⊠응용논문

Software Quality Metrics를 이용한 공학용 전산 프로그램의 품질특성 측정

조문성 · 남지희
한국원자력연구소

Quality Evaluation of Engineering Computer Programs Using Software Quality Metrics

Moon-Sung Cho · Ji-Hee Nam
Korea Atomic Energy Research Institute

Abstract

SQM (Software Quality Metrics) is a methodology whose primary objective is the measurement of compliances to requirements using a set of software life cycle properties called quality factors, which is based on the hierarchical relationships between factors, criteria and elements. For this study, two factors (Correctness, Maintainability) and five criteria were selected. In addition, several tens of quality elements were developed to support them. Qualities of three computer programs which are being used for engineering purpose were measured. As a result, it is concluded that SQM is a valuable method for continuously monitoring the pulse of software quality development and that it can be used as a tool for software quality assurance.

1. 서론

소프트웨어 품질보증이란 소프트웨어의 근본적인 결함이나 오류의 잠재적 가능성을 제거하고자 하는 것으로 소프트웨어의 라이프 싸이클 전단계에 적절하고 체계적인 품질보증 기법을 적용하여 소프트웨어 오류의 가능성을 감소시킴으로써 달성할 수 있다.

SQM(Software Quality Metrics)은 소프트웨어 품질을 그 개발과정에 병행하여 계량화(計量化)하는 기법으로서 소프트웨어 라이프싸이클, 즉 요건분석 단계로부터 시험, 설치 단계의 전 과정에 걸친 객관적인 평가를 수행할 수 있게 한다.

SQM의 첫 번째 목표는 '품질요건(Quality Factor)' 이라고 불리는 소프트웨어 라이프 싸이클의 고유 성질을 이용하여 소프트웨어 품질특성들의 만족도를 측정하는 것이다. 이러한 품질요건들의 속성을 명확히 정의하기 위한 시도는 일찍이 1968년 Hartwick과 Rubey에 의해 행해졌고, 그 후 1976년 B. Boehm에 의해 60개의 요건들로 분류되기에 이르렀다. 그 뒤를 이어 1977년에 RADC (Rome Air Development Center)의 후원을 받은 Walters와 McCall에 의해 11개로 추려졌으며 각각의 요건들에 대한 명확한 정의가 확립되었다. Walters와 McCall에 의한 품질요건과 그들에 대한 정의는 <표 1>에 보여진 바와 같다.[Walters, McCall, 1977]

각각의 품질요건은 품질기준(Quality Criteria)이라고 불리는 소프트웨어 속성의 집합에 의해 구체화되며, 각각의 품질기준은 다시 '메트릭 (Metric)'이라고 불리는 수량화할 수 있는 품질요소들에 의해 정량화된다. 그러나 주어진 소프트웨어 시스템의 모든 품질요건에 대한 만족도를 측정하는 것은 아니며, 측정을 위해 선별된 요건들이라 할지라도 그 중요도에 따라 가중치를 다르게 줄 수도 있다. 그리고 라이프 싸이클의 각 단계에서 얻어지는 결과물의 상이함 때문에, 특정요건에 속하는 메트릭들이 항상 일정한 것이 아니라 라이프 싸이클의 단계에 따라 변화하게 된다.

본 논문에서는 1977년 미국 RADC의 연구를 시발점으로 산업계로부터 관심을 끌기 시작한 SQM 기법을 고찰하고 그것을 체계화하여 공학용 컴퓨터 프로그램의 품질특성 측정에 응용함으로써, 소프트웨어 품질보증을 위한 도구로서 SQM 기법이 실용화될 수 있음을 제시하였다.

〈 표 1 〉 Walters와 McCall의 품질요건

요 건	정 의
정확성	프로그램의 시방 만족도 및 사용 목적에 부합되는 정도
신뢰성	요구되는 정밀도로 의도하는 기능을 이행하는 정도
효율성	기능의 이행을 위해 사용되는 자원(Resource)의 최소화 정도
방어성	소프트웨어나 데이터에 대한 비인가자의 접근을 통제하는 정도
재사용성	프로그램이 다른 어플리케이션에 사용되어질 수 있는 정도
사용편의성	프로그램의 조작, 입력데이터 작성 및 결과의 처리를 익히는데 드는 노력의 정도
유지보수성	사용중인 프로그램에 내재된 오류를 발견하고 교정하는데 드는 노력의 정도
테스트용이성	프로그램이 의도된 기능을 이행하는 지를 확인하는데 드는 노력의 정도
신축성	사용중인 프로그램의 수정에 드는 노력의 정도
범용성	프로그램을 다른 하드웨어 및 소프트웨어 환경으로 옮기는데 드는 노력의 정도
호환성	두 개 소프트웨어 시스템의 결합에 필요한 노력의 정도

2. 소프트웨어 품질의 계량기법

2.1 소프트웨어 라이프싸이클의 표준화

소프트웨어의 품질을 계량화하기 위해서는 소프트웨어 라이프싸이클의 표준화가 선행되어야 한다. 라이프싸이클을 표준화함으로써 소프트웨어 시스템의 체계적인 개발 및 사용이 가능할 뿐만 아니라, 각 단계에서의 입수 가능한 측정대상 자료들을 요건분석 단계로부터 사용단계에 이르는 라이프싸이클 전 단계에 걸쳐 표준화할 수 있기 때문이다.

소프트웨어 라이프싸이클은 소프트웨어의 규모나 과정의 복잡성 정도에 따라 부분적으로 수정될 수 있어야 한다. 그것은 평가대상 자료들이 소프트웨어에 따라 달라질 수 있다는 것을 의미한다.

일반적인 라이프싸이클 단계는 다음과 같다.[Bryant, Wilburn, 1987]

① 요건분석 단계
② 기능분석 단계
③ 상세 설계 단계
④ 코딩 및 소프트웨어 생성 단계
⑤ 시험, 설치 및 인증 단계
⑥ 사용 및 유지 보수 단계

요건분석 단계는 컴퓨터 프로그램이 필수적으로 만족시켜야 하는 요구사항들을 식별하는 단계로서 요건시방서가 주요 결과물이다.[Wilburn, 1983] 이 단계는 최종제품의 품질에 가장 큰 영향을 미치는 단계로서 심각한 오류들을 요건분석 단계에서 발견하여 교정함으로서 재작업, 재분석, 또는 재기획 등을 사전에 막을 수 있다.[Deutsch, 1982]

기능분석 단계에서는 소프트웨어 설계시방서를 생산하며 소프트웨어가 어떤 방법으로 요구사항을 만족시킬 것인지를 결정하여야 한다. 이 단계에서는 시스템을 기능에 따라 여러 부분으로 구분하여 각 부분이 설계시방서에 명시된 기능들을 될 수 있는 한 독립적으로 설계될 수 있도록 해야 한다.[Dunn, Ullman, 1982]

상세 설계는 요건시방서에 명시된 기능들을 논리적으로 더욱 세분화하는 것이다. 상세 설계는 연산 및 방정식의 정의, 세부 논리 및 소프트웨어 내에서 수행되어지는 데이터 운용 등을 포함하여야 한다. 이 단계에서는 뒤따르는 실행 단계를 위한 개념적인 해답 즉 청사진을 제공하게 된다. 이 단계에서 생산되는 주요 결과물은 상세설계시방서이다.

코딩 및 소프트웨어 생성 단계에서는 소프트웨어의 상세 설계 내용이 고급 또는 어셈블리 프로그램 언어로 번역되어지는 단계이다. 컴파일 및 어셈블리 에러가 수정되고 우선적으로 명백한 오류를 제거하기 위해 각각의 프로그램 모듈을 실행함으로써 프로그램의 예비 점검이 수행된다. 이 단계의 결과물은 프로그램 리스트로서 컴퓨터가 읽을 수 있고 처리할 수 있는 최초의 산물이다.

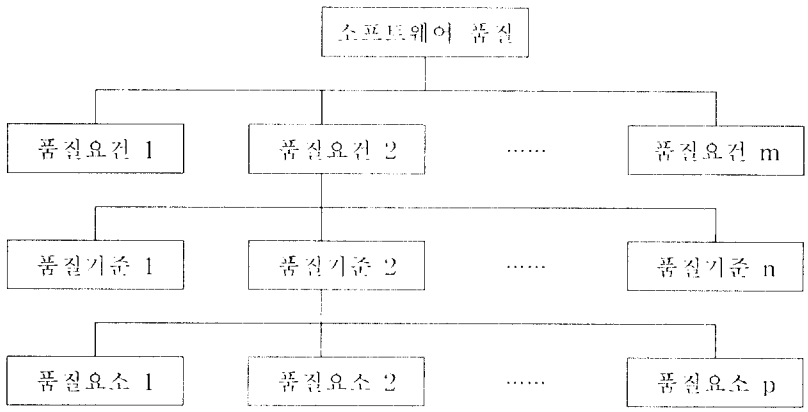
시험, 설치 및 인증 단계에서는 개발자에 의한 최종 테스트와 소프트웨어의 설치 및 인증시험이 수행되고 그 결과에 따라 소프트웨어가 인증된다. 시험은 모든 요건들이 만족되었으며 시험 계획에 따라 수행되었는지를 결정하기 위한 것이다. 시험의 결과는 시험 및 검증보고서로 작성되어 평가되어야 한다. 개발자에 의해 시험 및 설치가 완료되면 최종적인 인증을 위해 제 3의 독립된 조직에 의한 인증시험(Acceptance Test)이 실시되어야 한다.[6]

마지막으로 사용 및 유지 보수 단계는 소프트웨어 시스템을 실제로 사용하고 유지, 보수하는 단계이다. 유지란 본래의 요건사양서에서 요구하는 대로 프로그램 성능상의 오류를 수정한다거나 혹은 프로그램의 기능을 향상시키기 위하여 소프트웨어에 변화를 주는 것이다. 프로그램의 변화는 피할 수 없는 것이기 때문에 소프트웨어의 개정판 관리를 효율적으로 수행하기 위한 계획이 확립되어야 한다.[7]

2.2 SQM 체계[Murine, 1985]

독립된 소프트웨어 시스템이나 그 일부분을 구성하는 하부 시스템은 몇 개의 선택된 품질요건들에 대해 그 품질이 측정된다. 품질요건들은 몇 가지 품질기준에 의해 정의되는데, 이들 품질기준들은 각 품질기준들에 대해 고유한 것은 아니다. 그리고 각각의 품질기준은 측정 가능한 다수의 품질요소에 의해 정의된다<그림 1 참조>.

품질요소 뿐만 아니라 각 개의 품질요소에 대해 측정된 값 또한 메트릭이라고 하는데 이 메트릭이라는 용어는 각 개의 품질요건이나 품질기준에 대해 계량된 값을 의미하기도 한다. 그리고 품질요건의 선정은 라이프싸이클의 각 단계와 무관하게 이루어지나 품질기준 및 그와 관련된 품질요소를 선정하는 일은 그렇지 않다. 즉, 특정 품질요건과 관련된 품질기준의 선택은 소프트웨어 개발 단계에 따라 변할 수 있는 것이다.



〈 그림 1 〉 소프트웨어 품질측정체계

2.3 소프트웨어 품질요건의 선정

품질요건들은 보통 특정 소프트웨어 시스템과 관련되어 결정되어지며 SQM 체계에 있어서 유일하게 소프트웨어 라이프싸이클 단계와는 무관한 독립된 인자이다. 특정 소프트웨어 시스템의 측정을 위해 모든 요건이 적용되는 것은 아니며 품질요건의 적용에 따르는 비용, 시간 등을 고려하여 결정하여야 한다.

품질요건을 선정하는 방법에는 여러 가지가 있다.[Murine, 1986] 첫 번째 방법은 위에서 기술한 사항을 고려하여 체크리스트나 규격요건(Standard Requirements)에 따른 리스트 등을 이용하여 분석의 시작 단계에서 임의로 선택하는 방법이다. 두 번째 방법은 각 품질요건의 중요도를 고려한 방식으로서 다음과 같이 각각의 품질요건에 품질계수를 부여하는 것이다.

$$S_q = A_{1,1}q_1 + A_{1,2}q_2 + A_{1,3}q_3 + \cdots\cdots + A_{1,n}q_n = \sum_{j=1}^{n} A_{1,j}q_j$$

여기서 $A_{1,j}$는 해당 품질요건의 가중치를 반영키 위한 계수이며 q_j는 품질요건이다. 예를 들면, q_1 = 정확성(Correctiveness), q_2 = 신뢰성(Reliability) 등이며, S_q는 뒤에 설명될 '품질 점수(Quality Score)'를 나타낸다. 여기서 사용된 2개의 아래 첨자는 나중에 설명될 품질기준과의 관계를 보여주기 위함이다. 이러한 방식에서는 적용될 모든 품질요건에 0이 아닌 계수가 주어지며 동등한 중요도를 가진 품질요건에는 균일한 값의 품질계수가 주어진다. 만약 11개의 품질요건이 동등한 중요도를 가진다면

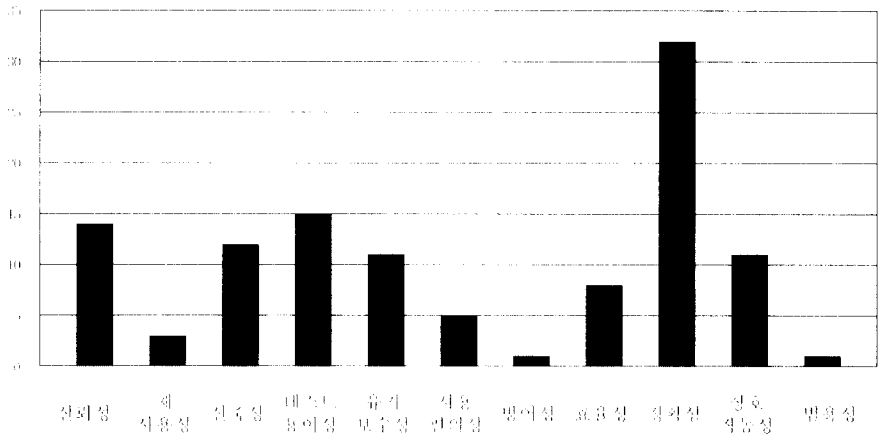
$$S_q = \sum_{j=1}^{11} q_j$$

가 되고, 혹은 정확도가 다른 품질요건에 비해 2배의 가중치를 가진다면

$$S_q = 2q_1 + \sum_{j=2}^{11} q_j$$

가 될 것이다.

세 번째 방법은 '일반투표(Popular Vote)' 방식으로서, 이 방식은 요건시방서(Requirement Specification)에 기술된 요건들을 <그림 2>와 같이 11개의 품질요건들과 관련지어 분류하고 가장 많은 요건들을 포함하는 품질요건들을 선택하는 방식이다.



< 그림 2 > 요건시방서에 나타난 품질요건별 중요도

2.4 SQM 품질기준의 선정

특정 소프트웨어 시스템에 고유한 품질요건이 선정되면 그에 따른 품질기준의 선택이 이루어져야 한다. 각 품질요건을 정의하는 품질기준들은 라이프싸이클 단계에 따라 달리 선정될 수 있으나 일반적인 관계를 나타내면 <표 2>와 같다.[Murine, 1988]

< 표 2 > 품질요건과 품질기준의 연계

품질요건 (Factor)	품질기준 (Criteria)	품질요건 (Factor)	품질기준 (Criteria)
정확성 (Correctness)	추적용이성 (Traceability) 완벽성 (Completeness) 일관성 (Consistency)	테스트용이성 (Testability)	모듈성 (Modularity) 단순성 (Simplicity) 자기서술성 (Self descriptiveness)
신축성 (Flexibility)	모듈성 (Modularity) 확장성 (Expandability) 자기서술성 (Self descriptiveness)	신뢰성 (Reliability)	일관성 (Consistency) 정확성 (Accuracy) 단순성 (Simplicity) 에러허용성 (Error Tolerance)
범용성 (Portability)	모듈성 (Modularity) 자기서술성 (Self descriptiveness) 장치독립성 (Hardware Independence)	효용성 (Efficiency)	실행효율성 (Execution Efficiency) 저장효율성 (Storage Efficiency)
방어성 (Integrity)	접근통제성 (Access Control) 접근감시성 (Access Audit)	유지보수성 (Maintainability)	일관성 (Consistency) 모듈성 (Modularity) 단순성 (Simplicity) 자기서술성 (Self descriptiveness) 간결성 (Conciseness)
사용편의성 (Usability)	훈련용이성 (Training) 통신용이성 (Communicativeness) 조작용이성 (Operability)		
재사용성 (Reusability)	모듈성 (Modularity) 장치독립성 (Hardware Independence) 자기서술성 (Self descriptiveness)	호환성 (Inter Operability)	모듈성 (Modularity) 통신호환성 (Communication Commonality) 데이터호환성 (Data Commonality)

2.5 SQM품질요소의 선정

품질요소의 선정은 SQM 적용 전 과정에 있어서 가장 중요한 일이다. 품질요소의 선정은 요소의 측정대상 뿐만 아니라 관련된 품질기준, 품질요건 및 컴퓨터 프로그램 구성 부분들의 품질을 측정키 위한 밑바탕이 된다. 품질요소를 선정하는 데 고려하여야 할 사항은 다음과 같다.

품질요소는 측정 가능한 데이터의 입수 가능성(Availability)을 충분히 고려하여 선택되어야 한다. 예를 들면, 요건분석단계에서의 주된 데이터는 요건시방서이며 코딩단계에서의 측정대상은 프로그램 리스트인 것이다. 그 외에도 대상 소프트웨어의 사용 유형, 목적 및 개발시 적용 규격 등이 모두 고려되어야 한다.

품질측정체계(SQM Hierarchy)에 있어서 품질요소가 차지하는 위상은 품질요소로 하여금 측정 대상 소프트웨어에 따른 조절 기능을 수행케 한다. 이러한 기능은 SQM의 전 과정에 걸친 재작업 없이도 측정 대상에 따른 분석의 수정을 가능케 하는 것이다. <표 3>은 임의의 품질기준인 '완벽성(Completeness)'을 정의하는 품질요소들을 요건분석단계, 상세설계단계 및 코딩단계로 분리 열거한 예이다.

〈 표 3 〉 '완벽성'을 정의하는 단계별 품질요소

단 계	품 질 요 소	단 계	품 질 요 소
요구분석 Requirement Analysis	- 모든 요구되는 기능은 명확하여야 한다. - 모든 참조기능은 반드시 정의되어야 한다. 의사결정점에서는 모든 가능한 경우가 고려되어야 한다. - 모든 데이타 항목의 단위는 일관되어야 한다.	상세설계	- 모든 데이타의 단위는 일관되어야 한다.
상세설계 Detailed Design	- 모든 기능은 어플리케이션에서 사용되어야 한다. - 호출 및 피호출 프로그램 인수 수는 동일하여야 한다. - 정의되어진 모든 파라미터는 모두 수행되어야 한다.	코딩 Coding	- 모든 데이터는 정의되거나, 계산되거나, 혹은 외부의 소스로부터 얻어져야 한다. 모든 데이터 항목은 반드시 사용되어야 한다. - 입력 파라미터는 반드시 고유하게 식별될 수 있어야 한다. - 코딩은 반드시 설계와 일치하여야 한다. - 모든 기능은 명확하여야 한다. - 모든 기능은 어플리케이션에서 사용되어야 한다.

2.6 품질의 계량 및 결과의 해석

주어진 시스템의 평가결과는 상당히 많은 양의 데이터가 되므로, 그 기록과정은 그러한 데이터를 수용할 수 있을 뿐만 아니라 각 품질요소의 점수를 파악함으로써 관련 품질기준이나 품질요건의 점수가 쉽게 산출되도록 기획되어야 한다. 이러한 목적으로 사용되어질 수 있는 것으로 정오표(Talley Sheet)가 있다. 그리고 정오표에 수록된 데이터를 메트릭 점수표(Metric Score Sheet)에 기록함으로써 품질기준의 점수가 얻어

진다. 메트릭 점수표는 품질요소 차원에서의 감사 결과를 묶고 요약하여 특정한 품질기준의 점수를 제시하게 되고, 이들 품질기준들의 점수에 근거하여 품질요건의 점수가 계산되어 진다.

2.6.1 직접비에 의한 해석(Direct Ratio Method)

이 방식은 품질기준의 각 '사상(Event)'에 대한 '만족(Compliance)'의 비를 구하는 것인데, 여기서 품질기준의 사상의 값은 모든 품질요소 측정치 중 분모의 단순 합이고, 만족의 값은 분자의 단순 합이다. 이를 식으로 나타내면 다음과 같다.

품질요건의 점수는

$$u(F_i) = \left(\sum_{j=1}^{m} n_{i,j}\right) / \left(\sum_{j=1}^{n} \delta_{i,j}\right)$$

여기서 $n_{i,j} / \delta_{i,j}$는 품질요건을 구성하는 품질기준의 점수 $u(C_{i,j})$를 나타내며 다음과 같이 구해진다.

$$u(C_{i,j}) = \left(\sum_{k=1}^{p} n_{i,j,k}\right) / \left(\sum_{k=1}^{p} \delta_{i,j,k}\right)$$

여기서 우항은 i 품질요건의 j 품질기준에 속하는 k 품질요소의 측정치

$$e_{i,j,k} = n_{i,j,k} / \delta_{i,j,k}$$

의 합이다.

이 방식은 각 품질요소의 특징만을 고려할 때에는 적합한 것이지만 품질기준의 비로 확장될 때에는 사상수의 차이에 따른 불균형이 고려되지 않는다. 이러한 이유 때문에 다음과 같은 방식들이 요구되는 것이다.

2.6.2 2차 평균에 의한 해석(Second Order Averaging Method)

이 방식은 각 품질기준의 해당 품질요건에 대한 기여도가 같으며 또한 각 품질요소 역시 해당 품질기준에 대한 기여도가 같다는 가정 하에서 적용되어질 수 있다. 이 방식을 식으로 나타내면 다음과 같다.

$$u'(F_i) = 1/m \sum_{j=1}^{m} u'(C_{i,j})$$

여기서

$$u'(C_{i,j}) = 1/p \sum_{k=1}^{p} e_{i,j,k}$$

이다.

2.6.3 가중 2차 평균에 의한 해석(Second-Order Averaging Method, Weighted)

이 방식은 품질요소, 기준 및 요건 각각의 중요도가 일정치 않을 경우, 각 요소, 기준 및 요건에 가중치를 부여함으로써 그 중요도를 반영키 위한 해석 방식이다. 이 방식을 식으로 나타내면 다음과 같다.

$$u''(F_i) = 1/\beta \sum_{j=1}^{m} b_{i,j} \cdot u''(C_{i,j})$$

이고

$$u''(C_{i,j}) = 1/\alpha \sum_{k=1}^{p} a_{i,j,k} \cdot e_{i,j,k}$$

이다. 여기서

$$\beta = \sum_{j=1}^{m} b_{i,j} \qquad \alpha = \sum_{k=1}^{p} a_{i,j,k}$$

이다.

3. SQM의 적용

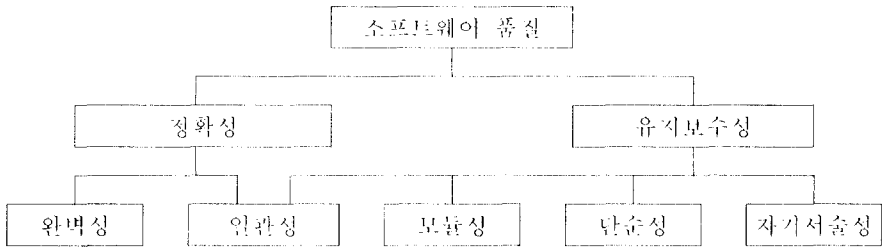
SQM의 실제 적용을 위해 3개의 공학용 전산 프로그램을 선택하였다. 이 프로그램들은 모두 라이프싸이클 제 5단계인 '사용 및 유지 보수 단계'에 있는 코드들로서 각각 A, B 및 C 라는 이름으로 칭하기로 한다. 이 프로그램들의 품질특성 측정을 위하여 2 가지의 품질요건을 선정하였다. 그것들은 정확성(Correctness)과 유지보수성(Maintain-ability)으로서 RADC에 의한 정의는 <표 1>과 같다.

이들 코드와 관련하여 입수 가능한 측정 자료는 프로그램 리스트이므로 코딩 및 소프트웨어 생성 단계에 적용할 수 있는 5가지의 품질기준을 선정하였으며 그 정의는 <표 4>와 같다. 선정된 품질기준과 품질요건의 관계는 <그림 3>과 같다.

〈 표 4 〉 SQM 적용을 위한 5가지 품질기준

품 질 기 준	정 의
완벽성 (Completeness)	요구되는 기능 구현의 완벽성
일관성 (Consistency)	소프트웨어의 설계, 구현 및 표기의 일관성
단순성 (Simplicity)	소프트웨어 기능 구현 방법상의 단순성
모듈성 (Modularity)	소프트웨어를 구성하는 모듈의 독립성
자기서술성 (Self descriptiveness)	소프트웨어 기능 구현에 대한 설명의 자세함

품질기준을 정의하고 계량화하기 위하여 40개의 품질요소를 개발하였다. 이들은 미 국방사규격 DOD-STD-2167A 및 NUREG에서 발행한 SQA Handbook Appendix. C로부터 추출된 것들로서 본 과제를 위해 선택된 5개의 품질기준과 관련지어 분류하였다.[11, 12] A, B 및 C의 3개 모드의 프로그램 리스트를 측정 대상으로 하여 품질요소 하나하나에 대한 검사를 수행하였다.



〈 그림 3 〉 품질요건과 품질기준의 연계

3.1 프로그램 A의 품질특성 측정

프로그램 A는 라이프 싸이클 제 5단계인 사용 및 유지 보수 단계에 있는 프로그램으로서 주 프로그램과 3개의 부프로그램으로 구성되어 있다. 소스 리스트를 대상으로 모두 1317개의 사상(Events)을 체크하였다. 각 메트릭에 대한 측정치를 <표 5>와 같이 만족/사상(Compliance/Events)과 같이 계산하여 소수로 나타내었으며 이들 측정치는 가중 2차 평균 방식으로 계산되었다.

〈 표 5 〉 SQM 메트릭에 대한 측정예

CP=Completeness		
CP 101	모든 외부 입력 및 그 소스는 정의되어야 한다.	1.00
CP 102	모든 외부 입력 및 그 형식은 정의되어야 한다.	1.00
CP 103	모든 외부 입력 및 그 입력 방법이 정의되어야 한다.	1.00
CP 104	모든 내부 입력 및 그 소스는 정의되어야 한다.	1.00
CP 105	모든 내부 입력 및 그 형식은 정의되어야 한다.	1.00
CP 106	모든 내부 입력 및 그 입력 방법이 정의되어야 한다.	1.00
CP 107	모든 외부 출력 및 그 의미는 정의되어야 한다.	1.00
CP 108	모든 외부 출력 및 그 대상이 정의되어야 한다.	1.00
CP 109	모든 서브루우틴 입력의 기능이 정의되어야 한다	0.00
CP 110	모든 서브루우틴 출력의 기능이 정의되어야 한다	0.00
CS=Consistency		
CS 201	연산기능의 변경없이 수정될 수 있는 파라미터들은 식별되어야 한다.	0.00
CS 202	형식 파라미터는 각 부프로그램의 첫머리에 위치하여야 한다.	1.00
CS 203	명명 규칙(Naming Convention)은 전체 프로그램에 걸쳐 일정하여야 한다.	1.00
CS 204	수치적 규칙(Numerical Convention)은 전체 프로그램에 걸쳐 일정하여야 한다.	1.00
MO=Modularity		
MO 301	프로그램은 오직 3가지 기본적인 제어문만을 사용하여야 한다. : 순차처리, If, Do.	0.95
MO 302	모듈을 구성하는 루틴들은 루틴당 실행문의 평균이 100개를 넘거나는 안되고 최대 실행문의 수가 200개를 넘거나는 안된다.	1.00
MO 303	프로그램은 재배치 가능한 복적 모듈로 구성되어야 한다.	1.00
MO 304	프로그램은 최적제어된 하향식 구조로 작성되어야 한다.	1.00
SI=Simplicity		
SI 401	모든 루틴은 각각 한 개의 입구와 출구를 가져야만 한다.	0.67
SI 402	GOTO문은 같은 루틴에 있는 명령문으로만 제어를 옮겨야 한다.	1.00
SI 403	GOTO문은 오직 앞으로만 제어를 옮겨야 한다.	0.82
SI 404	값의 공통 소스를 사용함으로써 형식 파라미터의 중복을 최소화하여야 한다.	0.50

SD Self Descriptiveness		
SD 501	중복이 필요할 경우, 파라미터의 식별을 위한 기호 설명이 있어야 하며, 중복 위치는 주석(Comments)에 밝히야 한다.	0.00
SD 502	모든 인수의 정의와 위치는 주석을 통해 설명되어야 한다.	0.00
SD 503	루틴명은 그 기능을 식별할 수 있도록 고유하게 선택되어야 한다.	1.00
SD 504	루틴명은 소적체계성 소프트웨어의 다른 구성요소(Component)와의 관계를 식별할 수 있도록 고유하게 선택되어야 한다.	0.00
SD 505	소프트웨어 각 구성요소의 기능은 시종적으로 서술되어야 한다.	1.00
SD 506	각 구성요소는 모듈의 첫머리에 그 입력자료에 대한 서술을 포함하여야 한다.	1.00
SD 507	각 구성요소는 모듈의 첫머리에 그 출력자료에 대한 서술을 포함하여야 한다.	0.00
SD 508	각 구성요소는 모듈의 첫머리에 그 기능에 대한 서술을 포함하여야 한다.	1.00
SD 509	각 구성요소는 모듈의 첫머리에 그 알고리즘에 대한 서술을 포함하여야 한다.	1.00
SD 510	각 구성요소는 모듈의 첫머리에 그 호출 구성요소에 대한 서술을 포함하여야 한다.	N/A
SD 511	각 구성요소는 모듈의 첫머리에 그 피호출 구성요소에 대한 서술을 포함하여야 한다.	N/A
SD 512	각 루틴을 설명하는 요약문에는 이해를 높이 위한 일반적 설명외에도 관련 문장 데이터에 대한 정확한 서술이 포함되어야 한다.	1.00
SD 513	각 루틴을 설명하는 요약문에는 이해를 높이 위한 일반적 설명외에도 관련 데이터명 데이터에 대한 정확한 서술이 포함되어야 한다.	1.00
SD 514	요약문에는 입력값의 허용범위가 명시되어야 한다.	0.00
SD 515	요약문에는 출력값의 허용범위가 명시되어야 한다.	0.00
SD 516	원래의 프로그래머와 개정작업을 수행한 프로그래머의 이름 및 작업내용, 회사명 (부서 포함) 및 작업완료일자가 포함되어야 한다.	0.00
SD 517	프로그램 전체에 걸쳐 주석이 사용되어야 한다.	1.00
SD 518	소스분은 그 자체로 혹은 주석에 의해, 원래의 개발 작업에 참여하지 않은 사람도 이해할 수 있도록 작성되어야 한다.	1.00

$$u(C_{1,1}) = 1/10 \sum_{k=1}^{10} e_{1,1,k} = 8/10 = 0.8$$

$$u(C_{1,2}) = u(C_{2,2}) = 1/4 \sum_{k=1}^{4} e_{1,2,k} = 3/4 = 0.75$$

$$u(C_{2,3}) = 1/4 \sum_{k=1}^{4} e_{2,3,k} = 3.95/4 = 0.99$$

$$u(C_{2,4}) = 1/4 \sum_{k=1}^{4} e_{2,4,k} = 2.99/4 = 0.75$$

$$u(C_{2,5}) = 1/16 \sum_{k=1}^{16} e_{2,5,k} = 9/16 = 0.56$$

여기서 $e_{i,j,k}$는 품질요건 F_i에 속하는 품질기준 C_j의 k번째 품질요소의 점수를 나타내며 $u(C_{i,j})$는 품질요건 F_i에 속하는 품질기준 C_j의 점수를 뜻한다. 따라서 $u(C_{1,2})$와 $u(C_{2,2})$의 점수는 품질요건 F_1과 F_2에 공통으로 속하는 품질기준 C_2의 점수를 나타내는 것이므로 당연히 같은 값이다. 이렇게 구해진 품질기준의 값에 근거하여 다음과 같이 품질요건의 점수가 구해진다.

$$u(F_1) = 1/2 \sum_{j=1}^{2} u(C_{1,j}) = [u(C_{1,1}) + u(C_{1,2})] \times 1/2 = 0.78$$

$$u(F_2) = 1/4 \sum_{j=2}^{5} u(C_{2,j}) = [u(C_{2,2}) + u(C_{2,3}) + \cdots + u(C_{2,5})] \times 1/4 = 0.76$$

이렇게 하여 얻어진 품질요건 및 품질기준의 점수를 요약하면 <표 6>과 같다.

〈 표 6 〉 컴퓨터 프로그램 A의 품질측정 결과

프로그램 C 평가결과 : 0.77			
정확성	0.78	유지보수성	0.76
		일관성	0.75
완벽성	0.80	단순성	0.75
일관성	0.75	모듈성	0.99
		자기서술성	0.56

3.2 프로그램 B의 품질특성 측정

프로그램 B는 라이프 싸이클 제 5단계인 사용 및 유지 보수 단계에 있는 프로그램으로서 주 프로그램과 3개의 부프로그램으로 구성되어 있다. 소스 리스트를 대상으로 모두 1223개의 사상(Events)을 체크하였다. 각 요소에 대한 측정치를 만족/사상(Compliance/Events)의 형태로 구하였으며, 앞에서와 같이 가중 2차 평균 방식을 이용하여 5가지 품질기준의 점수를 계산하였다. 프로그램 B의 품질요건 및 품질기준의 점수를 요약하면 <표 7>과 같다.

〈 표 7 〉 컴퓨터 프로그램 B의 품질측정 결과

프로그램 B 평가결과 : 0.66			
정확성	0.72	유지보수성	0.60
		일관성	0.64
완벽성	0.80	단순성	0.65
일관성	0.64	모듈성	0.97
		자기서술성	0.13

3.3 프로그램 C의 품질특성 측정

프로그램 C 역시 라이프 싸이클 제 5단계인 사용 및 유지 보수 단계에 있는 프로그램으로서 주 프로그램과 7개의 부프로그램으로 구성되어 있다. 소스 리스트를 대상으로 모두 4385개의 사상(Events)을 체크하였다. 각 요소에 대한 측정치를 만족/사상(Compliance/Events)의 형태로 구하였으며, 앞에서와 같이 가중 2차 평균 방식을 이용하여 5가지 품질기준의 점수를 계산하였다. 프로그램 C의 품질요건 및 품질기준의 점수를 요약하면 <표 8>과 같다.

〈 표 8 〉 컴퓨터 프로그램 C의 품질측정 결과

프로그램 A 평가결과 : 0.71			
정확성	0.74	유지보수성	0.68
		일관성	0.67
완벽성	0.80	단순성	0.98
일관성	0.67	모듈성	0.99
		자기서술성	0.06

3.4 결과의 고찰

SQM 기법을 응용하여 프로그램 A, B 및 C의 품질을 측정한 결과, 표 6, 7 및 8로 요약된 바와 같이 다른 프로그램에 비해 프로그램 A의 품질이 가장 우수한 것으로 판정되었다. 그 주된 요인은 동 프로그램의 '유지보수성'이 다른 프로그램에 비해 뛰어나기 때문인데, 그것은 결국 유지보수성의 평가를 결정짓는 4가지 품질기준 중 '자기서술성'의 비교 우위에 기인한 것으로 분석된다.

앞 절에서 정의된 바와 같이, 자기서술성의 점수가 높다는 것은 프로그램 혹은 프로그램을 구성하는 각 부분의 기능에 대한 서술이 상세하다는 것을 뜻하며, 그것은 곧 사용 중 발생되는 오류를 발견하고 수정하는 데 드는 노력이 상대적으로 적게 든다는 것을 의미한다.

이러한 관점에서 볼 때, B 및 C 코드는 완성 후의 유지 및 보수에 대한 배려가 거의 없이 개발된 코드임을 알 수 있으며, 여러 다른 코드에서도 공통적으로 찾아 볼 수 있는 이러한 경향은 프로그램의 자기서술성이 높다는 사실이 프로그램 기능의 향상과는 무관하다는 인식에서 비롯된 것으로 추측된다.

그러나 아무리 잘 만들어진 프로그램이라 할 지라도 사용 단계에서 오류가 발견될 수 있으며, 방법론의 개선이라던가 모듈의 호환성 등을 고려한다면 자기서술성의 중요성을 간과해서는 안될 것이다. 이러한 사실은 오류 교정에 따른 비용을 생각할 때 더욱 그러하다.

4. 결론

미국 Rome Air Development Center에 의해 정의된 품질요건과 품질기준, 그리고 본 과제 수행을 위해 개발된 다수의 품질요소들을 토대로 Software Quality Metrics 기법을 응용하여 공학용 컴퓨터 프로그램인 A, B 및 C 프로그램의 품질을 측정하여 이들에 대한 객관적 평가를 수행함으로써 공학용 전산 프로그램의 품질보증을 위한 도구로서 SQM 기법이 실용화될 수 있음을 제시하였다.

그러나 동 기법이 실제로 이용되기 위해서는 다음과 같은 점들이 선결되어야 할 것으로 사료된다. 첫째, 라이프 싸이클 각 단계에 대한 요건의 구체화, SQM적용을 위한 책임 및 절차의 수립 등, 소프트웨어 품질보증을 위한 프로그램이 확립되어야 한다. 둘째, 라이프 싸이클 각 단계에 따른 SQM 품질요소들이 충분히 개발되어야 한다. 이것은 동 기법의 유효성과 직결되는 것으로 관련 전문가들의 도움이 절대적으로 필요하다 하겠다.

참고문헌

[1] Walters, G.F. and McCall, J.A.(1977), *Factors in Software Quality*, RADC TR-77-369.

[2] Bryant, J.L. and Wilburn, N.P.(1987), *Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry*, NUREG/CR-4640, PNL-5784, pp. 3.1-3.7.

[3] Wilburn, N.P.(1983), *Guidelines - Software Verification*, HEDL-TC-2425, Westinghouse-Hanford Co., Washington.

[4] Deutsch, M.S.(1982), *Software Verification and Validation - Realistic Approaches*, Prentice-Hall Inc., New Jersey.

[5] Dunn, R. and Ullman, R.(1982), *Quality Assurance for Computer Software*, McGraw

Hill Co., New York

[6] IEEE(1983), *IEEE Standard for Software Test Documentation*, IEEE Standard 829-1983, Institute of Electrical and Electronics Engineers, Inc., New York

[7] IEEE(1983), *IEEE Standard for Software Configuration Management Plan*, IEEE Standard 829-1983, Institute of Electrical and Electronics Engineers, Inc., New York

[8] Murine, G.E.(1985) "Life Cycle Objective Measuring of the Software Product," Measurement Science Conference, Santa Clara, CA

[9] Murine, G.E.(1986) "Using Software Quality Metrics as a Tool for Independent Verification and Validation," IEEE Phoenix Conference on Computers and Communications

[10] Murine, G.E.(1988) "Integrating Software Quality Metrics with Software QA," Quality Progress

[11] MIL Standard(1988), *Defense System Software Development*, Military Standard, DOD-STD-2167A

[12] Bryant, J.L. and Wilburn, N.P.(1987), *Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry*, NUREG/CR-4640, PNL-5784