소프트웨어 재사용가능성의 정량적 측도

장 화 식† 박 만 곤††

요 약

소프트웨어 재사용은 생산성과 품질을 높일 수 있는 유망한 방법임에도 실제로는 잘 활용되지 못하고 있다. 이러한 문제는 소프트웨어 품질에 대한 정량적인 측도가 부재한 이유일 것이다. 본 논문에서 제안한 재사용 소프트웨어의 정량화는 기존의 소프트웨어로부터 모듈을 추출한 후 이 모듈을 재사용 평가 측도에 적용하여 재사용 여부를 측정하는 것이다. 먼저 재사용할 모듈을 측정하기 위해 품질의 인자를 범용성, 단순성, 유지보수성, 모듈성으로 구분하였으며, 인자별로 모듈을 분류 측정한 후 최종적으로 재사용 여부를 결정하게 된다. 제안한 측도의 장점은 재사용하고자 하는 모듈을 정량적으로 측정하므로 부적절한 모듈의 재사용을 조기에 정확히 발견할 수 있다.

On the Quantitative Metrics of Software Reusability

Hwa Sik Jang† and Man Gon Park††

ABSTRACT

The software reuse is a prospective way to improve software productivity and quality but not applied very well in practice, because there is no quantitative metric for software quality. In this paper we proposed the quantification of the reuse of software that we can measure the possibility of the reuse by applying the reuse assessment metric to the module after the extraction of a module from existing software. For measuring the module that can be reused, we divided the factors of quality by the generality, simplicity, maintainability and modularity, and identified and measured the module by the factors and finally decided the possibility of the software reuse. The advantage of the proposed metric is that we can find the inappropriate reuse of module exactly at the beginning by measuring quantitatively the module to be reused.

1. 서 론

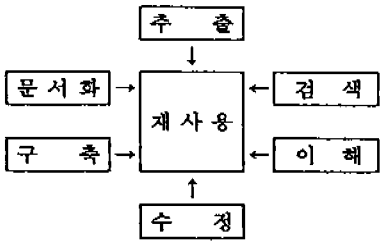
하드웨어 기술의 급속한 발전과 범용 컴퓨터의 보급은 소프트웨어 엔지니어들에게 위기의식을 불러 일으키게 되었으며, 새로운 소프트웨어를 요구하는 수요는 증가하고 소프트웨어 생산성과 품질은 이에 비해서 향상되지 못하고 있다. 이러한 '소프트웨어 위기' 현상을 해결하기 위한 방안으로 선진국에서는 소프트웨어의 생산성과 품질을 높이는 새로운 소프트웨어 개발 기법에 대해서 많이 연구하고 있다. 이를 위한 구체적인 기법으로 소프트웨어 재사용 가능성(software reusability)에 대한 연구가 활발히 진행되고 있는데, 이 기법을 사용함으로써 소프트웨어 개발기간을 단축할 수 있고, 소프트웨어 개발비용을 줄일 수 있으며, 오류가 감소함으로써 소프트웨어 신뢰도를 향상시킬 수 있다[2, 7].

소프트웨어 재사용이란 개발된 모듈, 문서화, 시험사례, 프로그램 등을 동일한 응용업무, 서로 다른 응용업무, 혹은 서로 다른 회사간에 재사용하거나 일부 수정 후 다시 사용할 수 있는 개념을 말한다[5]. 소프트웨어 재사용은 소프트웨어를 개발하는데 필요한 만한 모든 정보가 재사용될 수 있다[8]. 즉 소프트웨어 라이프 사이클 전체 단계에서 발생하는 정보를 재사용할 필요가 있다.

이전의 연구에 의하면 소프트웨어를 재사용한다면 생산성을 상당히 증대할 수 있다고 하였다[14, 15]. 특히 일본에서는 소프트웨어 공장의 개념이 성공을 거두어 년간 14%의 생산성 향상을 얻었다고 보고하였다[10]. 결국, 소프트웨어간에 중복되는 부분을 재사용함으로써 개발기간을 단축하고

† 정 회 원 : 밀양산업대학교 강사
†† 정 회 원 : 부산수산대학교 전자계산학과 교수
논문접수 : 1994년 1월 21일, 심사완료 : 1995년 3월 14일

품질을 높일 수 있다는 것이다. 일반적으로 재사용 시스템의 기능들은 (그림 1)과 같다[1, 2].



(그림 1) 재사용 시스템의 기능들
(Fig. 1) Functions of Reuse System

소프트웨어 재사용은 방법론적으로는 active 방법과 passive 방법으로 구분될 수 있다[1, 2]. active 방법은 목적 시스템을 생성하기 위해 수행되는 부분을 포함한다. 이것은 제너레이터(generator)를 이용한 패턴의 재사용이라고 볼 수 있다. 전형적인 시스템으로는 화면 생성기, 언어변환 시스템[3] 등이 있다. 반면에 passive 방법은 새로운 목적 시스템을 생성하기 위해 결합되는, 포괄적이고 재사용 가능한 소프트웨어 부품을 포함하는 소프트웨어 라이브러리를 만드는 방법을 말한다. 이것은 부가적인 재작업없이 소프트웨어와 기존의 소프트웨어 라이브러리를 통합하기 위해 사용될 수 있다. 이 방법과 연관된 전형적인 시스템으로는 서브루틴 라이브러리, UNIX의 파이프기능 등이 있다. 그리고 소프트웨어 라이프 사이클을 본래대로 사용할 수 있다는 장점이 있다[1].

기존의 재사용 평가 측도로서는 CARE 시스템과 McCall의 품질 인자가 있다[4, 12]. 이러한 측도들은 소프트웨어 모듈들을 재사용하기 위한 측도로서는 명확하지는 못하다. 왜냐하면 이들 두 가지 측도는 재사용하고자 하는 모듈을 추출하였을때 그 모듈을 정량적으로 측정하기 위한 방법을 제시하고 있지 않다. 따라서 본 논문에서는 여러가지 재사용 인자들 중에서 모듈을 중심으로 하여, 제2장에서는 기존의 재사용 평가 측도인 CARE 시스템과 McCall의 재사용 인자의 문제점에 대해서 기술하였으며, 제3장에서는 모듈의 재사용 가능성 평가의 정량화를 위한 품질 인자로써 범용성, 모듈성, 유지보수성, 단순성을 제시하였으며, 이에 따른 각각의 인자에 대해 정량화를 위한 측도를 제시하였다. 제4장에서는 앞 장에서 얻은 재사용가능도에 대한 측도에 가중치를 적용하지 않았을 경우와 가중치를 적용했을 경우에 대해 예제를 제시하였으며, 제5장에서는 본 논문에서 제안한 소프트웨어 재사용가능도에 대한 측도들의 장점과 문제점에 대해서 논의하였다.

2. 기존의 재사용 평가 측도

재사용을 위한 모듈 평가 모형은 소프트웨어 품질의 여러 측도를 사용하여 소프트웨어 부품을 재사용할 수 있는지를 판단하는 모형이다. 이런 평가 모형을 이용한 대표적인 시스템은 CARE 시스템이다[4]. 이 시스템에서는 부품들을 재사용하기 위하여 재사용에 영향을 미치는 품질 인자로써 사용성(usefulness), 비용, 품질을 정의하였으며, 이 품질 인자 측도로써 Halstead의 부품의 크기(volume), McCabe의 순환수(cyclomatic number)로 정의한 모듈의 복잡도(complexity), 실제 규모가 추정치와 얼마나 근접하느냐에 따라 측정하는 모듈의 정규성(regularity), 모듈의 호출수에 따른 재사용 빈도수(reuse frequency)를 기초로 하여 재사용 여부를 평가하고 있다.

2.1 CARE 시스템의 품질 인자 측도

(1) 모듈의 크기

모듈의 크기는 Halstead의 소프트웨어 과학(software science)을 사용하여 측정되어질 수 있다. 소프트웨어 과학은 프로그램을 오퍼레이터와 오퍼랜드라고 하는 독립된 원소들의 집합으로 정의한다.

CARE 시스템에서는,

η_1 = 오퍼레이터의 수,
η_2 = 오퍼랜드의 수
N_1 = 오퍼레이터가 사용된 총 수
N_2 = 오퍼랜드가 사용된 총 수

라고 정의하였으며, 오퍼레이터와 오퍼랜드를 사용하여 모듈의 크기는 다음과 같이 나타내었다.

$$V = (N_1+N_2)\ \log_2(\eta_1+\eta_2).$$

(2) 모듈의 복잡도

CARE 시스템에서는 프로그램 제어의 복잡도를 McCabe의 순환수를 사용하여 정의하였다.

$$v(G) = e - n + 2,$$

여기에서 e는 그래프 G에서 간선들의 수이고 n은 노드들의 수이다. 일반적으로 복잡도가 10이하일 경우에 소프트웨어는 매우 구조적이고 안정된 프로그램이라 할 수 있다.

(3) 모듈의 정규성

모듈의 정규성은 실제 규모가 추정치와 얼마나 접근하느냐를 측정하는 것이다. 모듈의 실제 길이를

$$N = N_1 + N_2$$

라하면, 추정된 길이는

$$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

이며, 모듈의 정규성은

$$r = 1 - \frac{N - \hat{N}}{N} = \frac{\hat{N}}{N}$$

으로 나타내었다. 이때 r이 1에 접근하면 그 모듈은 정규성이 높아진다.

(4) 모듈의 재사용 빈도수

$n(X)$를 시스템에서 X가 호출된 수라하고 시스템이 사용자가 정의한 모듈 X_1, X_2 … X_N과 표준 환경에서 정의한 모듈 S_1, S_2, … S_M으로 구성되어 있다고 가정할 때, 어떤 모든 C가 호출된 수와의 비율은

$$V_o(C) = \frac{n(C)}{\frac{1}{M}\sum_{i=0}^{M} n(S_i)}$$

으로 나타내었다.

2.2 McCall의 재사용 품질 인자

McCall은 모듈들을 재사용하기 위하여 재사용에 영향을 미치는 품질 인자로서 (그림 2)에서 보는 바와 같이 범용성(generality), 모듈성(modularity), 자기 문서화(self documentation), 하드웨어 독립성(hardware independence), 소프트웨어 시스템 독립성(software system independence)으로 정의하였다 [12]. 여기에서 범용성이라함은 재사용 가능한 모듈의 가장 기본적인 특성으로서 특정한 응용분야나 영역만을 위하지 않고 일반적으로 활용될 수 있는 정도를 말한다. 재사용되는 소프트웨어 구성요소의 기본단위는 모듈이므로 각 모듈은 정보은닉, 추상화와 같은 기본 여건을 갖추어야 하며 최소한의 결합도와 최대한의 응집력을 가져야 한다. 재사용되는 모듈들은 실행될 하드웨어 기종과는 무관해야 하며 타 소프트웨어와의 관계에서도 무관해야한다. 그리고 소프트웨어 구성요소의 재사용에 있어 모듈의 정확한 기능, 용법, 그리고 인터페이스를 알려주는 것이 필요하다.

(그림 2) McCall의 재사용 품질 요인
(Fig. 2) McCall's Reuse Quality Factors

2.3 CARE 시스템과 McCall의 품질 인자 문제점

CARE 시스템은 네가지 측도를 사용하여 재사용을 위해 추출된 모듈의 사용성, 비용, 품질을 정확히 측정하고자 하였다. 하지만 이러한 네가지 측도로써 품질의 인자를 정확히 측정하기는 어렵다. 왜냐하면 여기에서 품질의 측도로서 사용하는 네가지 방법은 오퍼레이터와 오퍼랜드에만 치중하였고, 또 프로그램 제어 흐름도만을 기준으로 함으로써 모듈을 설명하는 결합도, 응집력, 정보은닉, 주석 그리고 문서화의 정보는 전혀 반영하지 않고 있다. 또한 모듈의 재사용에 필요한 품질 인자를 제시하면서도 네가지 측도를 사용하여 측정하지 못하는 요인이 존재하기 때문이다.

McCall의 재사용 품질 인자로서는 생명주기 단계에서 최근들어 급격히 중요시되고 있는 유지보수 단계를 반영하지 않고 있다. 또한 각각의 품질 인자들을 정량적으로 분석하지 못하고 있다.

따라서, 제3장에서는 이러한 문제점을 기반으로 하여 재사용하기 위한 모듈을 정량적으로 측

정하기 위해 여러가지 측도를 제안하고자 한다.

3. 소프트웨어 재사용 평가의 정량화

재사용되는 모듈의 품질을 보증하기 위해서는 품질의 정량화를 통해 소프트웨어의 가시성을 높여 품질을 정확히 표현함으로써 공정성을 높일 수 있다. 본 장에서는 재사용 가능한 모듈을 평가하는 정량화된 품질 측도를 제안하고자 한다. 이를 위해 사용되는 품질 인자들을 범용성, 단순성, 모듈성, 유지보수성으로 구분하였으며, 이러한 인자들을 사용하여 재사용가능도를 구한다. 또한 여러가지 품질 인자 측도에 가중치를 적용하지 않은 경우와 가중치를 적용한 경우를 나누어서 모듈의 재사용 평가에 대한 정량화를 제안하고자 한다.

3.1 범용성(generality)

범용성은 많은 개발자들에게 공통적으로 필요하고 사용가능함으로써 일반적으로 활용될 수 있는 정도를 말한다. 예를들면, 수치적인 함수 기능이나 보고서 작성, 편집 기능 등은 그 기본 논리를 범용화된 논리로 표준화시킬 수 있다. 재사용될 모듈을 추출하였을때 그 모듈이 과연 개발자들에게 얼마나 많이 사용되어질 수 있는가를 판단하여야 한다. 그러므로 추출된 모듈은 그 기능이 다양해야 하며 일반성을 갖추고 있어야 한다. 이러한 기능을 갖춘 모듈은 프로젝트 내에서 사용되어진 빈도수와 관계가 있다. 따라서 추출된 모듈이 프로젝트 내에서 사용되어진 빈도수가 높았을때 범용성도 높다고 할 수 있다. 범용성을 정량화시키면 다음과 같이 정의할 수 있다.

$$G = \frac{MF}{TMN}, (MF \geq 2) \quad \cdots\cdots\cdots\cdots (1)$$

여기에서 G는 범용성을 나타내며, MF는 재사용하고자하는 모듈의 사용 빈도수이고 TMN은 모듈의 총 수이다.

3.2 단순성(simplicity)

단순성은 프로그램을 어려움없이 이해할 수 있는 정도를 나타낸다. 즉 재사용되기 위한 모듈의 제어구조와 논리구조가 얼마나 간단하게 구성되어 있는가, 코드들이 얼마나 효율적으로 사용되었는가를 나타내는 성질이다. 따라서 단순성을 정의하기 위해서는 다음과 같은 성질을 만족시켜야 한다.

(1) 모듈의 복잡도

McCabe의 순환성 복잡도는 물리적 크기에 의한 것이 아닌 프로그램의 구조, 즉 제어 흐름 그래프를 기초로 복잡도를 측정하였다. 순환성 복잡도는 간선들의 수와 노드의 수가 일치하면 복잡도는 항상 2가 되며, 간선들의 수가 증가할수록 복잡도는 증가하게 된다. 그러므로 복잡도에 영향을 많이 끼치는 것은 간선들의 수이며, 노드들은 복잡도에 커다란 영향을 미치지 않게 된다. 이를 기반으로 복잡도를 정량화시키면 다음과 같이 정의할 수 있다.

$$CP = 1 - \frac{BN}{TLN}$$

여기에서 CP는 복잡도를 나타내고, BN은 분기문의 수 즉 간선들의 수이며 TLN은 모듈의 총 라인수를 나타낸다.

〈표 1〉에서 보는 바와 같이 McCabe는 복잡도에 따르는 소프트웨어 품질을 다음과 같이 평가하고 있다.

〈표 1〉 모듈의 복잡도 분류
〈Table 1〉 Complexity Classification of Module

복잡도(v(G))	설 명
5 이하	매우 간단한 프로그램
5~10	매우 구조적이고 안정된 프로그램
20 이상	문제 자체가 매우 복잡하거나 구조가 필요 이상으로 복잡한 프로그램
50 이상	매우 비구조적이며 불안정한 프로그램

(2) 코드의 효율성

재사용되어지기 위해 추출된 모듈이 비효율적인 코드가 많이 사용되어졌다면, 그 모듈은 기억장소의 낭비를 초래하게 되며 또한 모듈의 성능을 저하시키는 원인이 된다. 그러므로 비효율적인 라인수가 적을수록 코드의 효율성은 높아지게 된다. 일반적으로 코드의 효율성을 저해시키는 요소는 중복되는 코드, 비실행 코드, 미사용 변수

이다. 이와 같은 요인들을 기반으로 코드의 효율성을 정량화시키면 다음과 같이 정의할 수 있다.

$$E = 1 - \frac{EDN}{TLN}$$

여기에서 E는 모듈의 효율성을 나타내며, EDN은 코드의 효율성을 저해하는 요인의 수이고, TLN은 모듈의 총 라인수를 나타낸다.

모듈의 복잡도와 코드의 효율성을 사용하여 모듈의 단순성(SM)을 정의하면 다음과 같이 나타낼 수 있다.

$$SM = (CP + E)/2$$
$$= (1 - \frac{BN}{TLN} + 1 - \frac{EDN}{TLN})/2$$
$$= (2 - \frac{BN + EDN}{TLN})/2. \quad \cdots\cdots (2)$$

3.3 유지보수성(maintainability)

최근에 들어 소프트웨어 예산에서 유지보수 비용이 차지하는 비중이 급격히 높아지고 있으므로, 개발의 생산성 향상만으로는 소프트웨어의 경제성을 찾을 수 없게 되었다. 따라서 유지보수의 중요성은 날로 증가하고 있다. 재사용 가능한 모듈을 추출하기 위해서는 모듈을 검색해서 주어진 기능에 따라 수정, 첨가 등을 해야함으로 모듈의 재사용을 위해서는 유지보수성이 중요한 요인이 된다. 본 논문에서는 유지보수를 위해 필요한 품질 인자를 가독성, 신뢰성, 구조성, 이식성, 수정용이성이라 두고 이를 정량화 하였다.

(1) 가독성(readability)

가독성이라함은 재사용하기 위해 추출된 모듈이 사용자가 얼마나 읽기 쉬운가를 나타낸다. 가독성을 측정하기 위한 요인으로는 코드내의 대·소문자의 구분, 들여쓰기 준수, 주석, 변수에 의미 부여가 있다. 여기에서 주석의 양은 코드의 양보다 많아야 하며, 코드의 양은 Halstead의 소프트웨어 과학을 사용한다. 코드내에서 이러한 요인을 위반한 수가 많으면 가독성은 저하된다. 이를 기반으로 가독성을 정의하면 다음과 같이 정의할 수 있다.

$$RA = 1 - \frac{RDN}{TLN},$$

여기에서 RA는 가독성을 나타내며, RDN은 가독성을 저해하는 요인의 수이며, TLN은 모듈의 총 라인수를 나타낸다.

(2) 신뢰성(reliability)

소프트웨어 신뢰도란 주어진 시간동안 주어진 환경하에서 소프트웨어가 고장나지 않고 사용될 수 있는 확률을 나타낸다. 신뢰성을 검증하기 위해서는 여러가지 시험사례(test case)를 만들어야 한다. 일반적으로 소프트웨어 신뢰도를 측정하는 측도로는 $MTBF$(Mean Time Between Failure)가 있다[11, 13]. $MTBF$는 $MTTF$(Mean Time To Failure)와 $MTTR$(Mean Time To Repair)을 필요로 한다. $MTBF$는 다음과 같은 식으로 정의된다.

$$MTBF = MTTF + MTTR.$$

이외에도 여러가지 측도들이 존재하나, 본 논문에서는 $ROCOF$(rate of occurence of failures) [6]를 사용하여 신뢰도를 구한다.

$$RF = 1 - \frac{FN}{TTN}$$

여기에서 RF는 신뢰성을 나타내며, FN은 시험의 실패횟수를 나타내고, TTN은 총 시험횟수를 나타낸다.

(3) 구조성(structuredness)

구조적 프로그래밍의 개념은 모든 알고리즘을 기술하는 데는 순차구조, 선택구조 그리고 반복구조면 충분하다는 논리인데 이러한 기법을 사용했을때 프로그램은 이해하기 쉽고 정확한 프로그램을 작성할 수 있게 된다. 구조성을 측정하는 요인으로는 구조적 프로그래밍 기법 사용 여부, GOTO문의 사용 여부, 단일 입·출구 사용 여부가 있다. 이러한 요인들을 위반한 수가 많으면 구조성은 저하된다. 이 를 기반으로 구조성을 정의하면 다음과 같이 정의할 수 있다..

$$ST = 1 - \frac{SDN}{TLN}$$

여기에서 ST는 구조성을 나타내며, SDN은 구조성을 저해하는 요인의 수이며, TLN은 모듈의 총 라인수이다.

(4) 이식성(portability)

이식성은 재사용되어지기 위해 추출된 모듈이 특정 하드웨어나 소프트웨어 환경에 구애 받지 않아야함을 나타낸다. 그러므로 모듈의 재사용성과 이식성 사이에는 서로 밀접한 관계가 있다고 할 수 있다. 따라서 이식성을 정량화 시키면 다음과 같이 정의할 수 있다.

$$PA = 1 - \frac{NNP}{TPN},$$

여기에서 PA는 이식성을 나타내며, NNP는 재사용하고자 하는 모듈이 이식되지 않은 횟수를 나타내고, TPN은 재사용하고자 하는 모듈의 총 이식 시도 횟수를 나타낸다.

(5) 수정용이성(modifiability)

수정용이성은 모듈을 수정하는데 있어 얼마나 쉽게 할 수 있는지를 나타낸다. 재사용되기 위해 추출된 모듈이 변경없이 다른 시스템에 적용될 수 있는 경우는 그리 흔하지 않다. 재사용되기 위해 추출된 모듈은 수정함에 있어 용이해야 하며, 프로그램의 특성상 지원하지 않는 기능이 존재할 경우에는 수정할 수 없는 코드로 분류하며, 수정이 용이하지 못하면 재사용하기에는 부적합하다. 따라서 수정용이성은 다음과 같이 정의할 수 있다.

$$MA = 1 - \frac{MNL}{TLN},$$

여기에서 MA는 수정용이성을 나타내며, MNL은 수정할 수 없는 라인수를 나타내고, TLN은 재사용되기 위한 모듈의 총 라인수를 나타낸다.

이와같이 가독성(RA), 신뢰성(RF), 구조성(ST), 이식성(PA), 수정용이성(MA)을 이용하여 유지보수성(MT)을 정의하면 다음과 같이 나타낼 수 있다.

$$\begin{aligned} MT &= (RA + RF + ST + PA + MA)/5 \\ &= [(1 - \frac{RDN}{TLN}) + (1 - \frac{FN}{TTN}) + (1 - \frac{SDN}{TLN}) \\ &\quad + (1 - \frac{NNP}{TPN}) + (1 - \frac{MNL}{TLN})]/5 \\ &= (1 - \frac{RDN + SDN + MNL}{TLN}) + \frac{FN}{TTN} \\ &\quad + \frac{NNP}{TPN})]/5. \quad \cdots\cdots (3) \end{aligned}$$

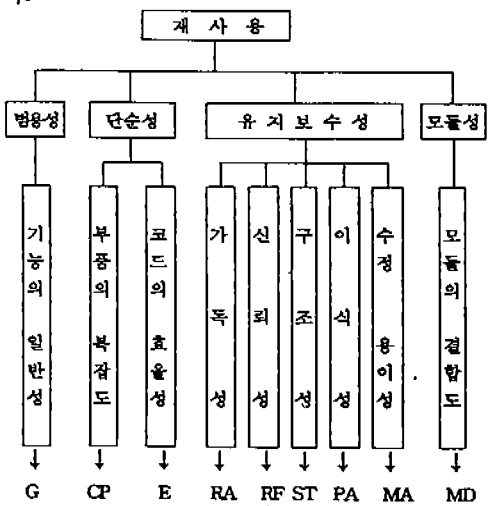
3.4 모듈성(modularity)

모듈성이란 시스템을 구성하는 특정 모듈의 변경에 따르는 영향이 다른 모듈에 최소한으로 미치도록 모듈들이 독립적인 기능을 갖도록 조직화된 성질을 말한다. 모듈과 모듈 사이에서 상호간의 대화가 잦고 오가는 매개변수들(parameters)의 수가 많으면 부자연스럽게 설계되었다는 것을 의미한다. 모듈간의 결합도는 일반적으로 자료결합도(data coupling), 스템프 결합도(stamp coupling), 제어결합도(control coupling), 공통결합도(common coupling), 내용결합도(content coupling)로 구분해 볼 수 있다. 결합도와 응집력은 품질을 평가하는 측도가 되는데, 모듈간의 결합도는 최소로, 응집도는 최대가 되어야 좋은 품질이라 할 수 있다. 모듈성을 정량화하면 다음과 같이 정의할 수 있다.

$$MD = 1 - \frac{MIN}{TLN} \quad \cdots\cdots (4)$$

여기에서 MD는 모듈성을 나타내며, MIN는 모듈간의 인터페이스(interface) 수를 나타내고 TLN은 재사용하고자 하는 모듈의 총 라인수를 나타낸다.

이상과 같이 재사용에 필요한 품질의 인자를 범용성, 단순성, 유지보수성, 모듈성으로 구분하였으며, 이들을 간략히 도식화하면 (그림 3)과 같다.



(그림 3) 재사용을 위한 인자의 도식화
(Fig. 3) Schematization of Factors for Reuse

식 (1), (2), (3) 및 (4)에 따라 9가지 측도에 대한 재사용가능도를 정의하면 다음과 같다.

$$
\begin{aligned}
&Reusability\\
&=[G+SM+MT+MD]/4\\
&=[\frac{MF}{TMN}+(2-\frac{BN+EDN}{TLN})/2+\{\frac{RDN+SDN+MNL}{TNL}\\
&\quad+\frac{FN}{TTN}+\frac{NNP}{TPN})/5\}+(1-\frac{MIN}{TLN})]/4\\
&=[\frac{MF}{TMN}+(1-\frac{BN+EDN}{2TLN})+(\frac{RDN+SDN+MNL}{5TLN}\\
&\quad-\frac{FN}{5TTN}-\frac{NNP}{5TPN})+(1-\frac{MIN}{TLN})]/4\\
&=[(3+\frac{MF}{TMN}-\frac{FN}{5TTN}-\frac{NNP}{5TPN}\\
&\quad-(\frac{5BN+5EDN+2RDN+2SDN+2MNL+10MIN}{10TLN}]/4. \quad \cdots\cdots (5)
\end{aligned}
$$

3.5 품질 인자에 가중치를 적용한 경우

앞 장에서는 가중치를 적용하지 않고 각각의 인자들이 동일한 조건일때 모듈의 재사용가능도에 대하여 알아보았다. 본 장에서는 각각의 인자에 가중치를 적용하였을 때를 고려해 보기로 한다. 여기에서 w_i는 각각의 인자에 대한 가중치이다. 그리고 $\sum_{i=1}^{9} w_i =$ 이다.

먼저 범용성은,

$$
\begin{aligned}
Generality &= G \times w_1\\
&= \frac{MF}{TMN} \times w_1 \quad \cdots\cdots (6)
\end{aligned}
$$

단순성은,

$$
\begin{aligned}
Simplicity &= CP \times w_2 + E \times w_3\\
&= (1-\frac{BN}{TLN}) \times w_2 + (1-\frac{EDN}{TLN}) \times w_3\\
&\quad\cdots\cdots (7)
\end{aligned}
$$

유지보수성은,

$$
\begin{aligned}
&Maintainability\\
&= RA \times w_4 + RF \times w_5 + ST \times w_6 + PA \times w_7 + MA \times w_8\\
&= (1-\frac{RDN}{TLN}) \times w_4 + (1-\frac{FN}{TTN}) \times w_5 + (1-\frac{SDN}{TNL}) \times w_6\\
&\quad + (1-\frac{NNP}{TPN}) \times w_7 + (1-\frac{MNL}{TLN}) \times w_8 \cdots (8)
\end{aligned}
$$

모듈성은

$$
\begin{aligned}
Modularity &= MD \times w_9\\
&= (1-\frac{MIN}{TLN}) \times w_9 \cdots\cdots (9)
\end{aligned}
$$

와 같이 된다.

이상과 같은 식들을 이용해서 모듈의 재사용가능도를 구하려면 위 식들을 모두 합하면 되는데 다음과 같이 나타낼 수 있다.

$$
\begin{aligned}
&Reusability\\
&= Generality + Simplicity + Maintainability + Modularity\\
&= \frac{MF}{TMN} \times w_1 + (1-\frac{BN}{TLN}) \times w_2 + (1-\frac{EDN}{TLN} \times w_3\\
&\quad + (1-\frac{RDM}{TLN}) \times w_4 + (1-\frac{FN}{TTN}) \times w_5 + (1-\frac{SDN}{TLN}) \times w_6\\
&\quad + (1-\frac{NNP}{TPN}) \times w_7 + (1-\frac{MNL}{TLN}) \times w_8 + (1-\frac{MIN}{TLN}) \times w_9\\
&\quad\cdots\cdots (10)
\end{aligned}
$$

4. 재사용 가능성의 측도 계산

제3장에서는 모두 9가지의 요인을 가지고 모듈의 정량화에 대해 기술하였다. 현재까지 모듈 재사용의 정량화에 관한 연구가 부재한 관계로 인하여 제3장 식 (5)에서 얻은 재사용가능도의 값이 어느 정도로 믿을 만한지는 판정하기 어렵다. 따라서 본 장에서는 앞 장에서 제시한 품질 인자에 대해 각각 값들을 적용시켜 품질 인자에 가중치를 주지 않았을 경우와 가중치를 다르게 주었을 경우를 분리해 재사용 가능도에 대한 값을 얻고자 한다.

어떤 프로젝트 A와 B에서 추출한 모듈 A′과 B′에 대한 정보가 〈표 3〉과 같이 각각 나타나고, 이들 프로젝트 내에서 추출한 모듈 A′과 B′은 같은 기능을 수행하는 모듈이라고 가정하자.

〈표 3〉 추출된 모듈에 대한 정보 비교 예
〈Table 3〉 An Example of Information Comparision for Module to be Extracted

정보 / 모듈	TMN	TPN	TLN	TTN	BN	EDN	RDN	SDN	MNL	MIN	FN	NNP	MF
A′	150	30	250	30	15	20	10	10	15	30	2	3	20
B′	150	30	250	30	25	35	20	10	20	35	2	3	15

4.1 품질 인자에 가중치 미적용 경우

〈표 3〉에서 나열된 A′과 B′의 값과 식 (5)를 사용하여 재사용가능도를 구해보면 다음과 같이 나타난다.

〈표 4〉 품질 인자에 가중치를 적용하지 않았을 경우의 재사용가능도 계산
〈Table 4〉 Reusability Evaluation without Weights on the Quality Factors

품질 인자 / 모듈	범용성	단순성		유지보수성					모듈성	재사용 가능도
		복잡도	코드의 표준성	가독성	선회성	구조성	이식성	수정 용이성		
A′	0.1333	0.9400	0.9200	0.9600	0.9333	0.9600	0.9000	0.9400	0.8800	0.7205
B′	0.1000	0.9000	0.8000	0.9200	0.9333	0.9600	0.9000	0.9200	0.8800	0.6917

이상의 두가지 결과에서 보듯이 각각의 모듈이 비록 같은 기능을 수행하고 여러 환경이 비슷하다 할지라도 모듈의 재사용에 대해 저해하는 요소가 많으면 재사용가능도의 값은 낮아지게 됨을 알 수 있다.

〈표 5〉 품질 인자의 가중치 예
〈Table 5〉 An Example of the Weights for Quality Factors

품질 인자 / 가중치	범용성	단순성		유지보수성					모듈성
		복잡도	코드의 표준성	가독성	선회성	구조성	이식성	수정 용이성	
W_1	0.10	0.15	0.10	0.08	0.08	0.08	0.08	0.08	0.25
W_2	0.05	0.15	0.15	0.06	0.08	0.07	0.05	0.08	0.30
W_3	0.05	0.10	0.10	0.15	0.15	0.10	0.05	0.15	0.15
W_4	0.10	0.08	0.12	0.10	0.10	0.15	0.15	0.15	0.05

〈표 6(a)〉 모듈 A′에 가중치를 적용했을 경우의 재사용 가능도 계산
〈Table 6(a)〉 Reusability Evaluation with the Weights on Module A′

품질인자 / 가중치	범용성	단순성	유지보수성	모듈성	재사용 가능도
W_1	0.0133	0.2330	0.3756	0.2200	0.8418
W_2	0.0067	0.2790	0.3287	0.2640	0.8784
W_3	0.0067	0.1860	0.5660	0.1320	0.8907
W_4	0.0133	0.6093	0.1856	0.0440	0.8522

〈표 6(b)〉 모듈 B′에 가중치를 적용했을 경우의 재사용 가능도 계산
〈Table 6(b)〉 Reusability Evaluation with the Weights on Module B′

품질인자 / 가중치	범용성	단순성	유지보수성	모듈성	재사용 가능도
W_1	0.0100	0.2210	0.3707	0.2150	0.8167
W_2	0.0050	0.2640	0.3247	0.2560	0.8517
W_3	0.0050	0.1760	0.5570	0.1290	0.8670
W_4	0.0100	0.1752	0.6020	0.0430	0.8305

4.2 품질 인자에 가중치 적용 경우

〈표 3〉에서 프로젝트 A와 B에서 추출한 모듈 A′과 B′에서 얻은 정보와 식 (10)을 이용하여 각각의 인자에 〈표 5〉의 가중치를 적용하면 재사용가능도는 〈표 6(a)〉와 〈표 6(b)〉 같이 나타난다.

〈표 5〉, 〈표 6(a)〉와 〈표 6(b)〉에서 보는 바와 같이 각각의 품질 인자에 다른 가중치를 적용했을 경우 재사용가능도의 값은 각기 다르게 나타난다. 재사용하고자 하는 모듈을 추출하였을 경우에 품질의 각 인자에 대해 어떤 가중치를 적용할 것인지는 사용자에 따라 다르다. 하지만 품질의 인자에 가중치를 적용하는 경우나 적용하지 않는 경우 둘 다 재사용가능도의 값이 1에 가까우면 모듈의 재사용 기회는 높게 된다.

5. 결 론

모듈의 재사용은 소프트웨어 생산성을 향상시키는 유망한 방법임에도 불구하고 크게 현실화되어 있지는 못하다. 그 이유로서는 관리자와 개발 담당자들의 거부반응, 재사용 기술 적용에 대한 동기 결여, 표준화의 부재, 법적 장애 등을 들 수 있다.

본 연구에서는 기존의 소프트웨어로부터 재사용 가능한 모듈을 추출하여 재사용할 수 있는지를 판정하기 위해 여러가지 정량적인 측도들을 제안하였다. 재사용 가능한 모듈을 정량적으로 분석함으로써 다음과 같은 장점을 얻을 수 있다.

첫째, 같은 기능을 가진 모듈이 존재한다면 실제 어느 모듈이 재사용하기에 적합한지를 정량적으로 비교 분석해 볼 수 있다.

둘째, 품질의 각 인자 즉 범용성, 단순성, 유지보수성, 모듈성의 가중치를 다르게 줌으로써 다양한 소프트웨어 모형을 적용시킬 수 있다.

세째, 적합하지 않은 모듈의 재사용을 조기에 방지할 수 있다.

네째, 재사용 가능한 모듈을 추출할때 지침을 제공한다.

본 연구에서 제안된 재사용가능성 측도가 9가지의 품질 인자에 의해서 모두 표현되었는지의 여부가 문제점이기에 품질 인자들에 대한 추가적 분석이 필요하다고 본다.

참 고 문 헌

[1] Arnold, S. P. and S. L. Stepoway, "The Reuse System : Cataloging and Retrieval of Reusable Software", Proceedings of COMPCONS'87, pp. 376-379, 1987.

[2] Biggerstaff, T. and C. Richter, "Reusability Framework, Assessment, and Direction", IEEE Software pp. 41-49, Mar. 1987.

[3] Boyle, J. M. and M. N. Muralidharan, "Program Reusability through Program Transformation", IEEE Transactions on Software Engineering, pp. 574-588, Sep. 1984.

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

[5] Freeman, P., "A Perspective on Reusability", Tutorial : Software Reusability, IEEE, pp. 2-8, 1987.

[6] Goos, G. and J. Hartmanis, Lecture Notes in Computer Science, Springer-Verlag, 1988.

[7] Horowitz, E. and J. B. Munson, "An Expansive View of Reusable Software", IEEE Transactions on Software Engineering, Vol. SE-10, No. 5 pp. 477-487, Sep. 1984.

[8] Jones, T. C., "Reusability in Programming : A Survey of the State of the Art", IEEE Transactions on Software Engineering, Vol. SE-10, No. 5, pp. 488-493, Sep. 1984.

[9] Lens, M., H. A. Schmid, and P. W. Wolf, "Software Reuse through Building Blocks", IEEE Software, pp. 34-42, July 1987.

[10] Matsumoto, Y., "A Software Factory : An Overall Approach to Software Production", Tutorial : Software Reusability pp. 155-178, Dec. 1986.

[11] Musa, J. D., A. Iannino and K. Okumoto, Software Reliability, McGraw-Hill, pp. 3-29, 1987.

[12] Pressman, R. S., Software Engineering, 3rd edition, McGraw-Hill, pp. 554, 1992.

[13] Shooman, M. L., Software Engineering, McGraw-Hill, 1983.

[14] Tracz,W., "Software Reuse Myths", ACM SIGSOFT Software Engineering Notes, pp. 17-21, Feb. 1987.

[15] Woodfield, S. N, D. W. Embley, and D. T. Scott, "Can Programmers Reuse Software?", IEEE Software, July 1987.

장 화 식

1993년 계명대학교 통계학과 졸업(이학사)
1995년 부산수산대학교 대학원 전자계산학과 졸업(이학석사)
1995년~현재 밀양산업대학교 강사
관심분야: 소프트웨어공학(소프트웨어 신뢰도, 소프트웨어 재사용, 소프트웨어 재공학 등).

박 만 곤

1976년 경북대학교 수학교육과 졸업(이학사)
1987년 경북대학교 대학원 전산통계학(이학박사)
1980년~81년 경남공전 전산학과 교수
1990년~91년 영국 리버풀대학 전자계산학과 객원교수
1992년~93년 미국 캔사스대학교 컴퓨터공학과 교환교수
1981년~현재 부산수산대학교 자연과학대학 전자계산학과 교수
관심분야: 소프트웨어 공학(소프트웨어 신뢰성 평가모형, 결합허용 소프트웨어 신뢰도추정, 소프트웨어 품질보증 등), 소프트웨어 재공학(BPR, 리엔지니어링 평가등).