한국퍼지 및 지능 시스템학회 논문지 1998, Vol. 8, No. 2. (1998)96~105

러프집합을 이용한 재사용성 결정 알고리즘 생성 시스템
Reusability Decision Generation system using Rough Set

최완규 · 이성주
Wan-Kyoo Choi and Sung-Joo Lee
조선대학교 전자계산학과

요 약

소프트웨어 재사용 분야에 있어서 우선적으로 연구되어야할 부분은 소프트웨어 부품의 품질 보증에 관한 연구이다. 그러나 기존의 연구들은 사용자 요구의 복잡, 다양화와 소프트웨어 복잡도 증가등과 같은 변화하는 환경에 능동적으로 대처하지 못한다.

따라서, 본 논문에서는 재사용되고 있는 부품들, 정량적인 척도들과 분류 기준들을 이용하여 변화하는 환경에 능동적으로 대처할 수 있는 적응성이 있는 재사용성 결정 알고리즘 생성 모델을 제안한다. 이 모델은 적응성 있는 재사용 결정 알고리즘을 찾기 위해서 데이터의 숨겨진 패턴들을 발견하는 효율적인 알고리즘을 제공하는 러프 집합 이론을 이용한다.

ABSTRACT

First of all, measuring the reusability of softwares is researched in the fields of software reuse researches. But the existing researches didn't cope with such changeable environments as the complexity and variety of users' requirement and the increase of software complexity etc..

Therefore, in this paper, we propose the adaptable reusability decision model, which can actively cope with the changeable environments using components being reused, the quantitative metrics and classification criteria. The proposed model employs rough set to find the adaptable reusability decision algorithm. Rough set provides efficient algorithms for finding hidden patterns in data.

1. 서 론

컴퓨터가 일반 사용자에게 널리 보급됨에 따라, 80년대 이후에는 소프트웨어 개발 환경을 중시하게 되었고, 또한 사용자의 요구는 복잡하고 다양화되고 있는 실정이다. 사용자의 요구 증가는 소프트웨어의 규모나 비용을 더욱 증가시키고 있으며, 이에 따른 유지·보수에 소요되는 인력 및 비용 또한 상대적으로 증가하고 있는 실정이다[14].

따라서 신뢰성이 높은 고품질의 소프트웨어를 생산하기 위해서는 기존의 개발 방법론으로는 한계가 있기 때문에 이를 해결하기 위한 하나의 방법으로 하드웨어 부품 생산 개념을 도입하여 소프트웨어에도 하드웨어 부품처럼 생산, 합성하는 소프트웨어 재사용에 관한 연구가 활발히 진행 중이다.

소프트웨어 재사용은 신뢰성이 확보된 기존의 소프트웨어를 새로 작성하는 소프트웨어의 일부분으로 다시 사용하는 방법이다[6]. 그러므로 소프트웨어 재사용 분야에 있어서 우선적으로 연구되어야할 부분은 소프트웨어 부품의 품질 보증에 관한 연구이다.

기존의 연구들은 모듈성(modularity)을 정량적으로 측정하여 소프트웨어 부품의 품질을 보증하려는 방향으로 진행되어왔다. 그러나 그들은 사용자 요구의 복잡, 다양화와 이로 인한 소프트웨어 복잡도등 변화하는 환경에 능동적으로 대처하지 못한다.

따라서, 본 연구에서는 이런 문제점을 해결하기 위해서 실제로 재사용되고 있는 부품들과 산업 현장에서 그 타당성을 인정받은 정량적인 척도들과

※ 본 논문은 1997년도 조선대학교 학술연구비에 의해 연구되었음

분류 기준을 이용하여 새로운 부품의 삽입과 삭제, 분류 기준의 변경 등의 환경변화에 능동적으로 대처할 수 있는 적응성이 있는 재사용성 결정 모델을 제안한다. 이 모델은 적응성 있는 재사용 결정 알고리즘의 생성을 위하여 러프 집합이론을 이용한다.

본 논문의 구성은 다음과 같다. 2장 및 3장에서는 소프트웨어 재사용 및 rough logic에 대해 간략하게 살펴본다. 4장에서는 본 연구에서 제안한 재사용성 결정 모델의 구조 및 구성 요소들을 제시한다. 5장에서는 본 모델을 이용하여 최소 재사용성 결정 알고리즘을 생성하는 과정을 구현하고, 환경 변화에 능동적으로 대처하여 재사용성 결정 알고리즘을 생성하는 과정 등을 제시한다. 마지막으로 결론에서는 본 연구를 통하여 얻은 결론에 대해서 언급하고 향후의 연구 과제를 제시하였다.

2. 소프트웨어 재사용

소프트웨어 재사용 방법을 이용한 소프트웨어 개발이 생산성 향상, 품질 개선 및 유지·보수 비용 절감 등 여러 가지 측면에서 효율적이지만, 다음과 같은 이유 때문에 소프트웨어 개발 과정에 널리 이용되지 않고 있다[5] [14].

첫째, 소프트웨어 부품을 위한 형식 명세서(formal specification)의 부족이다. 둘째, 소프트웨어 부품의 정확성을 증명하기 어렵다는 점이다. 셋째, 재사용 가능한 소프트웨어 부품들의 성능에 관한 문제이다. 넷째, 경제적인 측면이다. 다섯째, 조직적인 측면이다. 여섯째, 소프트웨어 재사용에 대한 부정적 심리적인 효과이다. 일반적으로 프로그래머들이 가지고 있는 "NIH(Not Invented Here)"증후군, 즉 다른 사람이 작성한 코드에 대한 불신과 코드를 이해해야 하는 불편함에 대한 선입견이다.

2.1 재사용성 측정

이와 같은 문제를 해결하기 위한 방안이 소프트웨어 재사용성의 측정이다. 소프트웨어 재사용성 측정의 목적은 부품의 품질 보증을 통하여 사용자들의 불신을 해소시키고 재사용율을 제고하여 소프트웨어의 생산성과 신뢰성을 향상시키려는 것이다.

재사용 소프트웨어가 갖추어야할 일반적인 기준은 일반성(generality), 모듈성(modularity), 기계 독립성(hardware independence), 자기 문서화(self-description)등 이다[1]. 재사용성 측정은 위에 제시된 네 가지 기준 중 주로 모듈성을 정량적으로 측정하려는 방향으로 진행되어 왔다[2] [17]. 이 연구들은 먼저 평가 기준들을 설정한 후 정량적인 측정 방법을 도입하여 평가 기준들을 측정하여 재사용성을 평가하며, 대표적인 연구로는 CARE시스템을[2] 들 수 있다.

2.2 기존 측정 방법의 문제점

기존의 평가 시스템들은 일반적으로 설정된 재사용 품질 기준을 표준으로 하였으므로, 특정 소프트웨어를 개발하기 위하여 요구되는 부품들이 설정된 품질기준을 만족하지 못하여 재사용이 불가능한 부품으로 분류되는 경우가 있다. 산업 현장에서의 많은 연구들은 McCabe의 순환수(cyclomatic number)가 20 이상이면 프로그램이 매우 복잡하거나 구조가 필요 이상으로 복잡한 프로그램이므로 프로그램을 분할하는 것이 바람직하다고 주장한다[8]. 그러나, 순환수 값이 20 이상인 프로그램이 구조화가 잘 됐을 경우, 유지 보수가 용이하다는 양면성이 존재하기도 한다[1].

소프트웨어 위기의 근본적인 원인 중 하나는 소프트웨어 복잡성의 증가로, 이는 소프트웨어 기능에 대한 사용자의 요구가 복잡하고, 다양화되는데 있다. 따라서 소프트웨어는 이런 사용자의 요구를 충족시키기 위해서 점차 기능과 구조가 복잡해지고 코드의 크기도 증가하고 있다[14]. 소프트웨 어의 크기, 알고리즘과 구조의 복잡도 증가는 기 존의 재사용 평가시스템들이 이미 설정한 기준의 변경을 요구하게 된다. 그러나 기준의 변경은 재 사용 관리 시스템 전체의 변경을 요구할 수 있으 므로, 환경변화에 능동적으로 대처하지 못하게 하 는 이유가 된다.

더욱 중요한 것은 제시된 평가 기준들과 정량적인 척도들을 모두 만족시키는 소프트웨어를 제작할 수 없으므로, 제시된 평가 기준과 정량적인 척도 중 사용자가 어느 것을 더욱 중요시하는가를 결정하는 것이다[15].

3. 러프 집합

불완전한 지식을 다루고 조작하는 문제에 대한

많은 접근 방법 중 가장 널리 알려진 방법론은 Zadeh에 의해 제안된 Fuzzy 집합 이론이다.

이 문제에 대한 또 다른 시도로서, 1982년 Pawlak에 의해 제안된 Rough set 개념은 모호성(vagueness)과 불확실성(uncertainty)에 대한 새로운 수학적 접근 방법이다[9].

객체들의 분류에 관한 지식은 정보 시스템(information system)으로 표현될 수 있으며, 정보 시스템의 각 항목은 객체에 관한 내용(statement)으로 보여질 수 있다. Rough logic[10]은 러프 집합 이론에 근거하여 분류(class)로서 간주되는 지식을 추론할 수 있게 한다.

3.1 알고리즘과 알고리즘의 감축

러프 논리에서의 의사결정 규칙 $\Phi \rightarrow \Psi$는 선행자(predecessor)와 후행자(successor)를 이용한 함축성 있는 표현식이다. 의사결정 규칙들의 유한 집합을 기본 의사결정 알고리즘(basic decision algorithm)이라 하며, 알고리즘에 포함된 모든 의사결정 규칙들이 PQ-기본규칙들이면 PQ-알고리즘, 즉 알고리즘 (P, Q)라 한다.

수용가능하고 일치하는 알고리즘이 있을 때, 러프 논리에서 알고리즘의 감축은 전체 시스템에서 불필요한 속성들을 제거하며, reduct와 core의 개념을 논리적 용어(logical term)들의 정의를 통하여 다음과 같이 수행한다.

알고리즘 (P, Q)에서, a∈P일 때, ((P,-{a}), Q)가 불일치이면 속성 a는 알고리즘 (P, Q)에서 필요이고, 모든 속성(a∈P)이 알고리즘 (P, Q)에서 필요이면, 알고리즘 (P, Q)는 독립이다. 알고리즘 (P, Q)에서 모든 필요인 속성들의 집합이 알고리즘 (P, Q)의 core이다. R⊆P인 알고리즘 (P, Q)가 독립이고 일치이면 R(R⊆P)은 알고리즘 (P, Q)에서 P의 reduct이다. R이 알고리즘 (P, Q)에서 P의 reduct이면, 알고리즘 (R, Q)는 알고리즘 (P, Q)의 reduct이다.

3.2 결정 규칙의 감축

의사결정 알고리즘은 규칙들의 집합에서 조건들의 감축을 통해서 단순화되며, 규칙들의 감축은 각 규칙들의 reduct와 core의 정의를 통해서 다음과 같이 수행한다.

식 Φ가 P-기본식이고 Q⊆P이면, Φ/Q는 Φ에서 a∈P-Q인 모든 원자식 (a, v_a)들을 제거함으로써 식 Φ로부터 획득되는 Q-기본식이다. 식 $\Phi \rightarrow \Psi$가 PQ-규칙이고, a∈P 일 때, $|\Phi \rightarrow \Psi| \subseteq |\Phi / (P-\{a\}) \rightarrow \Psi|$이면 속성 a는 규칙 $\Phi \rightarrow \Psi$에서 필요 이고, $\Phi \rightarrow \Psi$에서 필요인 모든 속성들의 집합이 $\Phi \rightarrow \Psi$의 core이다. 모든 속성들 a∈P가 $\Phi \rightarrow \Psi$에서 필요이면, $\Phi \rightarrow \Psi$는 독립이고, $\models_s \Phi \rightarrow \Psi$가 $\models_s \Phi/R \rightarrow \Psi$을 의미하면 R은 PQ-규칙 $\Phi \rightarrow \Psi$의 reduct이다.

3.3 결정 규칙의 최소화

알고리즘은 동일한 의사결정류들과 관련된 불필요한 의사결정 규칙들의 제거함으로써 최소화된다. 알고리즘의 최소화는 동일한 의사결정류를 갖는 의사결정 규칙의 집합에서의 reduct의 정의를 통하여 다음과 같이 수행한다.

정보 시스템 S=(U, A)에서, 특정 후행자 Ψ를 갖는 기본 알고리즘 A 에서 모든 기본 규칙들의 집합은 $A\ \Psi$로 표기하고, $IP\Psi$는 $A\ \Psi$에 속하는 의사결정 규칙들의 모든 선행자들의 집합이다.

$\vee IP\Psi$이 $IP\Psi$에서의 모든 식들의 논리합을 나타낼 때, $\models_s \Phi \equiv \vee IP\Psi$ $\{IP\Psi-\{\Phi\}\}$이면, A 에서의 기 본 의사결정 규칙 $\Phi \rightarrow \Psi$는 A에서 필요이다. $A\ \Psi$에서의 모든 의사결정 규칙이 필요이면 규칙들의 집합 $A\ \Psi$은 독립이다. $A\ \Psi$의 의사결정 규칙들의 부분집합 A'에서 모든 의사결정 규칙들이 독립이 고, $\models_s \Phi \equiv \vee IP\Psi \equiv \vee IP'\Psi$이면, A'은 $A\ \Psi$의 reduct이다.

러프 집합의 모든 개념들은 러프 논리로 표현할 수 있으며, 러프 논리는 러프 집합 이론에 근거하여 분류로 간주되는 지식을 논리적으로 추론할 수 있게 한다.

4. 재사용성 결정 모델

4.1 재사용성 결정 모델의 구조

모듈 재사용 시스템의 일반적인 모델은 〈그림 1〉과 같다[2].

본 연구에서 제안한 재사용성 결정 모델은 재사용 부품 라이브러리 관리 시스템의 서브 시스템으로서 부품의 등록, 삭제, 갱신 등의 작업이 이루어질 때 능동적으로 재사용 결정 알고리즘을 생성할 수 있다.

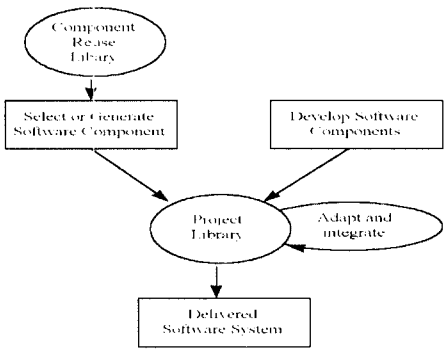

그림 1 재사용 부품 소프트웨어 개발 모델

본 논문에서 제안한 재사용 결정 모델은 측정 메트릭스 선택(Metric Selection), 결정 기준 설정(Criteria Selection), 결정 알고리즘 생성(Decision Algorithm Generation), 재사용성 결정(Reuability Decision)으로 구성되며 〈그림 2〉와 같다.

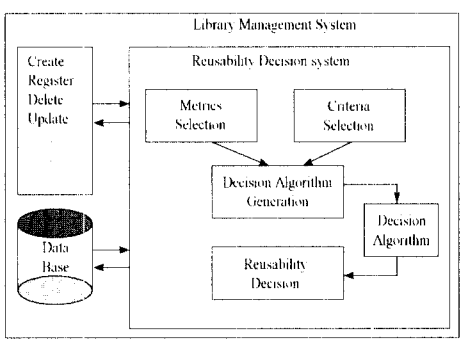

그림 2 라이브러리 부품 관리 시스템

4.2 측정 메트릭스

재사용 가능 부품으로 분류되는 부품들은 정확성, 효율성, 유지 보수성, 신뢰성 등이 있어야하고, 부품의 품질이 높아야한다. 높은 품질의 부품은 크기가 작고, 구조가 간단하며, 문서화가 잘 되어 있고, 동일한 언어로 작성되어야 한다[2].

기능 중심으로 구성된 부품의 재사용성 평가는 규모(size)와 난이도(difficulty)의 메트릭스를 수집한 정적 분석기에 의해 가장 쉽게 측정될 수 있다[1]. 따라서 제안된 모델에서는 기본적인 측정 메트릭스로 LOC(Lines of Code), McCabe의 Cyclomatic number, Halstead의 Volume, Difficulty, Effort를 조합하여 재사용성을 평가한다.

4.3 결정 기준의 선택

재사용성 결정은 실제로 모듈을 분류하는 기준에 따라서 모듈의 재사용성을 분류하는 것으로, 여러 연구들은 실험적인 연구들 통해서 모듈들 분류하는 기준들을 제시하였다.

산업 현장에서의 실험적인 연구들은 코드의 라인수(LOC)는 50-100 범위, 복잡도(Cyclomatic number)는 10 이하, 난이도(Difficulty)는 10 이하, 프로그램 노력도(Effort)는 1,000 이하가 적합하고, 볼륨(Volume)이 크면 부품의 이해와 적용을 어렵게 하므로, 볼륨은 평균적으로 200 - 10,000의 범위가 적합하다고 제시하였다[1][3][7][12][13].

그러므로, 본 연구에서는 이 측정값들의 경계를 기존의 여러 연구들을 수용하여 〈표 1〉과 같이 설정하였다.

표 1 제안된 모델을 위한 측정값들의 범위

Measure \ Domain	VS	LS	LC	VC
LOC	<50	50~100	100~150	>150
Cyclomatic Number	<10	10~20	20~50	>50
Volume	<200	200~1000	1000~10000	>10000
Difficulty	<10	10~30	30~100	>100
Effort	<50000	50000~100000	100000~300000	>300000

주)VS : Very simple LS : Little Simple
LC : Little Complex VC : Very Complex

이러한 경계들이 산업 현장에서의 연구와 경험을 통해서 증명되어 왔지만 이 경계들은 다소 임의적이며, 실제로 각 현장에서의 요구들은 변할 수 있다[1]. 그러므로 본 모델에서는 이 경계들이 변경되면 전체의 규칙들은 재구성되어 변경된 값들을 수용하는 새로운 규칙들을 생성하게 한다.

4.4 결정 알고리즘 생성

재사용성 결정은 실제로 모듈을 분류하는 기준에 따라서 모듈의 재사용성을 분류하는 것이며, 부품 분류에 관한 정보들은 부품 재사용에 관한 지

식이 된다.

본 연구에서 제안한 〈그림 3〉과 같은 구조의 결정 알고리즘 생성기는 러프 논리를 이용하여 주어진 지식들에서 다음과 같은 단계로 최소 재사용성 평가 규칙을 찾는다.

[Step 1] 속성 도메인의 정의
[Step 2] 속성값 도메인의 정의
[Step 3] 지식 표현
[Step 4] 지식의 감축
  Step 4A 중복된 규칙의 제거
  Step 4B 알고리즘의 감축
  Step 4C 의사결정 규칙의 감축
[Step 5] 알고리즘의 최소화

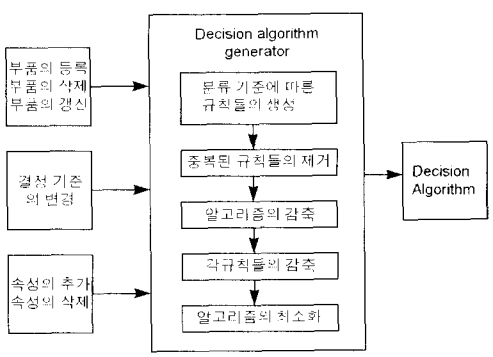

그림 3 알고리즘 생성기의 구조

4.4.1 속성 도메인의 정의

제안된 모델에서의 기본적인 측정 메트릭스는 LOC, McCabe의 Cyclomatic number, Halstead의 Volume, Difficulty, Effort를 조합하여 재사용성을 평가하며, 평가를 위한 속성 도메인은 다음과 같이 정의한다.

A = { Loc, McCabe's Cyclomatic number,
  Halstead's Volume, Difficulty, Effort }

4.4.2 속성값 도메인의 정의

본 모델은 기존 연구들을 수용하여 설정한 〈표 1〉의 경계 값들을 각 속성에 대한 속성값들의 도메인으로 설정하여 다음과 같이 정의한다.

V_{LOC} = {VS, LS, LC, VC}
$V_{Cyclomatic\ Number}$ = {VS, LS, LC, VC}
V_{Volume} = {VS, LS, LC, VC}
$V_{Difficulty}$ = {VS, LS, LC, VC}
V_{Effort} = {VS, LS, LC, VC}

〈표 1〉의 경계 값들은 각 현장에서의 요구들의 변화에 따라서 변경될 수 있으며, 그런 경우 속성 도메인은 자동적으로 재정의된다.

4.4.3 지식표현

부품 라이브러리에 등록 되어있는 부품들에서 각 척도(속성)들에 대한 값(속성값)을 추출하고, 정보 시스템(Information system)을 이용하여 그들을 표현한다.

정보 시스템으로 표현된 지식들은 재사용에 관한 지식들을 포함하며, 이 표에 나타난 지식들은 재사용을 판단하는 규칙들이 된다.

정보 시스템에는 중복된 규칙이 존재할 수 있으므로 중복된 규칙을 제거하며, 원소 범주들의 집합을 그대로 유지한 채, 불필요한 분할인 속성 또는 동치관계들을 제거하는 지식의 감축을 수행한다. 이 과정은 지식 베이스에서 모든 불필요한 지식을 제거하여 필요한 부분만을 갖게 한다.

4.4.4 알고리즘의 감축

알고리즘의 감축은 전체 시스템에서 불필요한 속성들을 제거하는 것으로서, 3.1에서 제시된 과정 과 유사하다. 그러나, 본 연구에서 제시한 모델은 모두가 재사용 가능하다고 전제된 모듈들에서 최소 재사용 결정 알고리즘을 찾는 것이므로 다음과 같이 알고리즘의 의미에 대한 정의를 통하여 알고리즘의 감축을 수행한다.

알고리즘 (P, Q)가 규칙 $r_1, r_2, r_3, \cdots\cdots, r_n$의 모임일 때, 알고리즘 (P, Q)의 의미는 규칙들의 모임 으로, 이래와 같이 정의한다.

$$|(P, Q)| = \{|r_1|, |r_2|, |r_3|, \cdots\cdots, |r_n|\}$$

알고리즘 (P, Q)에서 $a \in P$일 때, $|((P-\{a\}), Q)| \neq |(P, Q)|$이면 속성 a는 알고리즘 (P, Q)에서 필 요이고, 모든 속성 $a \in P$가 알고리즘 (P, Q)에서 필요이면, 알고리즘 (P, Q)는 독립이다. $R \subseteq P$인 알고리즘 (R,

Q)가 독립이고 I(R, Q)=I(R, Q)이면 R은 알고리즘 (R, Q)에서 P의 reduct이다.

일차적으로 감축된 알고리즘은 불필요한 속성들이 제거되어 반드시 필요한 속성들로 이루어진 새로운 정보 표가 된다.

4.4.5 규칙의 감축

속성의 제거를 통해서 일차적으로 감축된 의사결정 알고리즘은 각 의사결정 규칙에서 불필요한 속성들을 제거함으로써 더욱 단순화 되도록 3.2의 결정규칙 감축의 방법을 적용하여 규칙의 감축을 수행한다. 불필요한 속성이 제거된 감축된 규칙은 그 규칙에 필요한 속성들만으로 이루어진 새로운 규칙이다.

4.4.6 알고리즘의 최소화

감축된 알고리즘과 규칙에서 동일한 의사결정류들과 관련된 불필요한 의사결정 규칙들의 제거한다. 다른 규칙들이 제거된 규칙들의 역할을 감당할 수 있으므로, 의사결정 규칙들은 의사결정 과정을 방해하지 않고 삭제될 수 있도록 3.3의 알고리즘 최소화 방법을 적용하여 알고리즘의 최소화를 수행한다.

4.5 재사용성 결정

최소 의사-결정 알고리즘은 데이터 베이스에 저장되어 재사용성 결정의 기초가 된다. 재사용성 결정은 자체 개발된 부품이나 외부에서 획득된 부품들의 부품 라이브러리에 등록시 그들의 재사용성에 관한 질의를 처리한다.

그러나 실제로 각 현장에서의 요구나 외부 환경의 변화에 따른 기준에 합당하지 않은 부품들에 대한 추가나, 기존 부품의 삭제가 이루어질 수 있다. 이 경우에 재사용성 결정 모델은 추가 또는 삭제된 부품들의 정보를 이용하여 재사용성 결정 알고리즘을 재구성한다.

5. 실험 및 결과

5.1 재사용성 결정 알고리즘 생성

기존의 재사용 시스템에서 재사용이 이루어지고 있는 부품들에 대해서 정의된 속성 도메인에 대한 측정값을 구하여 이 값들을 속성값 도메인으로 분류하여 정보 표(Information table)로 구성한다.

본 모델의 실험은 C 컴파일러의 175개에 해당하는 run time 라이브러리를 대상으로 하였다.

5.1.1 규칙의 제거

테스트를 위한 175개의 규칙들 중에서 중복된 규칙을 제거하여 획득된 14개의 규칙으로 구성된 정보 표는 〈표 2〉와 같다.

〈표 2〉 중복된 규칙을 제거한 정보표 (중복)

attribute / rules	a	b	c	d	e	f
r1 (94)	0	0	0	0	0	1
r2 (52)	0	0	0	1	0	1
r3 (6)	0	0	0	2	0	1
r4 (2)	0	1	0	1	0	1
r5 (3)	1	1	0	2	0	1
r6 (5)	1	1	0	2	1	1
r7 (2)	1	1	0	1	0	1
r8 (1)	2	1	1	2	2	1
r9 (2)	0	1	0	2	0	1
r10 (1)	1	0	0	0	0	1
r11 (2)	2	2	0	2	2	1
r12 (1)	1	0	0	2	0	1
r13 (1)	1	1	1	2	2	1
r14 (1)	1	0	0	2	2	1

{a, b, c, d, e, f}는 각 측정 메트릭스 {Lines of Code, Cyclomatic number, Volume, Difficulty, Effort, Reusability}에 대응하고, 각 항목들의 값인 {0, 1, 2}는 각 측정값에 대한 속성값 도메인 {VS, LS, LC}에 대응한다.

5.1.2 알고리즘의 감축

두 번째 단계는 전체 시스템에서 불필요한 속성을 제거한다. 전체 시스템에서 불필요한 속성을 제거하기 위해서 정보 표를 다른 규칙과의 관계를 나타내는 식별가능 행렬로 표현한다. 〈표 2〉를 식별가능 행렬로 나타내면 〈표 3〉과 같다.

〈표 2〉에 대한 core와 reduct를 구하기 위해서 〈표 3〉에서 식별가능 함수를 구하고, 흡수법칙을 이용하여 모든 구성 요소들을 "곱"하면 다음과 같다.

F(A)=(d) (d) (b+d)…(b+c+e) (e) (b+c) = abde

〈표 3〉 표 2에 대한 식별 가능 행렬

	r1	r2	…	r13	r14
r1					
r2	d				
r3	d				
…	…				
r13	a+b+c+d+e	a+b+c+d+e	…		
r14	a+d+e	a+d+e	…	b+c	

{a, b, d, e}가 〈표 2〉의 core이면서 reduct이므로, 속성 c(Volume)는 제거될 수 있다. 결국 〈표 2〉는 〈표 4〉로 감축된다.

〈표 4〉 표 3에서 c가 제거된 정보 표

attribute / rules	a	b	d	e	f
r1	0	0	0	0	1
r2	0	0	1	0	1
r3	0	0	2	0	1
r4	0	1	1	0	1
r5	1	1	2	0	1
r6	1	1	2	1	1
r7	1	1	1	0	1
r8	2	1	2	2	1
r9	0	1	2	0	1
r10	1	0	0	0	1
r11	2	2	2	2	1
r12	1	0	2	0	1
r13	1	1	2	2	1
r14	1	0	2	2	1

5.1.3 규칙의 감축

각 규칙에 대한 core와 reduct를 구하여 각 규칙에 대하여 불필요한 속성을 제거하기 위해서 〈표 5〉를 식별가능 행렬로 나타낸 다음 각 규칙들 의 식별가능 함수를 구하고, 각 식별가능 함수에 흡수법칙을 적용하여 모든 구성요소들을 "곱"하면 각 규칙에 대한 reduct는 〈그림 4〉와 같다.

```
Fr1(A)=(d)(d)(b+d)…(a+b+d+e)(a+d+e)=ad
Fr2(A)=(d)(d)(b)…(a+b+d+e)(a+d+e)=bd
Fr3(A)=(d)(d)(b+d)…(a+b+e)(a+e)=abd
        …
Fr12(A)=(a+d)(a+d)(a)…(b+e)(e)=abde
Fr13(A)=(a+b+d+e)(a+b+d+e)…(b+e)(b)= abe
Fr14(A)=(a+d+e)(a+d+e)(a+e)…(e)(b)=be
```

그림 4 각 규칙에 대한 reduct

결국, 〈표 2〉는 각 규칙에 대한 core와 reduct를 나타내는 〈표 5〉로 감축된다.

〈표 5〉 표 2의 각 규칙들에 대한 reduct 리스트

attribute / rules	a	b	c	d	e	f
r1	0	-	-	0	-	1
r2	-	0	-	1	-	1
r3	0	0	-	2	-	1
r4	0	1	-	1	-	1
r5	1	1	-	2	0	1
r6	-	-	-	-	1	1
r7	1	-	-	1	-	1
r8	2	1	-	-	-	1
r9	0	1	-	2	-	1
r10	1	-	-	0	-	1
r11	-	2	-	-	-	1
r12	1	0	-	2	0	1
r13	1	1	-	-	2	1
r14	-	0	-	-	2	1

5.1.4 알고리즘의 최소화

마지막 단계인 알고리즘 최소화 단계에서는 임의의 규칙이 다른 규칙들이 수행하는 역할을 담당할 수 있으면, 그 규칙은 결정 과정에서 제거될 수 있다. 이 과정은 임의의 집합들의 합집합에서 불필요한 집합을 제거하는 것과 유사하다.

〈표 5〉의 모든 의사결정 규칙들은 이미 감축된 것이므로, 더 이상 단순화될 수 없다. 그러므로 〈표 2〉는 최종적으로 〈그림 5〉와 같은 최소 알고리즘으로 표현된다.

최종적으로 획득된 최소 의사결정 알고리즘은 임의의 모듈에 대한 재사용성을 판정하는 규칙들의 모임으로 구성된다.

5.1.5 재사용성 결정

테스트를 위한 4개의 모듈에 대해서, 정의된 각 메트릭스들에 대한 속성 도메인의 정의와 지식 표현은 〈표 6〉과 같다.

(Rule_1: $a_0d_0 \rightarrow f_1$) If LOC<50 and D<10, then reusable
(Rule_2: $b_0d_1 \rightarrow f_1$) If CN<10 and 10≤D<30, then reusable
(Rule_3: $a_0b_0d_2 \rightarrow f_1$) If LOC<50 and CN<10 and 30≤D<100, then reusable
(Rule_4: $a_0b_1d_1 \rightarrow f_1$) If LOC<50 and 10≤CN<20 and 10≤D<30, then reusable
(Rule_5: $a_1b_1d_2e_0 \rightarrow f_1$) If 50≤LOC<100 and 10 ≤ CN<20 and 30≤D<100 and E<50000, then reusable
(Rule_6: $e_1 \rightarrow f_1$) If 50000≤E<100000, then reusable
(Rule_7: $a_1d_1 \rightarrow f_1$) If 50≤LOC<100 and 10≤ D<30, then reusable
(Rule_8: $a_2b_1 \rightarrow f_1$) If 100≤LOC<150 and 10≤ CN<20, then reusable
(Rule_9: $a_0b_1d_2 \rightarrow f_1$) If LOC<50 and 10≤CN<20 and 30≤D<100, then reusable
(Rule_10: $a_1d_0 \rightarrow f_1$) If 50≤LOC<100 and D<10, then reusable
(Rule_11: $b_2 \rightarrow f_1$) If 20≤CN<50, then reusable
(Rule_12: $a_1b_0d_2e_0 \rightarrow f1$) If 50≤LOC<100 and CN<10 and 30≤D<100 and E<50000, then reusable
(Rule_13: $a_1b_1e_2 \rightarrow f_1$) If 50≤LOC<100 and 10≤ CN<20 and 100000≤E<300000, then reusable
(Rule_14: $b_1e_2 \rightarrow f_1$) If CN<10 and 100000≤ E<300000, then reusable

그림 5 최소 의사결정 알고리즘

이때, q2.c와 q4.c($a_0b_0c_1d_0e_0$)는 rule_3($a_0b_0d_2 \rightarrow f_1$)을 만족하므로 재사용 가능으로 결정할 수 있고, q1.c와 q3.c($a_0b_0c_1d_1e_0$) 또한 rule_2($b_0d_1 \rightarrow f_1$)를 만족하므로 재사용 가능으로 결정할 수 있다.

5.2 환경 변화에 따른 결정 알고리즘의 재구성

환경 변화에 따른 적응성의 실험은 〈그림 3〉을 이용하여 속성의 제거 및 추가, 모듈의 삽입, 분류 기준의 변경과 같은 환경 변화에 능동적으로 대처하여 알고리즘을 재구성하는 것을 보여준다.

5.2.1 속성의 제거 및 추가

〈표 6〉 테스트를 위한 4개의 모듈에서 획득된 정보 표

attribite / module	a	b	c	d	e
q1.c	0	0	1	1	0
q2.c	0	0	1	2	0
q3.c	0	0	1	1	0
q4.c	0	0	1	2	0

속성의 제거 및 추가의 경우, 속성들이 추가 또는 변경되었다는 정보가 재사용 결정 시스템에 통보되면, 결정 시스템은 재사용 부품 라이브러리에 저장된 모듈들에서 변경된 속성들에 대한 측정값들을 추출하여 능동적으로 재사용 결정 알고리즘을 생성한다.

〈표 2〉에서 속성 d(Difficulty)를 제거하고, 속성 d를 제외한 나머지 속성들로 구성된 최소 결정 알고리즘은 〈표 7〉과 같다.

〈표 7〉 속성 d를 제거한 정보 표에 대한 최소결정 알고리즘

attribite / module	a	b	c	e	f
r1	0	0	-	-	1
r2	0	1	-	-	1
r3	1	1	-	0	1
r4	-	-	-	1	1
r5	2	1	-	-	1
r6	1	0	-	0	1
r7	-	2	-	-	1
r8	1	1	-	2	1
r9	-	0	-	2	1

5.2.2 부품의 삽입

기존의 재사용 결정 알고리즘을 통해서는 재사용 불가능한 것으로 판명된 모듈을 재사용 부품 라이브러리에 강제적으로 삽입하는 경우가 있을 수 있다. 이 경우 재사용 결정 모델은 추가된 모듈의 정보를 기존의 재사용 결정 알고리즘과 결합하여 새로운 결정 알고리즘을 능동적으로 생성한다. 〈표 2〉에서 속성 d(Difficulty)가 제거되고, 새로운 모듈(규칙: $abce \rightarrow f$)이 삽입된 정보표의 최소 결정 알고리즘은 〈표 8〉과 같다.

〈표 8〉 새로운 규칙이 추가된 정보표에 대한 최소 결정

알고리즘

attribute / rules	a	b	c	e	f
r1	0	0	-	-	1
r2	0	1	-	-	1
r3	1	1	-	0	1
r4	-	1	-	1	1
r5	2	1	-	-	1
r6	1	0	-	0	1
r7	2	2	-	-	1
r7′	-	2	-	2	1
r8	1	1	-	2	1
r9	-	0	-	2	1
r10	1	2	-	-	1
r10′	-	2	-	1	1

5.2.3 분류 기준의 변경

소프트웨어 기능에 대한 복잡·다양한 사용자의 요구를 충족시키기 위해서 소프트웨어는 점차 더 복잡해지고 있다. 소프트웨어의 복잡도 증가는 분류기준의 변경을 요구하게된다.

〈표 1〉에서 제시된 분류 기준이 〈표 9〉와 같이 변경될 경우, 〈표 2〉에서 속성 d를 제거하고, 변경된 분류기준을 적용하여 감축된 최소 결정 알고리즘은 〈표 10〉과 같다.

〈표 9〉 변경된 분류 기준

Domain / Measure	VS	LS	LC	VC
LOC	<50	50~100	100~150	>150
Cyclomatic Number	<10	10~20	20~50	>100
Volume	<2000	2000~5000	5000~10000	>10000
Difficulty	<10	10~30	30~50	>50
Effort	<50000	50000~100000	100000~105000	>105000

〈표 7〉, 〈표 8〉, 〈표 10〉은 속성들의 변화와 규칙(모듈)들의 삽입, 삭제뿐만 아니라 속성들의 분류기준의 변화에 능동적으로 대처하여 재사용 결정 알고리즘을 생성하는 것을 보여준다.

6. 결 론

〈표 10〉 변경된 분류 기준을 적용한 정보 표에 대한 최소 결정 알고리즘

attribite / module	a	b	c	e	f
r1	0	0	-	-	1
r2	0	1	-	-	1
r3	1	1	-	0	1
r4	-	-	-	1	1
r5	2	1	-	-	1
r6	1	0	-	0	1
r7	-	2	-	-	1
r8	1	1	-	2	1
r9	-	0	-	2	1

본 연구는 실제로 재사용되고 있는 부품들과 정량적인 척도들과 분류 기준을 이용하여 부품의 삽입, 삭제, 분류 기준의 변경 등의 환경 변화에 능동적으로 대처하기 위해 러프 집합 이론을 이용한 재사용성 결정 모델을 제안하였다.

본 연구에서 제안된 모델은 이미 재사용되고 있는 C 컴파일러의 175개의 런 타임 라이브러리에 적용, 실험하여 다음과 같은 결론을 얻었다.

첫째, 프로그램의 크기, 구조, 알고리즘에서의 복잡도의 변화와 새로운 품질 기준을 쉽게 수용하여 모델 외부에서의 환경 변화에 능동적으로 대처할 수 있다.

재사용 부품 라이브러리에 대한 삽입과 삭제 및 부품의 변경이 발생하거나, 분류 기준의 변경이 발생시 부품 라이브러리에 있는 모듈들의 정보를 추출하여 새로운 최소 재사용성 결정 알고리즘을 능동적으로 생성할 수 있다.

둘째, 재사용성 결정 알고리즘의 생성이 자동화될 수 있으므로 재사용 관리 시스템에 가장 적합한 알고리즘을 쉽고 용이하게 구성할 수 있다.

셋째, 기존의 라이브러리 관리 시스템의 재사용성 평가 서브시스템으로 이용될 수 있다.

본 연구에서 제안된 모델을 범용적으로 사용하기 위해서는 다음과 같은 연구가 추가로 진행되어야한다.

첫째, 언어의 범용성으로, 특정 언어를 사용해야하는 제한 사항이 없이 재사용 요구에 부응하기 위해서 다른 언어들을 포함할 수 있는 연구가 필요하다.

둘째, 언어의 독립성으로, 절차 언어에 한정하지

않고 품질 평가 척도인 응집도(cohesion), 결합도(coupling) 등을 이용하여 객체 지향 언어들에 대한 품질 평가도 수용할 수 있어야한다.

셋째, 각 속성들의 중요도를 평가하고 그들의 활용 방안에 대한 연구가 이루어져야한다.

참고문헌

[1] L.J.Arthur, "Measuring Programmer Productivity and Software Quality", John Wiley & Sons, New York, 1985, pp.138-142

[2] Bruce Barnes, Thomas Durek, John Gaffney, Arthur Pyster, "A Framework and Economic Foundation for Software Reuse", Proceedings of the Workshop on Software Reusability and Maintainability, Oct, 1987

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

[4] M.Halstead, "Element of Software Science", Elsevier Noteh-Halland, New-York, 1977

[5] P.A.V.Hall "Software Components and Reuse-Getting More out of Your Code", Information and Software Technology, Feb, 1987, pp.38~43

[6] Horowitz,E., Munson, J.B., "An Expensive View of Reusable Software", IEEE Transaction on software Engineering SE-10(5), Sep, 1984, pp.552~563

[7] Lewis John, Henry Salie, "A Methodology for Integrating Maintainability Using Software Metrics", Proceedings:Conference on Software Maintenance, Miami, Florida, IEEE, Oct 16-19, 1989, pp.32-39

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

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

[10] Z. Pawlak, "Rough Sets-Theoretical Aspects of Reasoning about Data", Kluwer Academic Publishers, Dordrecht, Boston, London, 1991

[11] V.Y.Shen, S.D.Conte, H.E.Dunsmore, "Software Science Revisited : A Critical Analysis of the Theory and its Empirical Support", IEEE Trans., SE,VOL.SE-9,NO.2, 1976, pp.308-320

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

[13] Horst Zuse "Software Complexity-measures and methods", Walter de Gruyter, Berlin New York, 1991, pp.25-37

[14] 강문설, "재사용을 위한 소프트웨어 부품의 분류 방식 및 검색모델", 박사학위 논문, 전남 대학교, 1994

[15] 이경환 외, "소프트웨어 공학", 청문각, 1993, pp.303~315

[16] 이철희 외, "소프트웨어 공학", 홍릉과학 출판사, 1989

[17] 정화식, 양해술, "소프트웨어 재사용가능성의 정량적인 측도", 한국 정보처리학회 논문지 제 2권 2호, 1995, pp.176-184

이 성 주 (Sung-Joo Lee)

1970년 한남대학교 물리학과(학사)
1992년 광운대학교 전자계산학과(석사)
1998년 2월 대구효성카톨릭대학교(박사)
1988년~1990년 조선대학교 전자계산소 부소장
1995년~1997년 조선대학교 정보과학대학장
1981년~현재 조선대학교 전자계산학과 교수
관심분야 : 소프트웨어 공학, 프로그래밍 언어, 객체지향 시스템, 러프집합

최 완 규 (Wan-Kyoo Choi) 종신회원

1988년 서울대학교 종교학과(학사)
1992년~1993년(주)공성통신 전산실
1993년~1994년 한양시스템 전산실
1997년 조선대학교 전자계산학과 (석사)
1997년~현재 조선대학교 전자계산학과 박사과정
관심분야 : 소프트웨어 공학, 프로그래밍 언어, 객체지향 시스템, 러프집합