# 소프트웨어 품질측정을 위한 내부특성 계층화 모델의 제안과 평가

양 해 술† · 이 용 근††


## 요 약

고품질의 소프트웨어를 효율적으로 개발하기 위해서는 소프트웨어 개발의 상위 단계인 설계과정에서 설계품질을 측정하고 그것에 기초하여 설계품질을 향상시키는 방법이 중요하다고 할 수 있다. 그러나 지금까지 대부분의 연구는 사용자 측에서 측정·평가할 수 있는 척도인 품질특성, 품질부특성에 대한 것이 대부분이었고, 개발과정에서 개발자가 실제로 측정·평가할 수 있는 척도인 내부특성과 메트릭스에 대한 연구는 아직까지 미흡한 상태이다. 따라서 본 연구에서는 지금까지 제안되어온 소프트웨어의 내부특성이 포함하는 문제점을 지적하고 이를 해결하기 위해 내부특성의 계층 모델을 제안하였다. 제안 모델에 따라 새롭게 내부특성을 재정리함으로써 내부특성의 누락과 중복을 방지할 수 있다. 또한 구체적인 적용실험에서 메트릭스에 의한 측정의 효율성과 메트릭스의 측정 정밀도에 좋은 개선 효과가 있다는 것을 확인하였다.


# Proposal and Evaluation of Hierachical Model of Internal Characteristics for Software Quality Measurement

Hae-Sool Yang† · Ryong-Geun Rhee††


## ABSTRACT

For the effective development of high qualitative software, it is important to measure the "design quality" through the "process of design" which is the upper phase of software development. And based on it, to improve the "design quality" is also important. But, up to now, most of the researches have been in connection with the quality characteristics, and quality sub-characteristics that can be measured and evaluated by users, but the researches about the internal characteristics and metrics that can be measured and evaluated by developer in the process of the development have not been through going enough. Accordingly, the purpose of this research is to point out the problems included in the internal characteristics of software that have suggested until now, and to suggest the model class of the internal characteristics for the solution of these problems. Omissions and duplications of the internal characteristics can be prevented by arranging than newly according to the model suggestion. Furthermore, it is confirmed that developers can get the efficiency of measurement by metrics, and also can get good improvement effect of metrics measurement accuracy through the concrete application test.


† 종신회원:한국소프트웨어품질연구소(INSQ) 소장
†† 종신회원:한국소프트웨어품질연구소(INSQ) 실장/선임연구원
논문접수:1996년 2월 7일, 심사완료:1997년 4월 28일

## 1. 서 론

최근 들어 소프트웨어의 품질 및 생산성 향상에 대한 개발자나 경영자의 인식이 높아지면서 각 기업이나 연구소에서는 자체적으로 자기 실정에 맞는 새로운 품질관리 및 품질평가 방법론들을 제시하여 체계 구축과 평가를 실시하고 있다. 그러나 이 품질평가 체계나 평가 방식이 표준화되어 있지 못한 실정이기 때문에 그 나름대로 많은 문제점을 지니고 있다. 이를 해결하기 위해서는 국제적인 표준화와 병행하여 하루 빨리 우리 실정에 맞는 소프트웨어 품질평가 체계의 구축이나 방식을 확립하는 것이 급선무이다. 이와 같은 고품질 소프트웨어의 확보 현상이 널리 확산되면서 기존에 이루어졌던 평가 방식의 문제점의 주류를 이루었던 소프트웨어 개발 완료후의 품질평가 방식에서 벗어나 소프트웨어 개발 초기에서부터 완료에 이르기까지 일관성있게 소프트웨어의 품질을 높여 나가는 품질평가 형태로 바뀌어 가고 있다.

따라서 소프트웨어의 품질을 효과적으로 확보하기 위해서는 개발 초기 단계에서부터 품질을 고려하는 것이 바람직하다. 이를 위해서는 사용자 요구 정의로부터 분석·설계에 이르는 단계에서의 고품질의 확보가 매우 중요한 전제 조건이다. 이 때문에 소프트웨어 개발의 상위 단계인 설계과정에서 설계품질을 측정하고 그것에 기초하여 설계품질을 향상시키는 방법이 필요하다. 품질 메트릭스에 의해 정량적으로 품질관리를 하기 위한 지금까지의 대부분의 연구는 사용자 측에서 보아 측정·평가할 수 있는 척도인 품질특성, 품질부특성[1, 2, 4]에 관한 것이고, 개발과정에서 개발자가 실제로 측정·평가할 수 있는 척도인 내부특성과 메트릭스에 대한 연구는 국내에서 INSQ(INstitute of Software Quality)가 주도[6, 7, 13, 14]하고 있으나 아직까지 내부특성의 메트릭스와 평가 척도에 대해서도 개선의 여지가 있다고 본다.

본 연구에서는 먼저 ISO/IEC 9126에서 제안한 6가지 항목의 품질특성과 이를 세분화한 21가지 항목의 품질부특성에 따라 내부특성의 정의에 대해 살펴보았다. 여기에서 내부특성의 항목들 간에는 첫째, 실질적으로 중복되는 면이 있으며, 둘째, 개념상 다른 계층으로 분류해야 할 항목들이 존재하고 있다. 즉, 계층화를 고려하지 않음으로 인하여 지금까지 소프트웨어 품질 평가시 해당 메트릭스가 중복되거나 누락되어 평가의 효율성을 기대할 수 없었던 문제점이 발생하였다. 이와 같은 문제점을 해결하기 위해 본 연구에서는 내부특성의 계층 모델을 제안하고 적용실험을 통하여 그 효율성을 입증하였다.

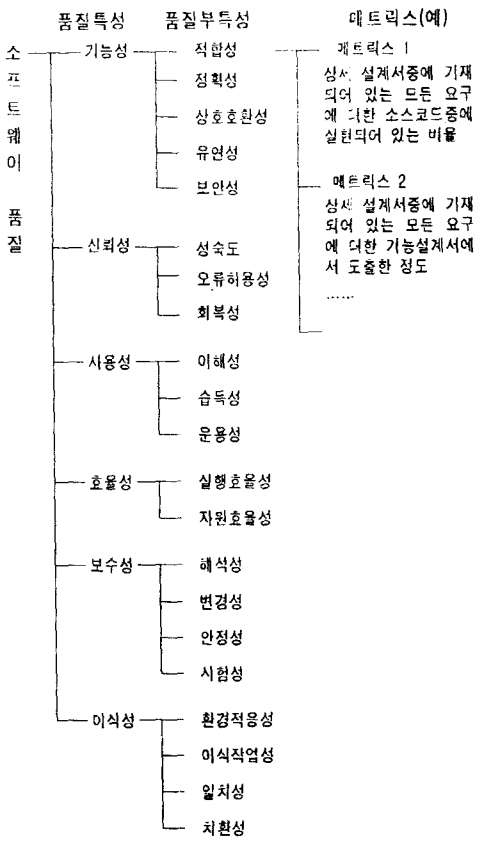
본 논문의 제 2장에서는 소프트웨어의 품질을 평가하기 위해 개발자 입장에서 측정할 수 있는 내부특성 및 메트릭스에 관련하여 기존의 소프트웨어 품질모델의 문제점을 살펴보았으며, 제 3장에서는 제기된 문제점을 해결하기 위한 방법으로 내부특성의 계층화 모델에 대해 제안하고 제안된 내부특성의 계층화 및 구조화에 대해 기술하였으며 제 4장에서는 실제 기업의 상세 설계서를 대상으로 내부특성 메트릭스를 이용하여 측정의 효율성 및 측정 정밀도 등을 평가하고 품질을 측정하여 이에 따른 문제점 분석 및 제안한 계층 모델의 유효성에 대해 고찰하였다.

## 2. 소프트웨어 품질모델의 문제점

### 2.1 소프트웨어 품질의 정량화

소프트웨어를 이용하는 사용자의 관점에서 품질의 구조(외부특성)에 대해서는 1991년에 국제품질표준으로 ISO/IEC 9126에서 표준화되었다[1]. 이 표준에서는 소프트웨어의 품질을 (그림 1)과 같이 품질특성, 품질부특성, 메트릭스의 3계층으로 표현하고 있다. 이와 같은 품질구조에 따라 품질을 정량적으로 측정하는 방법으로서 SQM(Software Quality Metrics)[3, 4, 5, 6, 7]과 SQMAT (Software Quality Measurement and Assurance Technology)[6], ISO/IEC 9126의 품질평가 프로세스 모델 등이 있다. SQM을 참고로 개발된 SQMAT에서는 사용자의 입장에서 본 품질특성을 품질요구 척도라고 부르고 그것을 정확성, 신뢰성, 보수성, 유연성, 사용용이성, 효율성, 안전성, 접속성의 8항목으로 정리하였다. 또한 설계자 입장에서의 품질특성을 품질설계 척도라 부르고 그것을 추적 가능성, 완전성 등 23항목으로 정리하고 있다.

또한 소프트웨어의 품질특성에 대해서도 국제적인 표준화의 노력이 ISO(국제 표준화 기구) 등에서 적극적인 활동으로 전개되고 있다. 소프트웨어와 관계된 중요한 국제 규격은 ISO 9000-3과 ISO/IEC 9126의 두가지가 있다. ISO 9000-3은 소프트웨어의 품질보증



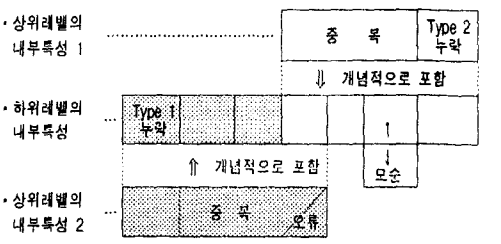

(그림 1) 소프트웨어의 품질 구조
(Fig. 1) Quality structure of software

을 위한 국제 규격이고 ISO/IEC 9126은 소프트웨어의 품질평가를 위한 국제 규격으로 소프트웨어 품질 보증의 기반이 되는 품질평가 기술에 대하여 구체적인 순서와 방법을 표준으로 제공하고 있다. 관련 국제 표준으로서는 여러가지 제안이 검토되고 있는 중이지만 현재는 ISO/IEC 9126에서 정의한 특성을 기준으로 삼아 품질평가를 실시하고 있다. 품질을 정량적으로 표현함으로써 어떤 품질특성, 품질부특성이 어느 정도 품질 목표를 달성하는가를 구체적으로 파악하여 그에 대한 정확한 대응을 하기 위해서는 개발작업의 효율화를 도모하여야 한다. 또한 요구되는 품질 달성의 레벨을 품질목표로서 개발초기에 제시할 수 있기 때문에 과도한 설계를 방지할 수도 있다.

그러나 위와 같은 품질정의는 사용자 관점에서 본 것이기 때문에 개발자가 개발과정에서 품질을 정량적으로 측정하려는 요구를 만족시키지는 못한다. 이 때문에 개발과정에서 평가할 수 있는 소프트웨어의 성질(내부특성)을 설정하고 이 내부특성과 ISO/IEC 9126의 외부특성과의 연관관계를 규명하였다[8, 9]. 이 결과 사용자의 품질요구에 대하여 어느 내부특성을 측정하고 평가하여 목표에 부합할 수 있는가를 확인하는 것이 가능하게 되었다. 소프트웨어의 품질 평가의 시점에서는 품질특성과 품질부특성 및 내부특성외에도 품질에 영향을 미치는 요인과 품질의 영향 등을 생각할 수 있다. 품질에 영향을 미치는 요인은 개발 요원의 기술, 개발 예산 등 소프트웨어 품질에 직접적, 간접적으로 영향을 미치는 속성의 집합이다. 또한 품질에 영향을 미치는 요인은 개발 프로세스의 입력에 관한 속성과 프로세스 자체에 관한 속성으로 구분할 수 있다. 품질의 영향은 사용자의 업무효율, 사용자의 만족도 등 소프트웨어 제품이 직접적, 간접적으로 사용자에게 미치는 영향을 나타내는 속성의 집합이다.

### 2.2 품질모델의 문제점

소프트웨어의 외부특성에 대해서는 ISO9126-1과 9126-2에 품질특성과 품질부특성이라고 하는 계층적인 품질이 정의되어 있고 내부특성에 관해서는 ISO9126-3에서 중간 산출물을 측정하기 위한 척도가 정의되어 있으나 내부특성 메트릭스를 외부특성 메트릭스로 매핑할 것인지는 구조적으로 정의되어 있지 않다. 그러므로 각각의 내부특성이 서로 어떤 연관관계가 있는지 명확하게 정의되어 있지 않기 때문에 다음과 같은 문제점이 발생한다(그림 2).




(그림 2) 내부특성의 구조와 문제점
(Fig. 2) Structure and problem of internal characteristics

(1) 상위의 품질 부특성에 있어야 할 특성이 최하위의 내부특성으로 보고 있는 것이 있기 때문에 다음과 같은 현상이 생긴다.

a. 포함하는 내부특성의 메트릭스와 중복된다(그림 2 상위레벨의 내부특성 1 및 2의 중복).

b. 개념에 누락이 있어도 알 수 없다(그림 2 상위레벨의 내부특성 1의 Type 2 누락).

(2) 동일 레벨에 있어야 할 내부특성의 관계가 애매하다. 이 때문에 다음과 같은 문제점이 생긴다.

a. 개념이 중복되는 것이 있고 같은 메트릭스에서 여러번 측정하는 경우가 있다(그림 2 상위레벨의 내부특성 1 및 같은 2의 오류와 중복).

b. 동일한 이유로 본래의 내부특성이 아닌 다른 내부특성의 메트릭스에서 측정하는 경우가 있다(그림 2 상위레벨의 내부특성 1 및 같은 2의 오류와 중복).

c. 누락되어 있는 내부특성이 있어도 알 수가 없다(그림 2 하위레벨의 내부특성의 Type 1 누락).

d. 모순된 내부특성이 있어도 알 수가 없는 현상이 발생한다(그림 2 하위레벨의 내부특성의 모순).

상기의 문제점으로 인하여 다음과 같은 측정상의 오류가 발생할 가능성이 있다.

측정 오류 : (2)-b, (2)-d

측정 누락 : (2)-c, (1)-b

중복 측정 : (1)-a, (2)-a

## 3. 내부특성의 구조화

### 3.1 내부특성의 계층화

먼저 내부특성의 정의를 살펴보면 〈표 1〉과 같으며 그 중에서 내부특성의 계층성에 대한 문제를 살펴보기 위해 완전성, 추적가능성, 자기기술성, 무모순성 및 단순성을 추출하기로 한다.

- 완전성 : 계획 또는 요구된 *기능*이 충분히 실현되어 있는 성질
- 추적가능성 : 개발공정에 따라 요구명세로부터 실현된 것(*기능*)으로의 관련을 추적할 수 있는 성질.
- 자기기술성 : 기능의 설명 및 *기능과 기능*간의 관련을 설명하고 있는 성질
- 무모순성 : 시스템이나 프로그램이 가지는 동일한 *기능*에 모순이 없거나 프로그램과 도큐멘 트간에 모순이 없는 성질.

이들 4가지 정의는 *[기능]*에 의해 어떤 관련을 가지고 있는 것을 나타내고 있다. 그리고 기능이 충분히 실현되어 있는 완전성과 기능에 모순이 없는 무모순성을 확인할 수 있으면 추적가능하다는 것으로부터 완전성과 무모순성은 추적가능성의 구성요소이다. 즉, 내부특성의 정의에서 **추적가능성⊃완전성, 무모순성**의 계층성이 내재되어 있음을 알 수 있다.

본 연구에서 추적가능성의 개념에는 포인터의 유무와 전개 내용의 올바름 2가지의 개념이 포함되는 것이라고 해석하고 있으며 이와 같이 내부특성이 포함하고 있는 개념에 착안하여 내부특성을 재구성하고 계층 구조를 명확히 하였다.

또한 본 연구에서는 새로운 최상위의 내부특성으로 3가지의 특성을 설정하였다.

- 개발특성 : 개발해야 할 기능에 관한 성질
- 일반특성 : 개발자 입장에서 보아 소프트웨어가 본래 가져야만 하는 성질
- 이 용 성 : 사용자 입장에서 보아 소프트웨어가 본래 가져야만 하는 성질

일반적으로 소프트웨어는 사용자의 요구에 준하여 기능으로 실현시키기 위해 우선 요구사항대로 개발이 진행되고 있는가의 여부를 확인해야 한다. 그리고 제품으로서의 성능과 장해 등에 관한 배려가 필요하고 마땅히 가져야만 하는 성능도 개발과정에서 확인하여야 한다. 또한 실제로 소프트웨어의 사용성을 향상시키기 위해서 소프트웨어가 갖추어야 하는 성질(이용성)도 필요하게 된다. 이들 최상위의 내부특성을 마련하여 각 내부특성을 그 구성요소와 위치를 부여함으로써 각각의 내부특성의 성질을 명확하게 하는 것이 쉬워진다.

### 3.2 내부특성의 계층 모델

내부특성간에 계층화를 발견하고 내부특성의 구조를 재구성하기 위해 각 내부특성이 어느 내부특성의 개념에 포함되는가라는 관점에서 내부특성의 정의를 살펴보았다.

여기에서 *진한 이태릭체* 부분을 살펴보면 기능의 누락은 추적가능성과 관련되고 기능의 미준비 및 부적합은 완전성과 관련된 개념이라는 것을 알 수 있다. 또한 변경용이성은 보수작업의 용이성과 제품관리성

〈표 1〉 내부특성의 정의의 일부
〈Table 1〉 A part of definitions of internal charateristics

| 내부 특성 | 정 의 | 관련 품질특성 | | | | | |
|---|---|---|---|---|---|---|---|
| | | 기능성 | 신뢰성 | 사용성 | 효율성 | 보수성 | 이식성 |
| 1. 완전성(Completeness) | 계획 또는 요구된 기능이 충분히 실현되었는지를 평가 | ○ | ○ | | | | |
| 2. 추적가능성(Traceability) | 개발 공정에 따라 요구 명세로부터 실현된 것(기능)으로의 관련을 추적할 수 있는 성질 | ○ | ○ | | | ○ | |
| 3. 일관성(Consistency) | 설계 기법이나 표기법, 용어, 기호 등이 통일되었는지를 평가 | ○ | ○ | | | ○ | |
| 4. 자기기술성 (Self-descriptiveness) | 기능의 설명 및 기능과 기능간의 관련을 설명 하고 있는지를 평가 | ○ | ○ | | | ○ | ○ |
| 5. 무모순성(Coherence) | 시스템이나 프로그램이 가지는 동일한 기능에 모순이 없거나 프로그램과 도큐멘트간에 모순이 없는지를 평가 | ○ | | | | | ○ |
| ⋮ | ⋮ | | | | | | |
| 36. 제품관리성 (Product Management) | 시스템(소프트웨어)의 각각의 부품들에 대하여 제품관리가 용이한지를 평가 | | | | | ○ | |
| 37. S/W시스템 독립성 (S/W System Independence) | 시스템이 특정 소프트웨어 환경(OS, 컴파일러, 언어, 유틸리티 등)에 의존하는지를 평가 | | | | | ○ | ○ |
| 38. 기계독립성 (Machine Independence) | 시스템이 특정의 하드웨어 환경(시스템 구성, 기종, 장치, 단말 등)에 의존하는지를 평가 | | | | | ○ | ○ |
| 39. 데이타독립성 (Data Independence) | 시스템이 특정 데이타 환경(데이타, DB, DBMS 등)에 의존하는지를 평가 | | | | | | ○ |
| 40. 전달성(Communicativeness) | 시스템의 입출력 형식이나 내용이 어느 정도 사용이 용이하게 통일되어 있는지를 평가 | | | | | | ○ |

[계층화의 정의 예]

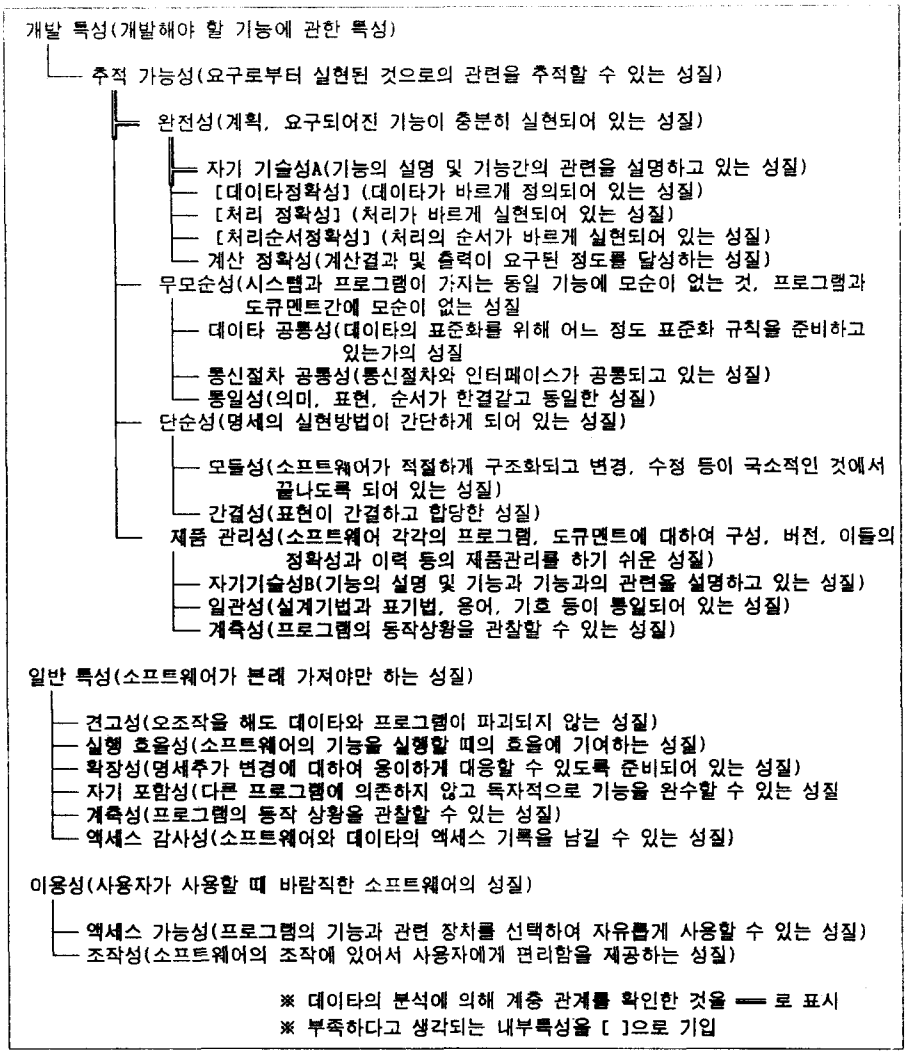
| 계 층 화 | |
|---|---|
| 정의 | 기능의 설명 및 기능간의 관련을 설명하고 있는 성질 |
| 비고 | 소프트웨어가 갖고 있는 기능 및 기능간의 관련이 프로그램(소스 코드)과 설명서나 도큐멘트에 얼마나 기술되어 있는가를 나타내는 것이다. 이것에 의해 *기능의 누락/미준비* 및 *기능*의 부적합이 적어짐과 동시에 *변경이 용이*하게 된다. |

에 관계된다. 또한 자기기술성을 규정하고 있는 '관련을 설명'은 추적가능성의 '관련을 추적', 완전성의 '기능의 충분', 제품관리성의 '관리 용이'를 상세화한 개념이라고 할 수 있으며 이들로부터 우선 다음과 같은 계층화를 도출할 수 있다.

추적가능성 완전성 제품관리성
|- 자기기술성 |- 자기기술성 |- 자기기술성

이와 같이 모든 내부특성에 대해서 내부특성간의 대응관계를 부여한 후 대응관계를 수정하고 보다 상세한 포함관계를 정리한다. 예를 들면 추적가능성에 포함되는 자기기술성은 바로 밑에 가지(branch)로 존재하는 것이 아니고 추적가능성의 바로 하위의 내부특성인 완전성, 제품관리성에 포함되는 가지로 존재하는 것 등을 정리하였다.

이 결과 제안하는 내부특성의 모델구조를 (그림 3)에 나타내었는데 여기에서는 추적 가능성의 구조만을 상세하게 보였으며 뒤에 기술하는 모델의 평가결과도 포함되어 있다. (그림 3)을 살펴보면 계층적으로




(그림 3) 내부특성의 계층화 모델
(Fig. 3) Hierarchical model of internal characteristics

내부특성을 정리함으로써 각각의 내부특성이 품질을 구성하는 기본 요인 가운데 어디에 위치하는지가 명확해짐과 동시에 내부특성이 의미하는 것이 보다 선명해진다. 또한 동시에 누락의 중복, 의미없는 상세화 등 모델 자신의 문제점 발견이 용이하기 때문에 내부특성의 계층화모델을 이용하면 앞에서 지적한 문제점들을 해결할 수 있다. 완전성의 하위 내부특성으로 나타나 있는 [데이타정확성], [처리정확성], [처리순서정확성]은 모델을 검토하는 과정에서 새롭게 부여된 것으로 완전성이라는 상위 내부특성의 개념에서 필연적으로 도출된 것이다. 이와 같이 상위 개념에서 누락되어 있는 하위 개념을 도출할 수 있는 것도 구조화의 커다란 특징이라고 할 수 있다.

## 4. 평가 모델

### 4.1 평가 방법

A사가 개발중인 종합정보시스템의 상세 설계서를

대상으로 내부특성 메트릭스를 이용하여 품질을 측정하고 내부특성의 계층모델의 유효성을 평가하였다. 평가 대상 종합정보시스템은 크게 4가지 업무 즉, 정보통신, 무역정보, 경영정보, 사무자동화로 구분되어 있으며 각 업무는 부수적인 몇 개의 세부 업무들로 구성되어 있다. 평가는 각 내부특성간의 관계를 살펴보고 다음과 같은 사항을 확인/검증하는 것이다.

①내부특성의 계층구조의 타당성

②비계층구조이기 때문에 발생하는 문제점

그러기 위해서는 각 내부특성마다 그 정의 및 보충설명을 참고로 메트릭스를 설정하고 메트릭스마다 품질측정을 실시한다. 이 때 내부특성의 계층성은 고려하지 않고 메트릭스는 평가가 항상 동일하게 이루어지도록 객관적으로 정의되어야 한다. 이 측정에서는 내부특성의 정의 가운데 있는 각 내부특성을 단적으로 표현하는 키워드를 도출하고 이것을 메트릭스로서 정의한다.

예를 들면, 추적가능성의 [요구에서 실현된 것으로의 관련을 추적하는 것이 가능하다]고 하는 정의는 기능이 누락되지 않도록 전개시키고 전개된 내용이 올바르다는 것을 나타낸다. 그리고 전자에 대해서는 〈표 2〉중 TR1.1과 같이 [설계 도중에 요구된 내용이 누락되어 있지 않은 것]을 확인하는 메트릭스로 설정한다. 다음에 설정된 메트릭스를 이용하여 품질을 측정한다. 이 TR1.1의 예에서는 실제로 요구 명세서와 기능 명세서의 내용을 확인하고 요구 명세서에 기술되어 있는 요구에 대해서 만약 기능 명세서에 기술되어 있지 않으면 기능 누락이라고 판단하고 문제점으로 지적한다. 그리고 문제점으로 지적한 것 가운데 해당 메트릭스로 지적하는 것이 타당하다고 생각되는 문제점에 대해서는 올바른 지적이라고 분류하고 다른 메트릭스로 지적해야 할 문제점이라고 생각되는 것은 잘못된 지적이라고 분류한다. 예를 들면, 본 분석의 TR1.1의 측정에서는 (1) 기능 누락, (2) 기입명 오류, (3) 기능의 상세화 부족의 3종류의 문제점을 지적하였다. 그러나 (2)로 분류되는 문제점은 CP1.7 또는 SD1.2로, (3)으로 분류되는 문제점은 CP1.1의 각각의 메트릭스에서 지적해야 하는 문제점이라고 판단하고, (2), (3)으로 분류되는 문제점을 잘못된 지적이라고 분류한다. 본 평가에 사용한 메트릭스의 일람표를 〈표 2〉에 나타내었다.

〈표 2〉 메트릭스 일람표

〈Table 2〉 Metrics table

| 내부특성 | 메트릭스 | 정의 |
|---|---|---|
| 추적가능성 | TR1.1 | 요구명세서중의 모든 요구가 기능 명세서중에 있는 것 |
| 완전성 | CP1.1 | 요구된 모든 기능이 바르게 정의되어 있는 것 |
| | CP1.2 | 모든 기능의 참조명은 애매하지 않을 것 |
| | CP1.3 | 모든 데이타의 근원을 알 수 있는 것 |
| | CP1.4 | 참조된 기능 모두가 정의되어 있는 것 |
| | CP1.5 | 모든 조건, 처리가 각 판단점에 대해 정의되어 있는 것 |
| | CP1.6 | 모든 요구명세 조회표가 해결되어 있는 것 |
| | CP1.7 | 입출력 항목은 사용자 요구에 따르고 있는 것 |
| | CP1.8 | 데이타 및 정수의 속성이 정의되어 있는 것 |
| 자기기술성 | SD1.2 | 모든 참조된 기능, 데이타에 적절한 명칭이 부여된 것 |
| 데이타공통성 | DC1.3 | 모든 데이타의 단위가 일관성이 있는 것 |
| | DC1.4 | 데이타의 정의가 일관성이 있는 것 |
| 단순성 | S1.8 | 각 기능이 읽기 쉬운 것 |
| 모듈성 | MO1.2 | 각 기능의 계층화가 되어 있는 것 |
| 견고성 | RB1.1 | 각 기능에 대해 에러분석을 하고 에러처리가 설정된 것 |
| | RB1.2 | 각 기능에 대해 입력데이타의 허용범위에 대하여 명확한 요구가 있는 것 |
| | RB1.3 | 조작원의 오조작에 대해 처리방법이 정의되어 있는 것 |
| 확장성 | EX1.1 | 추가기능을 위한 준비가 되어 있는 것 |
| 조작성 | OP1.1 | 입력 방법은 통일되어 있는 것 |
| | OP1.2 | 입출력에는 업무에 적합한 용어를 사용하고 있는 것 |
| | OP1.3 | 에러 메시지가 적절한 것 |
| | OP1.4 | 조작순서가 사용자 요구가 적합한 것 |
| | OP1.5 | 중요한 것은 재확인할 수 있는 것 |
| | OP1.6 | 같은 입력을 여러번 보내지 않고 끝내는 것 |

### 4.2 평가 결과

#### 4.2.1 내부특성의 계층성의 검토

3.1절에서 추적 가능성을 예로 메트릭스를 이용하여 내부특성의 계층성을 검증한 결과를 기술하였다. 〈표 3〉에서 추적가능성이 추적하는 기능에 누락이 없는것, 즉 완전성, 무모순성, 단순성 및 제품관리성으로부터 구성되는 것은 확실한 것이므로 측정결과로

부터 다음과 같은 사항을 알 수 있다.

- 추적가능성의 메트릭스에 의해 지적된 문제점 =
  추적가능성 메트릭스에 의해 지적되어야 할 문제점 +
  완전성의 메트릭스에 의해 지적되어야 할 문제점 +
  자기 기술성 메트릭스에 의해 지적되어야 할 문제점
- 완전성의 메트릭스에 의해 지적된 문제점 =
  완전성의 메트릭스에 의해 지적되어야 할 문제점 +
  자기 기술성의 메트릭스에 의해 지적되어야 할
  문제점 +...

위의 결과는 추적가능성 ⊃ 완전성 ⊃ 자기기술성이라는 가정을 증명하는 것이다.

#### 4.2.2. 내부특성의 문제점과 평가

메트릭스에 의해 지적된 설계서의 문제점을 내부

〈표 3〉 추출된 오류의 일부
〈Table 3〉 Sample of extracted error

| 메트릭스 | 바른 지적 | 잘못된 지적 |
|---|---|---|
| TR1.1 (추적가능성) | • 지적 없음 | • 기능 내용 기술 누락(a)<br>• 기능의 기술 오류<br>• 출력 기입명 오류 |
| CP1.1 (완전성) | • 기능 내용의 기술 누락(b)<br>• 기능의 처리 내용 오류<br>• 화면 추이 순서 오류<br>• 기능 키 할당 부여 오류 | • 출력 기입명 오류<br>• 기능명 오류<br>• 입력 항목 오류 |
| CP1.2 (완전성) | • 지적 없음(본래 (e)의 오류를 여기서 지적해야 한다) | • 지적 없음 |
| CP1.3 (완전성) | • 자료 발생 장소가 불분명<br>• 자료 정의 누락 | • 자료 속성이 미정의<br>• 자료 정의가 일관되지 않음<br>• 자료 입력 체크가 없음(c) |
| ⋮ | ⋮ | ⋮ |
| RB1.2 (견고성) | • 자료 입력 체크가 없음(d) | • 지적 없음 |
| OP1.2 (조작성) | • 지적 없음 | • 기능명이 부적절(e) |
| ⋮ | ⋮ | ⋮ |

주) 중복측정 : a의 예 = (b) → (a) 관계인 (a)로 대응
c의 예 = (d) → (c) 관계인 (c)로 대응
측정오류 : b의 예 = (e)로 대응
d의 예 = 해당 없음

특성의 계층구성을 고려하여 재정리한 결과를 〈표 3〉에 나타내었다. 이 표의 [바른 지적/잘못된 지적]란의 기술에서 다음과 같은 문제점을 확인할 수 있다.

(1) 중복 측정
- 추적가능성에서 지적된 [기능 내용의 기술 누락] (원래는 완전성)
- 완전성에서 지적된 [데이타입력의 체크가 없음] (원래는 견고성)

(2) 측정 오류
- 조작성에서 지적된 [기능명이 부적절](원래는 완전성)

이것으로부터 2장에서 세운 가설을 증명할 수 있다. 메트릭스에 의한 지적중 바른 지적/잘못된 지적 및 중복된 지적을 내부특성의 계층성에 기초하여 정리한 결과를 〈표 4〉에 기술하였으며 중복 지적율, 잘못된 지적율, 중복 메트리스율, 잘못된 메트릭스율은 다음과 같다.

① 중복 지적율(52%)
- 중복지적사항을 포함하는 메트릭스 ∋ TR1.1, CP1.1, CP1.3, CP1.6, DC1.3, RB1.3
- 중복지적율 = [(중복지적수)/{바른 지적수)(지적누락수)}] * 100

② 잘못된 지적율(30%)
- 잘못 지적한 사항을 포함하는 메트릭스 ∋ CP1.3, OP1.1, OP1.2
- 잘못된 지적율 = [(잘못된 지적수)/{(바른 지적수)(지적누락수)}] * 100

③ 중복 메트릭스율(33%)
- 중복 메트릭스율 = {(중복 지적사항을 포함하는 메트릭스)/(전 메트릭스)} * 100

④ 잘못된 메트릭스율(13%)
- 잘못된 메트릭스율 = {(잘못된 지적사항을 포함하는 메트릭스)/(전 메트릭스)} * 100

#### 4.2.3 분석 및 고찰

소프트웨어의 품질을 측정하기 위한 내부특성 메트릭스에 의해 지적된 오류건수를 살펴보면 중복 지적/중복 메트릭스에 의한 문제점과 잘못된 지적/잘못된 메트릭스에 의한 문제점으로 나누어 볼 수 있다. 이들 문제점은 내부특성의 평가에 많은 영향을 미치

〈표 4〉 메트릭스에 의해 지적된 오류 건수
〈Table 4〉 The number of indicated error by metrics

| 메트릭스 | 지적한 오류 건수 | | | 계 |
|---|---|---|---|---|
| | 바른지적 | 중복지적 | 잘못된지적 | |
| TR1.1 | 0 | 7 | 0 | 7 |
| CP1.1 | 11 | 5 | 0 | 16 |
| CP1.2 | 3 | 0 | 0 | 3 |
| CP1.3 | 11 | 8 | 16 | 35 |
| CP1.4 | 1 | 0 | 0 | 1 |
| CP1.5 | 1 | 4 | 0 | 5 |
| CP1.6 | 0 | 5 | 0 | 5 |
| CP1.7 | 4 | 2 | 0 | 6 |
| CP1.8 | 5 | 0 | 0 | 5 |
| SD1.2 | 2 | 0 | 0 | 2 |
| DC1.3 | 0 | 5 | 0 | 5 |
| DC1.4 | 5 | 0 | 0 | 5 |
| S 1.8 | 7 | 0 | 0 | 7 |
| MO1.2 | 0 | 0 | 0 | 0 |
| RB1.1 | 5 | 0 | 0 | 5 |
| RB1.2 | 16 | 0 | 0 | 16 |
| RB1.3 | 0 | 5 | 0 | 5 |
| EX1.1 | 0 | 0 | 0 | 0 |
| OP1.1 | 0 | 0 | 1 | 1 |
| OP1.2 | 0 | 0 | 7 | 7 |
| OP1.3 | 7 | 0 | 0 | 7 |
| OP1.4 | 1 | 0 | 0 | 1 |
| OP1.5 | 0 | 0 | 0 | 0 |
| OP1.6 | 0 | 0 | 0 | 0 |
| 계 | 79(54.9%) | 41(28.5%) | 24(16.7%) | 144(100%) |

므로 이는 측정시에 내부특성의 계층화를 도입하면 해결할 수 있다.

(1)중복지적/중복메트릭스

메트릭스에 의해 추출된 문제점 가운데 약 50%는 실질적으로 동일 메트릭스에 의해 반복 측정됨으로써 발생한다. 이들 측정의 시점에서는 내부특성의 계층성을 의식하고 있지 않기 때문에 모든 메트릭스가 독립적으로 측정된 결과라고 생각된다. 그러나 이들 문제점 전부는 측정시에 내부특성의 계층 구조를 도입함으로써 제거할 수 있다. 중복 측정은 표면적으로는 측정 정밀도를 높여 품질에 좋은 영향을 미치는 것으로 보여지나 계층성을 도입하여 수정하면 실제로는 효율성이 나쁘다는 것을 알 수 있다.

〈표 4〉에서 살펴보면 중복 측정의 대부분이 완전성에 관한 메트릭스에서 발생했다는 것을 알 수 있는데 이는 완전성의 개념이 바르게 정리되어 있지 않고 설정된 메트릭스의 개념에 중복이 있기 때문이다. 즉, 측정시점의 완전성은 준비되어 있지 않고 추가해야 할 사항이 있는 것을 구조화에 의해 판명한 것은 3.2절에서 기술한대로이다.

(2)잘못된 지적/잘못된 메트릭스

잘못된 메트릭스의 적용에 의한 목적이외의 내부특성에 관한 문제점의 추출은 30%였다. 이것은 내부특성간의 포함관계가 정리되어 있지 않기 때문에 잘못 적용된 내부특성의 메트릭스가 확장 해석된 결과라고 생각할 수 있다. 실제 이들의 잘못된 지적의 약 70%는 확장해석에 의한 잘못된 적용에 의한 것이고 남은 30%는 메트릭스의 의미를 잘못 해석한데서 기인한 것이다.

잘못된 지적을 포함하는 측정은 단순히 메트릭스 값이 잘못 집계된 것만으로 그치는 것이 아니라 측정 대상의 제품의 품질을 잘못 평가해 버리고 말기 때문에 측정 정밀도에 영향을 미치는 것이 된다. 예를 들면, 이번 측정에서 완전성에 관한 메트릭스(CP1.3)에 관한 모든 지적 건수 35건 가운데 16건(약 46%)이 잘못된 지적이었고 이것은 중복을 제외한 완전성의 모든 지적건수(바른 지적 건수 +잘못된 지적 건수)52건의 약 31%에 해당한다. 따라서 잘못된 지적이 완전성의 평가에 크게 영향을 미친다는 것을 확인하였다.

(3)개선 효과

전술한 바와 같이 비계층화된 내부특성에 의해 지적된 오류 건수에 대해 개선된 계층화 메트릭스를 이

〈표 5〉 개선된 메트릭스에 의한 평가 결과
〈Table 5〉 Evaluation result by improved metrics

| 구 분 | 중복 지적 | 잘못된 지적 | 중복 메트릭스율 | 잘못된 메트릭스율 | 평균 |
|---|---|---|---|---|---|
| 지 적 율 | 52% | 30% | 33% | 13% | 32% |
| 개선비율 | 78.6% | 89.5% | 82.5% | 92% | 85.6% |

용하여 4.1절에서 평가한 모델을 재평가한 결과 〈표 5)와 같은 개선 효과를 보았다.

〈표 5)를 보면 내부특성의 비계층화에 의한 중복 지적, 잘못된 지적, 중복 메트릭스율, 잘못된 메트릭스율에 대해 해소 비율이 평균 85.6%로 나타났다.

## 5. 결 론

고품질의 소프트웨어를 개발하기 위해서는 기존에 행해졌던 개발이 완료된 후의 평가 작업이 아닌사용자 요구 정의로부터 개발 완료에 이르기까지 개발 전 공정에 걸쳐 단계별로 품질평가를 하는 방향으로 전환되었다. 이에 따라 고품질, 고신뢰성의 소프트웨어를 개발하기 위해 개발 상위 단계인 설계단계에서부터 설계품질을 측정하고 그것에 기초하여 설계품질을 향상시켜 나가는 방법이 매우 중요하다고 할 수 있다. 그러기 위해서는 품질 메트릭스에 의한 전반적인 품질관리가 이루어져야 하며, 이에 대한 지금까지의 연구는 대부분 사용자측의 평가 척도였고 실제적으로 개발자 입장에서의 평가할 수 있는 내부특성 및 메트릭스에 대한 연구는 아직까지 미비한 상태이다. 본 논문에서는 지금까지 제안되어온 소프트웨어의 내부특성에 대해 살펴보고 이 내부특성이 포함하는 문제점을 지적하고 이것을 해결하기 위한 방법으로 내부특성간의 관계를 정립하여 서로의 연관관계를 계층적으로 구성하는 계층 모델을 제안하였다. 이 모델에 따른 새로운 내부특성을 재정리함으로써 기존의 내부특성이 갖고 있던 문제점인 내부특성의 누락과 중복을 명확하게하여 품질평가 결과의 신뢰성을 좀 더 높일 수 있다고 본다. 따라서 새로운 시각으로 개발자 입장에서 평가할 수 있는 내부특성을 사용자 입장에서 평가하는 품질특성 및 품질부특성과 마찬가지로 계층화를 이루었다. 또한 구체적인 적용실험에서 메트릭스의 중복 지적(28.5%)과 잘못된 지적(16.7%)에 대해 개선한 메트릭스에 의해 측정의 효율성 및 메트릭스의 측정 정밀도에 평균 85.6% 정도의 해소 비율을 얻을 수 있었다.

앞으로의 연구과제는 여러 프로젝트에 적용하여 효과의 일반성을 확인함과 동시에 누락되어 있는 내부특성의 추가, 메트릭스의 중복 및 잘못된 적용에 대하여 지속적인 분석과 실험이 필요하다고 본다.

## 참 고 문 헌


[1] ISO/IEC 9126(Information Technology Software Production Evaluation Quality Charateristics and Guidelines for their use), ISO/IECJTC1 SC7, 1991.

[2] G. E. Murine, "Software Measurement Techniques," ASQC Quality Congress Transactions, 1988.

[3] K. Torii, "A new Framework for Software Quality Assurance", 1st International Workshop on Software Quality Improvement, 1989.

[4] B. W. Boehm, J. R. Brown, M. Lipow, "Quantitative Evaluation of Software Quality", 2nd ICSE, pp. 592-605, 1976.

[5] J. P. Cavano, J. A. McCall, "A Framework for the Measurement", ACM Software Quality Assurance Workshop, pp. 133-139, 1978.

[6] T. Sunazuka, M. Azuma, N. Yamagishi, "Software Quality Assesmant Technology", 8th ISCE, pp. 142-148, 1985.

[7] Evans, M. W., and J. J. Marciniak, "Software Quality Assurance and Management", Wiley-Interscience, 1987.

[8] 三宅 武司, 竹中 市郎, 小田 英雄, "ソフトウェア品質メトリクスの適用", 第 40回 情報處理學會全國大會, pp. 1057-1058, 1990.

[9] "ソフトウェア開發·시스템의 文書化標準化調査研究報告書 STD-WG5", 日本規格協會, 1992.

[10] 三宅武司, 富士仁, 西山茂, "設計レビュ-項目の十分性に關する檢討", 第47回情報處理學會全國大會, 1993.

[11] 김명옥, 양해술 외 2인, "소프트웨어 품질평가 도구 지원을 위한 정량적인 측정모델", 한국정보과학회 가을학술발표논문집 제20권 2호, pp. 867-870, 1993. 10.

[12] 이용근, 양해술, "품질기능 전개를 이용한 소프트웨어 품질보증 체계의 정형화", 한국정보처리학회 춘계학술발표논문집 제 1권 1호, pp. 159-162, 1994. 4.

[13] 양해술, "소프트웨어의 품질평가 체계와 자동화 도구의 개발", 장기기초연구과제 최종보고서, 한국통신연구개발원, 1995. 12.

[14] 양해술, "품질관리 방법론 및 지원도구의 개발", STEP 2000 제1차~3차년도 최종연구보고서, 과학기술처, 1997. 9.

[15] 양해술, "소프트웨어 품질측정 기록 및 지원 툴킷의 개발", 시스템공학연구소 위탁연구과제, 제1차년도 중간보고서, 1997. 6.

[16] 양해술, "한남대학교 종합정보시스템의 품질감리 및 평가", 설계단계 품질 감리 및 평가보고서, 한국S/W품질연구소, 1997. 7.

**양 해 술**

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)

1878년 성균관대학교 정보처리학과 정보처리 전공(석사)

1991년 日本 오사카대학교 기초공학부 정보공학과 소프트웨어공학 전공(공학박사)

1975년~1979년 육군중앙경리단 전자계산실 시스템분석장교

1986년~1987년 日本 오사카대학교 객원연구원

1980년~1995년 강원대학교 전자계산학과 교수

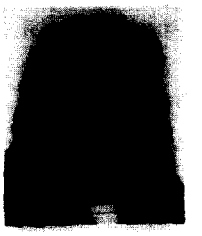
1993년~1994년 한국정보과학회 학회지 편집부위원장

1994년~1995년 한국정보처리학회 논문지편집위원장

1994년~현재 한국산업표준원(KISI) 이사

1995년~현재 한국소프트웨어품질연구소(INSQ) 소장

관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가, 품질감리, 품질컨설팅, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리.



**이 용 근**

1988년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)

1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)

1989년~1992년 강원대학교 전자계산학과 조교

1994년~1995년 한림전문대학 전산정보처리학과 강사

1996년~현재 경희대학교, 경원대학교 공과대학 전자계산공학과 강사

1995년~현재 한국소프트웨어품질연구소 설장/선임연구원

관심분야: 소프트웨어공학(특히, 소프트웨어 품질보증과 품질평가, 품질감리, 객체지향 프로그래밍, 객체지향 분석과 설계 방법)