.00578	
가설 평균차	0	
자유도	6677	
t 통계량	0.214245	
P(T<=t) 단측 검정	0.415181	
t 기각치 단측 검정	1.645083	
P(T<=t) 양측 검정	0.830363	
t 기각치 양측 검정	1.96032	

표 11. 분류집합 High에 대한 F-검정 통계량
Table 11. T-statistics for classification High

	High of T_A	High of T_B
평균	67.86865	67.9939
분산	185.7981	189.8011
관측수	1279	1640
공동(Pooled) 분산	188.0473	
가설 평균차	0	
자유도	2917	
t 통계량	-0.24485	
P(T<=t) 단측 검정	0.403295	
t 기각치 단측 검정	1.645376	
P(T<=t) 양측 검정	0.806589	
t 기각치 양측 검정	1.96078	

5. 결 론

소프트웨어 측정값에 근거하여 소프트웨어를 분류할 때 동치관계의 요구조건인 반사적, 대칭적, 추이적 특성이 항상 만족되는 것이 아니므로, 본 연구에서는 동치관계보다는 추이적 특성이 없는 허용적 관계를 만족하는 허용적 러프집합에 근거한 프로그램 분류기준을 제시하였다.

프로그램의 분류기준 제시를 위하여, 각 객체들에 관한 허용적 클래스들을 산출하고, 다음으로 허용적 클래스들의 중심 값을 클러스터링하여 프로그램 분류 기준을 설정하였다.

실험에서 사용된 데이터들의 기술통계량을 보여주고, 알고리즘을 하나의 실험 집단에 적용하여 소프트웨어 분류기준을 산출하였다. 산출된 기준을 또 다른 실험 집단에 적용하여 산출된 분류기준을 통해서 두 집단을 분류할 때 두 집단간에 차이가 없다는 것을 보여주었다.

본 연구는 지금까지 단순 실험을 통해서 제시하였던 소프트웨어 메트릭에 근거한 소프트웨어의 평가기준을 개선하여 러프집합 및 퍼지집합 이론을 이용하여 인간의 직관에 더욱 유사한 평가기준을 산출할 수 있음을 보여주었다.

본 연구에서 제시된 결과를 현장에 적용하여 현장에서 제시된 결과와 본 연구에서 제시한 결과를 비교 분석하는 연구가 요구된다.

참 고 문 헌

[1] Caldiera, G. and V.R. Basili, "Identifying and Qualifying Reusable Software Components", *IEEE Computer*, pp. 61-70, Feb. 1991.

[2] K.Funakoshi and T.B.Ho, "Information retrieval by rough tolerance relation", *The 4th international Workshop on rough sets, Fuzzy sets, and Machine Discovery*, Tokyo, Nov. 1996.

[3] Horst Zuse, *Software Complexity-Measures and Methods*, New York:Walter de Gruyter, pp. 25-37, 1991.

[4] Sadahiro Isoda, "Experience report on software reuse project: its structure, activities, and statistical results", *Proceedings of the 14th international conference on Software engineering*, pp. 320-326, 1992.

[5] Karl J. Ottensteion, Linda M. Ottensteion, "The program dependence graph in a software development environment", *ACM SIGPLAN Notices*, vol. 19, no. 5, pp. 177-184, May 1984.

[6] M.Kretowski and J. Stepniuk, "Selection of objects and attributes a tolerance rough set approach", *ICS Research Reports*, 1994.

[7] Lewis John, Henry Salie, "A Methodology for Integrating Maintainability Using Software Metrics", *Proceedings: Conference on Software Maintenance*, Miami, Florida, IEEE, pp. 32-39, Oct. 1989.

[8] Lowell J. Arthur, *Measuring Programmer Productivity and Software Quality*, New York:John Wiley & Sons, pp. 138-142, 1985.

[9] D.J.Luck, H.G. Wales, D.H.Taylor, *Marketing Research*, N.J.:Prentice-Hall, pp. 611-612, 1970.

[10] T.McCabe, "A Complexity Measure", *IEEE Trans.*SE., SE-2, pp. 308-320, 1976.

[11] Pawlak Z., "Rough sets", *International Journal of Computer and Information Sciences*, 11, pp. 341-356, 1982.

[12] Pawlak Z., "Rough Logic", *Bulletin of the Polish Academy of Sciences, Technical Science* 35, pp. 253-258, 1987.

[13] Pawlak Z., *Rough Sets-Theoretical Aspects of Reasoning about Data*, London: Kluwer Academic Publishers, 1991.

[14] Slowwinski R. and Vanderpooten D. "Similarity relations as a basic for rough approximations, *ICS Research Reports*, 1994.

[15] Slowwinski R. and Vanderpooten D. "A Generalized definition of rough approximations", *ICS Research Reports*, 1996.

[16] Szentes J., Gras j., "Some Practical Views of Software-Complexity metrics and a Universal Measurement Tool", *First Australian Software Engineering Conference*, Canberra, pp. 14-16, May 1986.

[17] Y.Y.Yao, T.Y.Lin, "Generalization of Rough Sets using Modal logics", *http://www.mathcs.sjsu.edu /irss/WSC/ rough_modal.ps*, 1999.

[18] 김대진, 김철현, "허용적 러프집합을 이용한 필기체 숫자 인식", *한국퍼지및지능시스템학회논문지*, vol. 9, no.1, pp. 113-123, 1999.

[19] 김우철외, *현대통계학*, 영지문화사, 1989.

[20] 김은정, 박양규, SPSS *통계분석*, 21세기사, 2000.

[21] 조형기, 민준형, 최종욱, "클러스터링을 이용한 차종 인식 모형", *한국정보처리학회 논문지*, vol. 3, no. 2, pp. 369-380, 1996.

[22] 이경환 외, *소프트웨어 재이용을 위한 연구*, 연구보고서, 과학기술처, pp. 102~103, 1989.

저 자 소 개

김 성 애 (Sung-Ae Kim)
E-mail : sakim@cafe.chosun.ac.kr
1987 년 : 조선대학교 전자계산학과 (학사)
1990 년 : 조선대학교 전자계산학과 (이학석사)
1998년~현재 : 조선대학교 전자계산학과 박사과정

관심분야 : 소프트웨어공학, 품질평가, 객체지향시스템

최 완 규 (Wan-Kyoo Choi)
E-mail : wkchoi@kwangju.ac.kr
1988 년 : 서울대학교 종교학과(학사)
1992 년~1993 년 : (주)공성통신
1993 년~1995 년 : 한양시스템
1997 년 : 조선대학교 전자계산학과 (이학석사)
2000 년 : 조선대학교 전자계산학과 (이학박사)
2000 년~현재 : 광주대학교 컴퓨터전자통신공학부 전임강사

관심분야 : 객체지향 시스템, 러프집합, 퍼지제어, 전자상거래

이 성 주 (Sung-Joo Lee)
E-mail : sjlee@mail.chosun.ac.kr
1970 년 : 한남대학교 물리학과(학사)
1992 년 : 광운대학교 전자계산학과 (이학석사)
1998 년 2월 : 대구 가톨릭대학교 (이학박사)
1988 년~1990 년 : 조선대학교 전자계산소 소장
1995 년~1997 년 : 조선대학교 정보과학대학장
1981 년~현재 : 조선대학교 컴퓨터공학부 교수

관심분야 : 소프트웨어 공학, 프로그래밍 언어, 객체지향 시스템, 러프집합, 전자상거래