

소프트웨어 품질관리 방법에 관한 연구

임대혁* · 장영숙**

〈 목 차 〉

- | | |
|-------------------------|---------------|
| I. 서론 | 5. 매트릭스 선정 |
| II. 소프트웨어 품질관리 이론적 고찰 | 6. 평점수준 설정 |
| 1. 소프트웨어 | 7. 품질측정 |
| 2. 소프트웨어 품질 | 8. 평점 |
| 3. 소프트웨어 특수성 | 9. 종합평가 |
| 4. 선진외국의 소프트웨어 품질관리실태 | 10. 분석 및 시정조치 |
| III. 소프트웨어 품질관리 방법 및 기법 | IV. 결론 |
| 1. 품질 요구사항 정의 | 참고문헌 |
| 2. 품질평가 계획 | Abstract |
| 3. 품질목표 설정 | |
| 4. 가중치 설정 | |

I. 서론

세계 각국은 다가오는 지식 정보사회에서 경제 패권을 잡기 위한 핵심요소가 정보화임을 인식하고 다양한 추진 전략을 마련하여 일을 추진하고 있다. 농업혁명과 산업혁명 이후 제3의 혁명으로 간주되고 있는 정보화를 통한 '디지털 혁명'은 이제 거부할 수 없는 커다란 흐름으로 다가오고 있으면, 이미 세계 경제 질서는 정보화를 통해 재편되고 있는 실정이다. 또한 한국전산원에 의하면 우리나라 정보화 수준은 '88년의 경우 미국의 9.8배, 유럽 선진국들의 8.6배, 일본 5.3배, 싱가포르 8.6배, 대만 2배였으나 매년 발전을 거듭하여 1996년 이르러서는 미국이 우리나라의 7.6배, 유럽 선진국들은 4.9배, 일본 4.2배, 싱가포르 4.9

* 해천대학 경영정보전공 교수

** 신성대학 품질경영과 교수

배, 대만 0.8배 수준으로 나타나 선진국과의 정보화 수준격차가 점점 좁혀 졌으며 대만보다는 오히려 앞서고 있는 것으로 나타났다.

이에 편승하여 3A, 즉 공장 자동화(FA: Factory Automation), 사무자동화(OA: Office Automation), 가정 자동화(HA: Home Automation)의 진행이 가속화되어감에 따라 소프트웨어가 차지하는 비중이 날로 급증하고 있는 것도 사실이다. 그러나 이러한 소프트웨어에 대한 품질관리체계 대해서는 선진 외국에 비해 아주 미흡한 실정에 놓여 있다. 사용자가 만족할 수 있는 고품질의 소프트웨어 생산을 위해서는 품질관리의 원칙적인 사고방식 즉, 크로스비(P.B. Crosby)가 말한 “제품을 처음부터 잘 만들어라(do it right the first time)” 혹은 화이겐바움(A.V. Feigenbaum)이 강조한 “제작공정에 품질을 주입하라(build quality into process)”라는 개념으로 소프트웨어 개발주기(Development life cycle) 전 단계에 걸쳐서 품질관리 활동을 하지 않으면 안 된다.

선진 외국의 경우 특히 미 국방성 (DoD : Department of Defense)에 납품되는 모든 소프트웨어에 대해서는 정해진 규격과 절차에 따라 품질관리활동을 엄격히 실시하고 있다. 일본의 경우는 소프트웨어 품질관리를 눈으로 보이게끔 가능한 한 정량적인 척도로 나타내고자 하는 것이 특징이라 볼 수 있다.

따라서 본 논문에서는 이러한 선진 외국의 소프트웨어 품질관리 체계를 연구하여 우리나라 실정에 맞는 소프트웨어 품질관리 방법 및 기법을 제시하고자 한다. 연구대상은 소프트웨어회사, 컴퓨터 메이커 및 해당 조직에서 자체 개발하는 불특정 다수가 사용하는 응용소프트웨어(application software)로 한정하였다.

구성 및 전개는 2장에서 소프트웨어 품질관리에 대한 관련 연구를 살펴보고, 3장에서는 구체적인 방법 및 기법을 제시 하였다. 4장에서는 연구의 결과와 향후 연구과제에 대해서 언급했다.

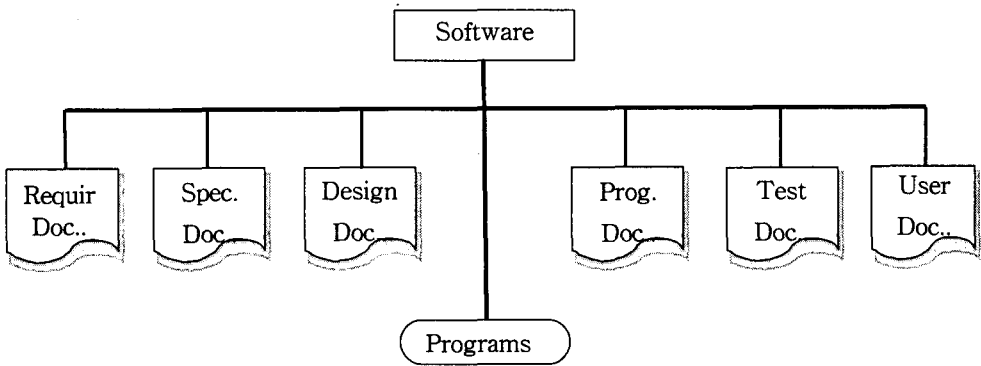
II. 소프트웨어 품질관리 이론적 고찰

1. 소프트웨어(S/W : Software)

소프트웨어라 함은 프로그램과 이를 작성하기 위하여 사용된 설계서·기술서 기타 관련 자료를 말한다. ISO/IEC 9126에 서는 “컴퓨터 프로그램과 절차, 규칙, 그리고 컴퓨터 시스템의 운용에 관련된 문서와 자료”를 소프트웨어라고 정

의하고 있다.

Bersoff는 소프트웨어를 정보의 부분집합이라 정의했고, IEEE에는 “컴퓨터 프로그램, 처리절차, 규칙과 컴퓨터시스템의 운영과 연관성이 있는 모든 관련 문서와 데이터”로 규정되어 있다. Mills H.D는 소프트웨어를 <Fig. 1>과 같이 7 가지 형태의 집합이라고 정의했다.



<Fig. 1> Software concept

2. 소프트웨어 품질

소프트웨어 품질(Software Quality)에 대한 정의 또한 다양하게 정의되어 왔으며, 먼저 IEEE에서는 소프트웨어 품질을 “소프트웨어가 요구하는 속성들 (desired combination of attributes)을 갖고 있는 정도”라고 정의 하고 있다. 여기서 요구하는 속성들이란 신뢰성, 유지보수성, 가용성 등을 말한다. 미 국방성 (DoD)은 “소프트웨어의 명세된 최종 아이템 사용을 수행하는데 가능한 소프트웨어의 속성정도”라고 정의하였고, Schulmey은 “전체 소프트웨어 제품의 사용 적합성”이라고 정의했다.

3. 소프트웨어 특수성

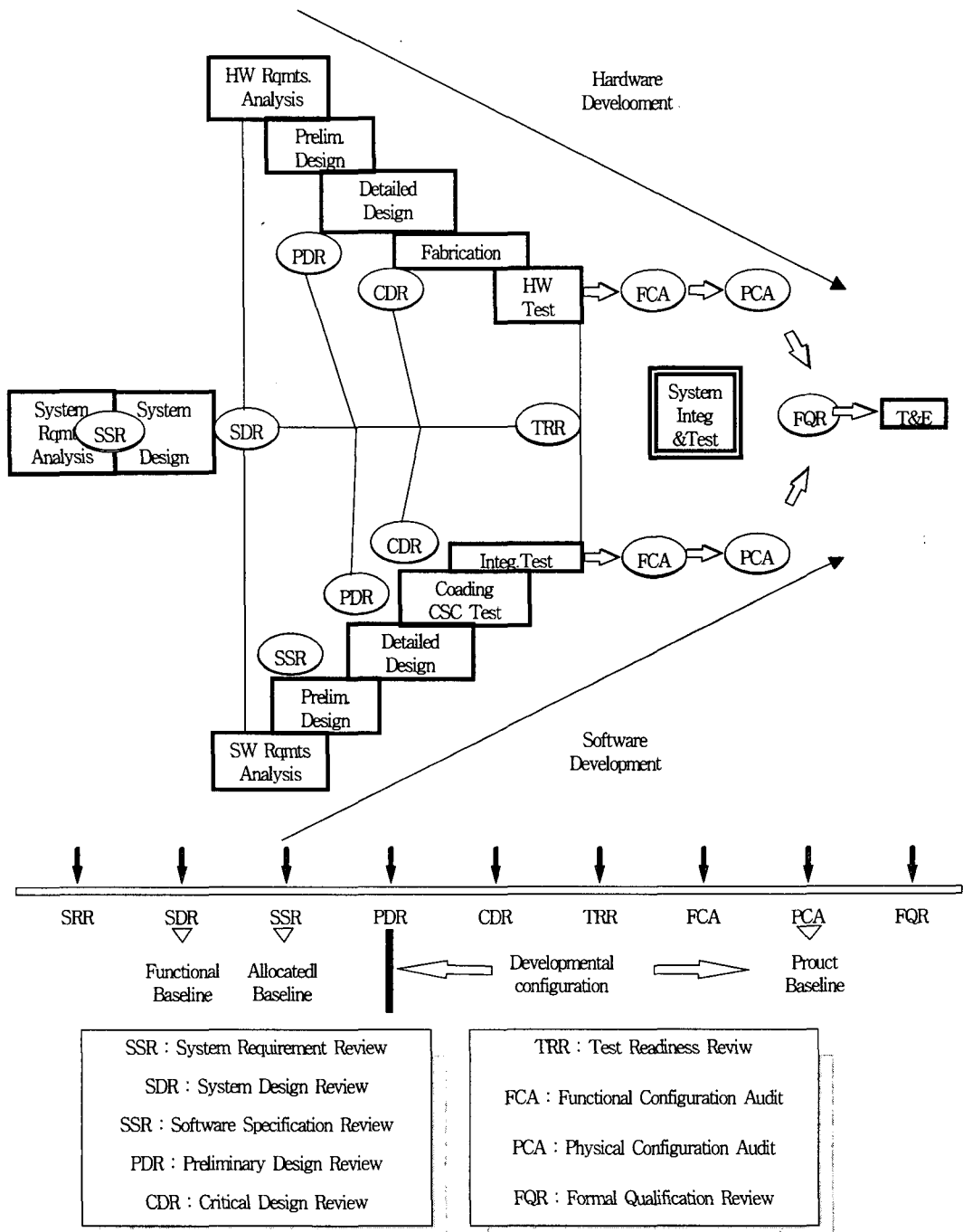
소프트웨어 분야에서는 제품자체가 눈에 보이지 않는다는 점과 개발과정을 기계화하기 어렵다는 점 등으로 인해 품질관리를 체계적으로 확립하는데 어려움이 따른다 하드웨어 품질관리와 비교해서 소프트웨어 품질관리는 다음과 같은 특수성을 지니고 있다.

- 소프트웨어의 경우, 인간적인 요소(human factor)가 차지하는 비중이 대단히 크다.
- 소프트웨어는 품질을 공정 가운데서 확보 또는 만들어 넣는다고 하는 사고방식이 하드웨어에 비하여 약한 경향이 있다.
- 품질을 생각하는 경우에 이용자 입장에서의 접근이 미흡한 경우가 종종 발생한다.
- 소프트웨어의 경우 논리적으로 확고하게 짜여져 있다면 완성된 것에 대하여 저해를 받는 일이 거의 없다는 것 등이다.

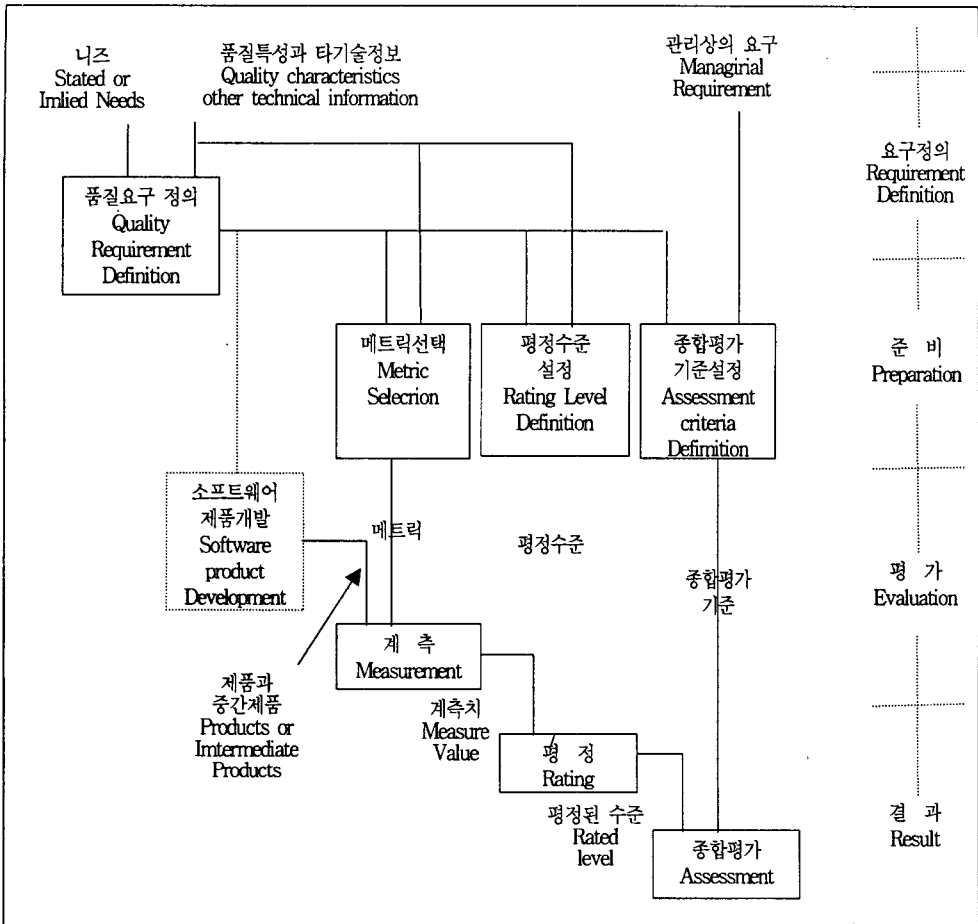
4. 선진 외국의 소프트웨어 품질관리 실태

미국에서는 소프트웨어 개발주기 전 단계에 걸쳐서 품질을 확보할 수 있는 체계를 갖추어 실시하고 있다. 특히 국방성인 경우에는 첨단 군사 장비까지 고 품질 확보를 위해서 계획부터 형상관리(Configuration Management)에 이르기까지 모든 활동을 표준화하여 실시하고 있다. <Fig. 2>는 현재 국방성에서 개발되는 하드웨어와 소프트웨어 개발공정을 나타내고 있다.

일본은 최초로 소프트웨어 공장으로 불리게 된 히다찌(日立)에서는 소프트웨어를 하드웨어 제품과 동일하게 취급하여 품질관리 활동을 하고 있다는 점이다. 소프트웨어의 특수성은 인정하되, 대책을 취하면서 독립된 검사부문을 설치하여 설계에서 유지보수까지 일관된 품질관리를 도모하고 있다. 히다찌의 소프트웨어 품질관리의 가장 특징적인 것을 소개하면 첫째 소프트웨어 검사 부서를 독립적으로 설치하여 검사부분이 품질보증의 총수로 위치를 확고히 하고 있는 것, 둘째 총합 적인 소프트웨어 생산관리시스템(CAPS : Computer Aided Production Control System for Software)이 구축되어 소프트웨어의 신뢰성, 생산성 향상을 도모하는 일, 셋째 분임조 활동(QCC : Quality Control Circle)을 적극 활용하고 있다는 점이다 다시 말하면 품질관리 사고방식은 같고 단지 취급하는 대상이 다르므로 수행 방법을 달리하면 된다는 논리다. 이렇게 하여 소프트웨어 제품의 프로그램 계층구조에서 공정관리, 품질관리 정보를 총합 데이터베이스화하여, 이 Data를 기초로 볼 수 없는 Software 생산공정을 가시화(可視化)함으로써 소프트웨어 품질관리 정도를 향상시키려고 노력하고 있다.



<Fig. 2> Development process of hardware



<Fig. 3> Evaluation process model

III. 소프트웨어 품질관리방법 및 기법

소프트웨어 품질을 관리하기 위해서는 먼저 개발제품에 요구되는 품질특성과 평가기준에 대한 명확한 정의가 필요하다. 소프트웨어 품질은 하드웨어 품질과는 달리 보는 사람마다 다르게 해석하는 경우가 있기 때문이다. 즉 사용자는 사용하기 쉽고 수행 성능이 좋으면 품질이 좋은 것으로 생각하는 반면에 유지 보수자는 오류(error)를 발견하고 디버깅(debugging)이 쉬우며 새로운 기능을 추가하기 용이한 면을 중요시한다. 또한 프로젝트 규모나 품질요구수준, 기술수준 및 시급도 등의 환경 여건을 고려하여 알맞은 개발과정 평가기준을 설정함은 물론 개발과정평가에 있어서는 정성적인 내용을 많이 취급하게 되므로 주관성

의 개입을 완전히 배제하기는 어렵지만 가능한 범위 안에서 평가 상대를 개개의 단위로 세분하고 기준을 명확히 정의함으로써 좀더 정확한 평가가 이루어지도록 해야 한다.

본 논문에서는 소프트웨어 품질관리 방법 및 기법을 국제 표준인 ISO/IEC 9126에서 말하는 평가 절차 모델인 <Fig. 3>에 따라 도출·구현하고자 한다.

1. 품질 요구사항 정의

소프트웨어 개발주기의 첫 단계로 사용자의 요구를 명확히 정의하고, 요구되는 조건을 분석하는 과정이다. 소프트웨어 품질은 ISO/IEC 9126에 의하면 Table 1에 제시된 것과 같이 기능성(functionality), 신뢰성(reliability), 사용성(usability), 이식성(portability)에 의해서 평가된다. 또한 6가지 주요 특성에 종속되는 품질 부특성 21가지를 사용자 요구사항 시 반영될 수 있도록 노력해야 한다. 사용자 만족을 위해서는 이 단계의 중요성을 아무리 강조해도 지나치지 않다.

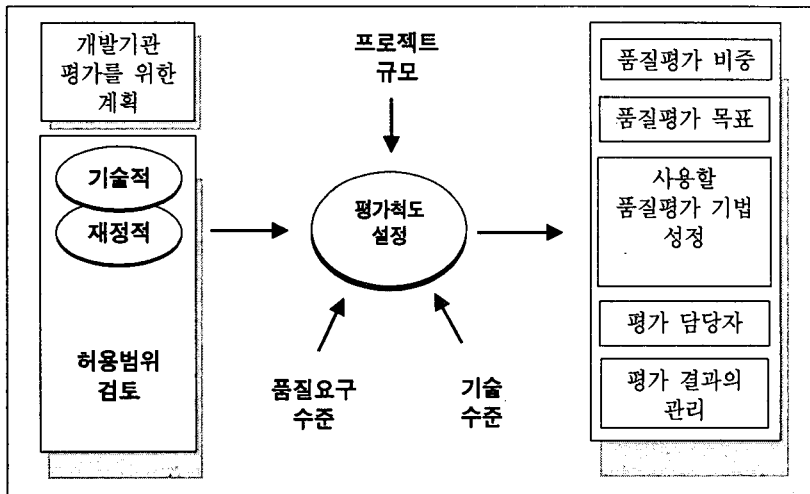
<Table 1> ISO/IEC 9,126 Sorware quality characteristics

품질특성	개 념	부특성
기능성	일련의 기능 존재와 규정된 기능 특성과 관련된 속성의 집합	적합성, 정확성, 상호운용성, 유연성, 보안성
신뢰성	명시된 기간동안 명시된 조건에서 그 성능 수준을 유지하는 소프트웨어 능력과 관련된 속성의 집합	성숙성, 오류허용성, 회복성
사용성	사용을 위한 노력과 그러한 사용에 대한 개개인의 심사와 관련된 속성의 집합	이해성, 습득성, 운용성
효율성	규정된 조건에서 소프트웨어의 성능 수준과 사용된 자원의 양 사이에 관련된 속성의 집합	실행효율성, 자원효율성
보수성	규정된 수정을 수행하기 위하여 필요한 노력과 관련된속성의 집합	해석성, 변경성, 안전성, 시험성
이식성	다른 환경으로 이전되는 소프트웨어 능력과 관련된 속성의 집합	환경적응성, 이식작업성, 일치성, 치환성

2. 품질 평가 계획

이 단계에서는 시스템의 규모, 기술수준 및 특성에 알맞은 계획수립이 이루어

져야한다. 개발제품에 대해서도 품질 목표설정 과정을 통하여 생성된 품질평가 점검 항목을 근거로 정성적 또는 정량적으로 <Fig. 4>와 같이 평가가 이루어지도록 하여야 한다.



<Fig. 4> Planning phase activities of quality evaluation

3. 품질목표 설정

품질목표란 개발하고자 하는 소프트웨어가 목표로 하는 품질 특성을 말하며, 개발 제품 품질평가 척도의 최상위 레벨인 품질 인자 중 적용 가능한 것의 집합이다. 품질 목표는 시스템의 전 단계에 걸쳐 개발 및 평가의 지표가 되므로 프로젝트의 특수성 등 여러 요인을 고려하여 실현 가능하도록 설정되어야 한다. 소프트웨어가 목표로 하는 품질은 시스템의 특성에 따라 달라지며, 시스템 특성은 사용자 요구명세서와 프로젝트 계획서를 참고하여 파악 할 수 있다.

일반적인 시스템 특성들에 대해 강조되는 품질 인자들을 보면 다음과 같다.

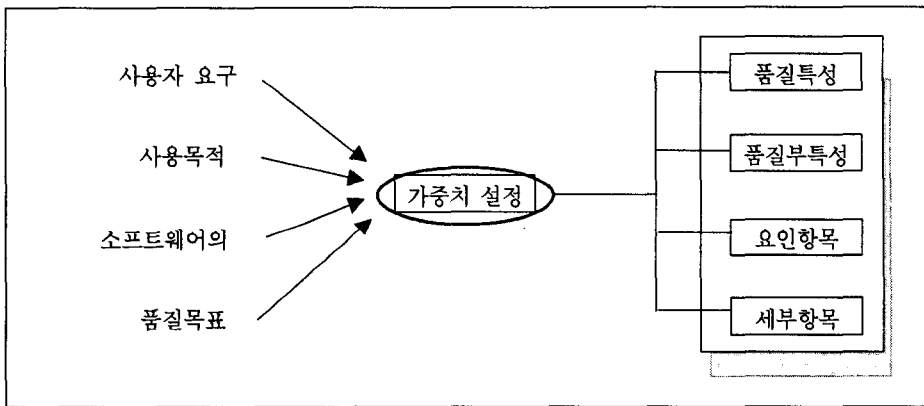
- 시스템 생명주기가 길면 유지보수성이 강조되어야 한다.
- 시스템 하드웨어의 확장, 변경 가능성이 높으면 이식성이 중요하다.
- 사용자의 요구가 확고히 정의되어 있지 않고 업무의 변화 또는 확장가능성이 있으면 유연성을 가지도록 개발하여야한다.
- 네트워크 구성계획이 있으면 상호 운용성 개념이 중요하다.
- 시스템의 변경 가능성이 높으면서 주요기능

이 장기간 필요한 경우에는 주요 기능을 수행하는 모듈들에 대한 재사용성이

요구된다.

4. 가중치 설정

가중치 설정은 사용자의 요구를 정확히 파악하여 계측 대상의 소프트웨어 품질특성의 중요도를 표시하는 단계이다. 대상 소프트웨어는 사용 목적에 따라 품질특성의 중요도가 다르기 때문에 그 목적에 적합하도록 가중치를 적절하게 <Fig. 5>와 같이 부여해야 한다.



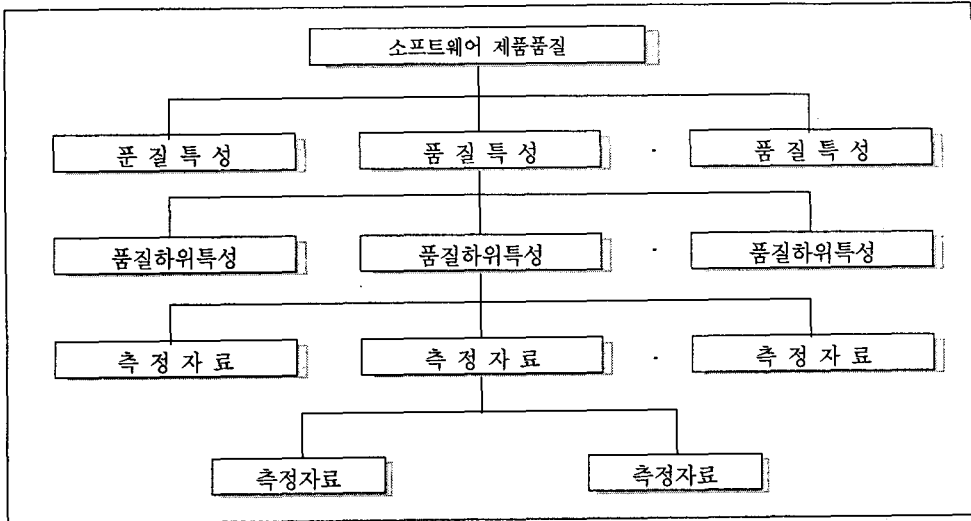
<Fig. 5> Weight decision

5. 매트릭스 선정(metrics selection)

소프트웨어 제품의 특성과 상호 관련되는 매트릭스를 개발할 필요가 있다.

개발주기상의 단계에 따라 혹은 개발환경에 따라 동일한 매트릭스를 사용할 수가 없다. 시스템 분석 단계에서는 정량적으로 측정할 수 있는 매트릭스가 많지 않으므로 Albrecht가 말한 기능점수(Runction Point)를 사용하는 것이 좋으며, 설계단계에서는 조건문, 조건문변수, 기능수, 자료 요소의 비율 등의 매트릭스를, 구현단계에서는 원시 프로그램을 대상으로 한 매트릭스를 사용할 수 있다. 선정된 매트릭스는 <Fig. 6>과 같이 품질 하위특성에 영향을 미치며 항상 일류적이지 않기 때문에 앞에서 언급한 가중치 설정을 신중하게 해야 된다.

예를 들면 ISO/IEC 9126에서 언급되고 있는 효율성(eficiency) 중의 실행 효율성(time behaviour)에 대한 매트릭스 선정을 위한 기본 개념을 소개하며 <Table 2>와 같이 나타낼 수 있다고 본다.



<Fig 6> Evaluation Metrics of Software Quality

<Table 2> Metrics of Time Behaviour

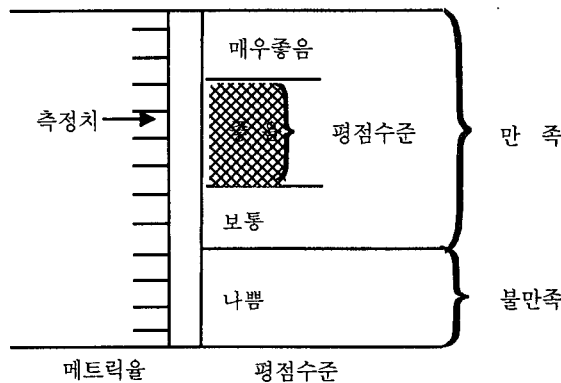
속성집합	정 의	측 정 식
1. 턴어라운드 시간 (Tt)	처리요구가 발생된 시점에서 처리결과가 얻어지기까지의 경과시간	$Tt = Pdt - Pt$ Pdt : 처리결과 산출시간 Pt : 처리요구 발생시간
2. 응답시간 (Rt)	컴퓨터 시스템에 대하여 조회 또는 요구 종료에서 응답초기까지의 경과시간	$Rt = Rst - Ret$ Rst : 컴퓨터 시스템에서 응답 개시시간 Ret : 컴퓨터 시스템으로의 요구 종료시간
3. CPU 경과시간(Ct)	프로그램의 실행개시에서 종료까지 필요한 경과시간	$Ct = Ret - Rst$ Ret : 프로그램 실행종료시간 Rst : 프로그램 실행개시시간
4. CPU 실행시간 (CRt)	프로그램의 실행개시부터 종료까지 요구되는 CPU 시간	$CRt = \sum_{i=1}^n Pi + \sum_{i=1}^n Oi$ Pi : 프로그램 실행에 필요한 CPU시간 Oi : 프로그램 실행에 따르는 OS나 관련 프로그램의 실행에 필요한 CPU시간
5. 입출력 처리시간 (IOpt)	외부기억과의 입출력에 요구되는 시간	$IOpt = \sum_{i=1}^n (IO1i + IO2i)$ IO1i : OS에 의해서 입출력 처리명령이 발행된 시간 IO2i : OS가 입출력 처리의 완료를 받는 시간

6. 대기시간(Wt)	프로그램의 실행이 OS나 자원에 의한 인터셉트에 의해서 대기한 시간	· $Wt =$ 프로그램실행의 총 대기 시간
7. 네트워크 처리시간(NPt)	컴퓨터 센터와 단말간에서 전송의 개시 처리에서 종료 처리까지 걸린 시간	· $NPt = \sum_{i=1}^n (CRi + CSi)$ CRi : 센터 또는 단말에서 송신처리가 수신된 시간 CSi : 센터에서 단말까지 또는 단말에서 센터까지 송신처리를 개시한 시간
8. 단말 처리시간(TPt)	단말에서 처리의 개시에서 종료까지 요구된 시간	· $TRt = \sum_{i=1}^n Pi$ Pi : 단말처리시간
9. 처리량(Pa)	단위시간내에 컴퓨터 시스템에 의해 수행된 업무량	· $Pa =$ 컴퓨터 시스템에서 처리된 업무량 / 소요시간
10. 트랜잭션 처리건수(TPn)	단위 시간당 처리한 트랜잭션의 건수	· $TPn =$ 트랜잭션 처리건수 / 소요시간

6. 평점수준 설정(rating level definition)

메트릭스를 사용하여 계량적으로 측정된 값은 시스템의 특성에 따라 만족할 수도 있고, 불만족일 수도 있다. 이는 시스템의 특성과 사용자의 기대요구가 다르기 때문에 미흡, 보통, 만족 등과 같이 몇 개의 범주로 분류되어야 하는데 이를 위해 평점수준을 설정하게 된다. 이 값은 메트릭(metric scale)으로 사상된다.

<Fig. 7>은 측정치와 평점수준을 나타낸 것으로 평점수준의 범위가 클수록 정확하게 평가할 수 있으나 상대적으로 노력이 많이 요구된다.



<Fig. 7> Measured value and rated

7. 품질 측정(Quality Measurement)

소프트웨어 제품 생산물 또는 개발 과정상 얻어지는 명세서 등과 같은 기초 데이터를 가지고 실제로 측정하는 단계이다. <Table 3>은 소프트웨어 개발단계별 산출물 및 주요 품질관리 평가항목을 나타낸 것이다.

가령 시스템 설계단계에서 모듈화 및 복잡도에 관한 품질특성을 측정한다고 가정하면 다음과 같이 정량화 할 수 있다. 먼저 모듈화에 대한 측정기법으로는 McCabe[15]의 사이클로매틱 수(Cyclomatic number)나 Halstead[16]의 이론 등이 있는데 본 논문에서는 Halstead가 제안한 모듈 복잡도를 측정하기로 한다.

- n_1 : 프로그램에서의 구분되는 연산자(operator)의 수
- n_2 : 프로그램에서의 구분되는 피연산자(operand)의 수
- N_1 : 프로그램에서의 연산자들의 총 출현회수
- N_2 : 프로그램에서의 피연산자들의 총 출현회수를 나타내는 변수가 있다.

위의 변수들로부터 얻어지는 측정치는 다음과 같다.

프로그램 단어(vocabulary) : $n = n_1 + n_2$

프로그램 길이(length) : $N = N_1 + N_2$

프로그램 부피(volume) : $V = N \log_2 n$

개발노력 (development effort) :

$$E = V \cdot D$$

$$V = (N_1 + N_2) \log (n_1 + n_2)$$

$$D = (n_1 / 2) \cdot (N_2 / n_2)$$

V는 모듈의 크기(Volume)이며, D는 모듈의 개발 나이도(difficult)이다. 개발 노력 E의 값이 1,000정도이면 모듈의 구성이 복잡한 경우로 모듈을 분할하거나 모듈의 구현에 사용된 연산자의 수인 n을 줄여야 한다. E의 값이

$E \leq 1,000$ 이면 우수,

$1,000 < E \leq 10,000$ 이면 보통,

$E > 10,000$ 되면 불량이다.

<Table 3> Evaluation viewpoints of software products

개발단계	개발제품	주요 평가관점
요구 분석	<ul style="list-style-type: none"> · 요구분석 명세서 · 시스템시험계획서(STP) 	<ul style="list-style-type: none"> · 요구정의 일관성/완전성 · 요구사항의 정확도 · 기능/성능 요구의 타당성 · 시스템 요구의 추적성 · 시험평가 계획의 충분성
시스템 설계	<ul style="list-style-type: none"> · 시스템설계서(SDS) · 통합시험계획서(ITP) 	<ul style="list-style-type: none"> · 설계의 일관도 및 완전도 · 모듈화, 단순화, 독립화 · 소프트웨어 구조의 표현, 이해도 및 일치성 · 코드부여방법의 적절성 · 운용 용이성의 고려여부 · 데이터 표현의 공통도 · 시험평가 계획의 충분성 · 오류허용 정도 · 처리절차의 정확도 · 액세스 관리도 · 요구사항에 대한 추적성
프로그래밍	<ul style="list-style-type: none"> · 프로그램 	<ul style="list-style-type: none"> · 시스템 설계서에 대한 추적성 · 코딩표준의 준수여부 · 상세설계에 대한 일관도 및 완전도
통합 시험	<ul style="list-style-type: none"> · 통합시험 명세서(ITS) · 통합시험 보고서(ITR) 	<ul style="list-style-type: none"> · ITP 및 SDS에 대한 추적성 · 시험사례의 충분성 · 인터페이스의 정확도 · 처리기능의 정확도 · 시험결과의 신빙성 및 ITS에의 추적성 · ITS와 ITR의 일관도 및 완전도
시스템 시험	<ul style="list-style-type: none"> · 시스템시험 명세서(STS) · 시스템시험 보고서(STR) 	<ul style="list-style-type: none"> · RAS 및 STP에 대한 추적성 · 시험 사례의 충분성 · 처리기능의 완전도 · 액세스의 통제 및 기록 · 운용 용이도 · 오류에 대처하는 능력 · 처리효율 · 시험결과의 신빙성 및 STS에의 추적성 · 데이터베이스의 일치성, 무결성, 보안성 · STS와 STR의 일관도 및 완전도

<Fig. 8>은 Ada 프로그램으로 소수(素數)를 구하는 프로그램 예이다.

```

begin
for N in 2..X loop
  for I in 2..N loop
    if (N mod I) = 0
      then exit ;
    end if ;
  end loop ;
  if I = N
    then Put (N);
  end loop ;
end ;

```

<Fig. 8>

위 프로그램에서 $n_1 = 17$, $n_2 = 5$,

$N_1 = 34$, $N_2 = 12$ 가 된다.

프로그램 단어 : $n = 17 + 5 = 22$

프로그램 길이 : $N = 34 + 12 = 46$

프로그램 부피 : $V = 46 \log_2 22 = 205.2$

개발노력 : $E = V \cdot D$

$V = 205.2$

$D = 17 \times 5 / 2 \times 12 \approx 3.54$

$E = 205.2 \times 3.54 = 726.41$

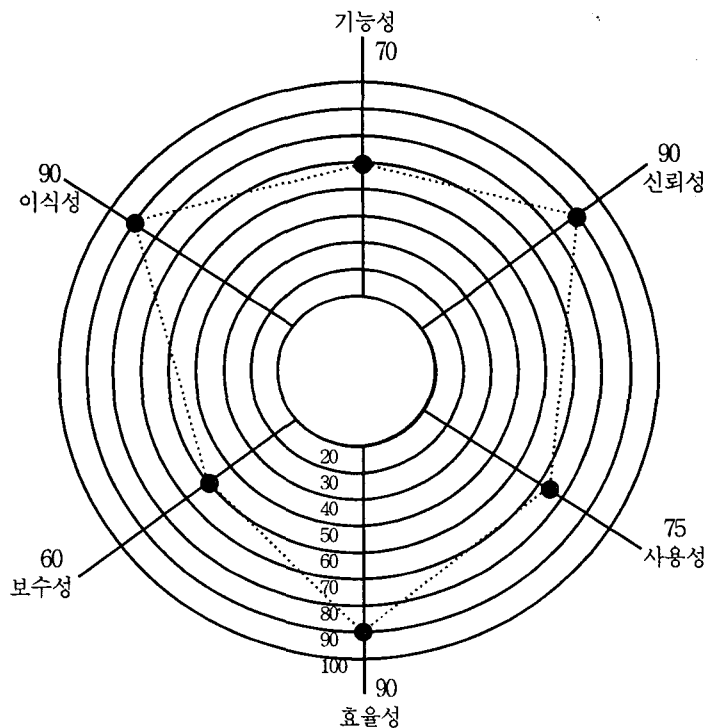
이 되므로 양호한 프로그램이라고 판정을 내릴 수 있다.

8. 평점(rating)

측정값을 평점수준과 비교하여 소프트웨어의 품질특성이 어느 정도 위치에 있는가를 정하는 단계이다.

9. 종합평가(assessment)

세부적으로 측정 대상의 품질특성마다 평점 된 결과를 평가기준으로 결과값을 산출하고, 이와 같이 얻어진 종합치를 가지고 소프트웨어 품질의 전체적인 평가를 행하는 것이다. ISO/IEC 9126에서 언급되는 6가지 품질특성 즉, 기능성(functionality), 신뢰성(reliability), 사용성(usability), 효율성(efficiency), 보수성(maintainability), 이식성(portability)을 눈으로 보이는 관리를 위해 <Fig. 9>와 같이 레이더 도표(radar chart) 예로 나타낸 것이다.



<Fig. 9> Radar chart of software quality characteristics

10. 분석 및 시정조치

평가과정에서 나타난 문제점을 신속히 피드백(feedback)시켜 시정조치를 한다. 이때 정해진 양식에 따라 기록·유지하여 자료의 역사성을 가지게 하고, 이렇게 함으로써 소프트웨어의 형상관리도 함께 도모할 수 있다. 시정조치를 요구할 때에는 일방적인 지시나 무조건적인 개선을 요구하기보다는 가능한 한 해결 방안이나 기법 등을 제안함으로써 소프트웨어의 품질 향상을 기하는 것이 바람직하다.

IV. 결 론

소프트웨어의 개발 및 유지보수에 관련된 비용이 급증하고 대규모 소프트웨어 개발에서의 품질저하에 대한 심각성이 높아짐에 따라 품질관리 체계확립이 요구된다. 이러한 체계확립을 위한 접근방법이 여러 가지 있으나 본 논문에서는 국제표준 규격인 ISO/IEC 912에서 제안하고 있는 평가 절차 모델에 따라서 품질관리 방법 및 기법을 개발해 보았다.

주요 내용을 요약해 보면 첫째, 사용자 만족, 더 나아가 사용자 감동을 가져올 수 있는 소프트웨어를 개발하기 위해서는 다 만들어 놓은 제품에 만족, 불만족 판정을 내리는 행위가 아니라 소프트웨어 개발주기 상에서 품질을 확보해야 한다는 것이다. 둘째, 개발과정에서 오류(error)를 조기에 발견하여 feedback 시켜주어야만 유지보수 비용이 적게된다. 셋째, 설계단계에서의 모듈화 및 복잡도 측정에서는 합리적인 프로그램 길이(length)와 크기(volume)가 되도록 해야 한다. 넷째 소프트웨어의 특성상 'soft' 한 보이지 않는 부분을 가시화(可視化)하여 눈으로 보이는 관리를 할 수 있도록 구현해 보았다.

향후 과제로는 소프트웨어 개발주기 단계별 즉, 시스템 분석, 시스템 설계, 프로그래밍, 테스트 및 유지보수 단계에서의 품질 측정 및 평가를 위한 매트릭스(metrics) 개발과 ISO/IEC 9126에서 언급되고 있는 품질특성 및 부특성에 대한 측정항목 도출과 가중치 부여 방법이 연구되어야 한다.

참 고 문 헌

1. 정보통신부(1998), “정보화 촉진에 관한 동향과 시책에 관한 내용”.
<http://www.nca.or.kr/NCA98/news/98ap-0.htm>
2. 한국전산원, “국가 정보화 백서”, 1998
3. 장영숙(1994), “미 국방성(DoD) 품질보증 활동소개, 국방품질관리연구소”.
4. ISO/IEC 9126(1991), “Information technology- Software Product Evaluation- Quality Characteristics and guide-lines for their use.”
5. Bersoff E. H. et al.(1980). Software Configuration Management : An Investment in Product Integrity, Prentice Hall, Englewood Cliff.
6. IEEE-STD-729(1983), “Software Engineering Technical Committee of the IEEE Computer society - IEEE Standard Glossary of Software Engineering Terminology,” New York.
7. Mills H.D.(1980), “Principles of Software Engineering,” *IBM System Journal*, Vol.,19, No.4.
8. U.S. Depart. of Defense, DOD-STD-2167(1985), Defense system Software Development, Washington, D.C.
9. U.S. Depart. of Defense, DOD-STD-2168(1985), Defense system Software Development, Washington, DC.
10. Schulmeyer G.G. and Mcmanus J.I.(1987), “Handbook of Software Quality Assurance, Van Nostrand Reinhold Company Inc., N.Y.
11. 森口 一(1990), ソフトウェア, 品質管理, トフイツ, 日本規格協會.
12. 시스템공학연구소(1997), 소프트웨어 품질평가기술개발(I), 정보통신부.
13. Albrecht A.J.(1979), “Measuring Application Development Productivity,” Proc. Guide/Share Application Symp.
14. 양해술(1997), “소프트웨어 품질측정 기록 및 지원 툴킷의 개발,” 시스템공학연구소 위탁 연구과제.
15. McCabe T.J., “A complexity Measure,” *IEEE Trans. on Software Eng.*, Vol. SE-2, No.4, 1976.
16. Halstead M.H(1997)., *Elements of Software Science*, New York : Elsevier North-Holland.

Abstract

A Study on Method for software Quality control

Im, Dae-heug · Jang, Young-suk

As the competition between countries has become higher and the notion of software quality has been widely spread, it has been necessary to develop technologies that can ensure and produce high quality software. With the advent of information-oriented society, quality control has to transfer to the quality control activities focused on software system instead of those activities focused on hardware system. If so, how do we get to handle the quality control on the basis of the new approach.

Also, as software applications have grown, so too has the importance of software quality. In order to manage software quality, the technology to specify and evaluate both the software product quality and development process quality objectively quantitatively is most important.

To produce products of good quality, we need a more progressive quality control system according to the need of software development life cycle. In other words, we do software right the first time or build quality in the process. On the basis intermediate and final time or build quality in the process. On the basis of data achieved, we can evaluate the products according to the consequences of the data,

What are the problems to contrive the software quality control system?, we can promote the quality of products. To achieved that goal, we can provide a suitable the technique and method of software quality control.