「經營情報硏究」 第18號
大韓經營情報學會
2006年 6月

Software 품질의 정량적 측정과 평가

임 대 혁*

〈목 차〉

Ⅰ. 서 론

세계 각국은 인터넷을 통하여 하나로 통합되어가고 있으며 글로벌 경영으로 인한 세계 일류 기업들과의 직접경쟁은 가속 되어가고 있다. 이러한 글로벌 경영으로 인하여 지식정보의 중요성이 다시 한 번 대두되어 경제 패권을 잡기 위한 핵심요소가 되어 정보화가 선택사항이 아닌 필요불가결한 요소가 되었다.

농업혁명과 산업혁명 이후 제3의 혁명으로 간주되고 있는 정보화를 통한 '디지털 혁명'은 이제 거부할 수 없는 커다란 흐름이 되었으며, 이미 세계경제 질서는 정보화를 통해 재편되어 오고 있는 실정이다. 또한 한국전산원에 의하면 우리나라 정보화 수준은 매우 놀라운 속도로 발전하여 초고속망과 인터넷 사용자의 수준은 최고임을 보여주고 있다. 그러나 서버의 사용자와 보안 소프트웨어의 개발 면에서는 아직은 뒤떨어진 것으로 나타나고 있다. 이 분야에서도 더욱 노력한다면 다른 나라와 동등하거나 앞서는 분야도 많을 것으로 생각된다.

이에 편승하여 모든 부분이 자동화되어 가는데, 특히 가정 자동화(HA : Home Automation), 사무자동화(OA : Office Automation), 공장 자동화(FA : Factory

* 혜천대학 디지털마케팅과 교수

Automation)의 진행이 가속화되어감에 따라 소프트웨어가 차지하는 비중이 매우 급증하고 있다. 그러나 이러한 소프트웨어의 개발은 어느 정도 많은 진전이 있었으나 소프트웨어의 품질관리체계에 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 "제작 공정에 품질을 주입하라(build quality into process)"라는 개념으로 소프트웨어 개발주기(Development life cycle) 전 단계에 걸쳐서 품질관리 활동을 하지 않으면 안 된다.

미국의 경우 모든 분야에서 그렇지만 특히 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 측정방법 및 기법을 제시하였다. 4장에서는 연구의 결과와 향후 연구과제에 대해서 언급했다.

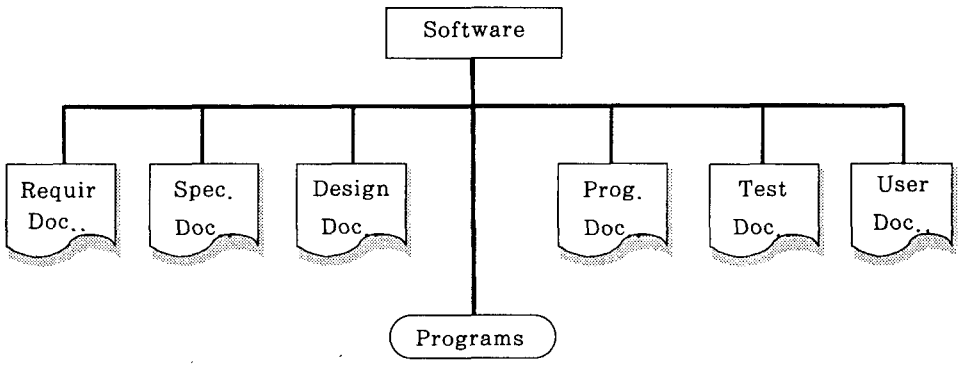
Ⅱ. 소프트웨어 품질관리

1. 소프트웨어(S/W : Software)

소프트웨어라 함은 하드웨어라고 불리는 컴퓨터 기계장치부에 대응하는 부분으로 시스템과 문서를 사용하고 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 정의하고 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 〈그림 1〉과 같이 7가지 형태의 집합이라고 정의했다.



〈그림 1〉 Software concept

2. 소프트웨어 품질

소프트웨어 품질(Software Quality)에 대한 정의 또한 다양하게 정의되어 왔으며, 최근에 품질에 대한 관심이 매우 높아져서 품질이 매우 좋은 소프트웨어를 어떻게 개발하느냐가 주요 관심사항이 되어왔다. 먼저 IEEE에서는 소프트웨어 품질을 "소프트웨어가 요구하는 속성들(desired combination of attributes)을 갖고 있는 정도"라고 정의하고 있다. 여기서 요구하는 속성들이란 Robert L. Glass에 의하면 이식성(potability), 신뢰성(reliability), 효율성(efficiancy), 사용성(usability), 테스트용이성(testability), 이해용이성(understandability), 수정용이성(modifiability) 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 정의하기가 어려운 부분이 있다. 소프트웨어 개발은 생산공정인가 창조인가 하는 부분이다. 소프트웨어의 개발을 생산공정으로 다루려고 하는 접근방법은 대개의 경우 그 가치가 과대평가되어 있다고 볼 수 있지만 그러한 접근이 매력적이기 때문에 때때로 많이 선택되어 지고 있다. "포드시스템과 같은 체

계를 구축하면, 마치 소프트웨어개발을 생산공정과 같이 통제할수 있어요." 라고 말하는 사람도 있다. 또한 관리자의 입장에서도 매력적이어서 개발자를 부품 취급함으로써 개발이 잘 안될 때 쉽게 바꾸는 경우도 있다. 이제 창조의 관점에서 소프트웨어의 개발을 생각해 보면 그것들은 생산공정의 통제기법을 통해 언제든지 만들어 질수 있는 것인가? 그렇지 않다. 그러한 유형의 소프트웨어를 개발하는데 있어 가장 중요한 요소는 개발자의 영감이다. 그리고 그것을 기반으로 한 동기 부여와 창조적 실행력의 극대화이다. 이러한 점에서 소프트웨어의 품질관리를 체계적으로 확립하는데 어려움이 따른다. 하드웨어 품질관리와 비교해서 소프트웨어 품질관리는 다음과 같은 특수성을 지니고 있다.

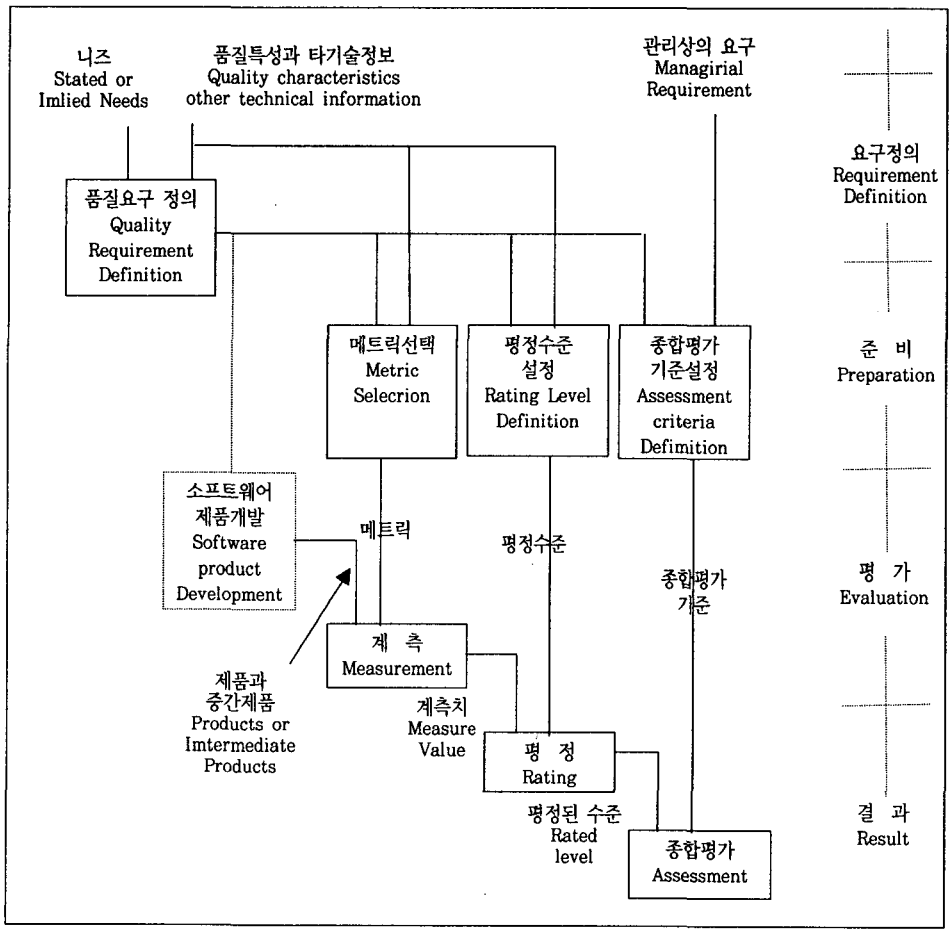
- 소프트웨어의 경우, 인간적인 요소(human factor)가 차지하는 비중이 대단히 크다.
- 소프트웨어는 품질을 공정 가운데서 확보 또는 만들어 넣는다고 하는 사고방식이 하드웨어에 비하여 약한 경향이 있다.
- 품질을 생각하는 경우에 이용자 입장에서의 접근이 미흡한 경우가 종종 발생한다.
- 소프트웨어의 경우 논리적으로 확고하게 짜여져 있다면 완성된 것에 대하여 저해를 받는 일이 거의 없다는 것 등이다.

4. 소프트웨어 품질관리 실태

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

일본에서 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 총합적인 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 수행방법을 달리 하면 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.



〈그림 2〉 Evaluation process

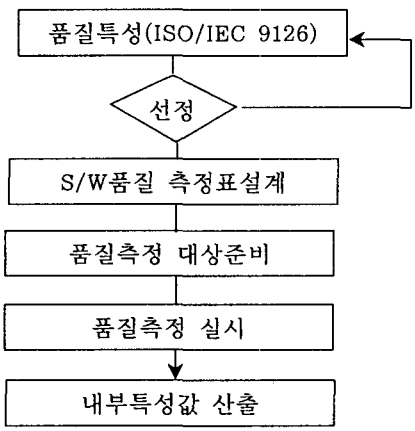
Ⅲ. 소프트웨어 품질측정 및 평가

소프트웨어를 보다 더 잘 이용하고 유용하게 사용하려면 소프트웨어의 품질을 관리해야 하는데 먼저 소프트웨어에 요구되는 품질특성과 평가기준에 대한 명확한 정의가 필요하다. 소프트웨어 품질은 하드웨어 품질과는 달리 사용하는 사람과 유지보수하는 사람이 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 확장하기 쉬운 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 주관성의 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 〈그림 3〉에 따라 도출 · 구현하고자 한다.

1. 품질 측정 개요

정량적인 소프트웨어 품질측정을 위해서 먼저 품질특성을 선정해야 하고 이 특성에 따라 품질측정표를 만들어야 한다. 그 다음 개발단계별 품질측정 대상물을 준비하고 품질측정을 실시하면 된다. 이를 흐름도로 나타내면 다음과 같다.



〈그림 3〉 소프트웨어 품질측정 절차도

2. 품질 특성 선정

소프트웨어 품질은 품질특성으로 구성되는데 ISO/IEC 9126-1에서 제시하고 있는 품질특성을 이용하기로 한다. 왜냐하면 세계 각국의 소프트웨어 전문가들에 의해 심층 연구되고 또한 국제표준화에 부응한다는 좋은 이점이 있기 때문이다.

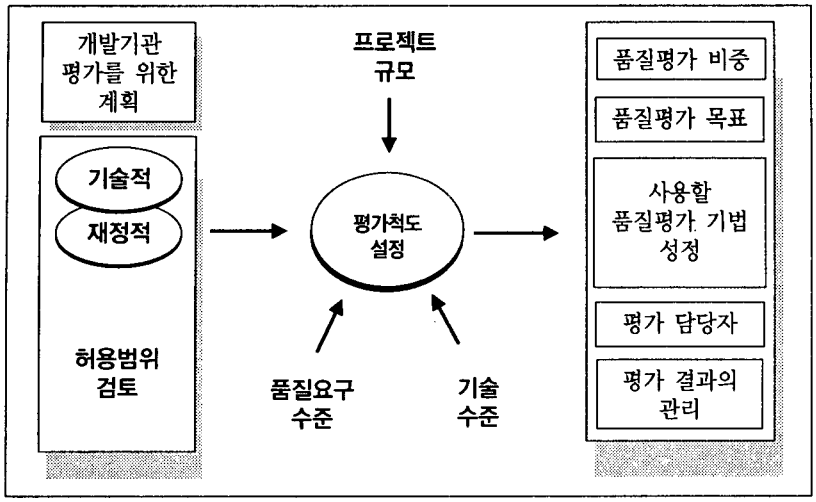
〈표 1〉 주품질특성 코드 및 정의

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 관련된 속성의 집합	해석성, 변경성, 안전성, 시험성
이식성	다른 환경으로 이전되는 소프트웨어 능력과 관련된 속성의 집합	환경적응성, 이식작업성, 일치성, 치환성

ISO/IEC 9126에서 제시하고 있는 품질특성으로는 〈표 1〉에서와 같이 주 품질특성으로써 기능성(functionality), 신뢰성(reliability), 사용성(usability), 이식성(portability)6개 항목이 있다. 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.

3. 품질 평가 계획

이 단계에서는 시스템의 규모, 기술수준 및 특성에 알맞은 계획수립이 이루어져야한다. 개발제품에 대해서도 품질 목표 설정 과정을 통하여 생성된 품질평가 점검항목을 근거로 정성적 또는 정량적으로 〈그림 4〉와 같이 평가가 이루어지도록 하여야 한다.



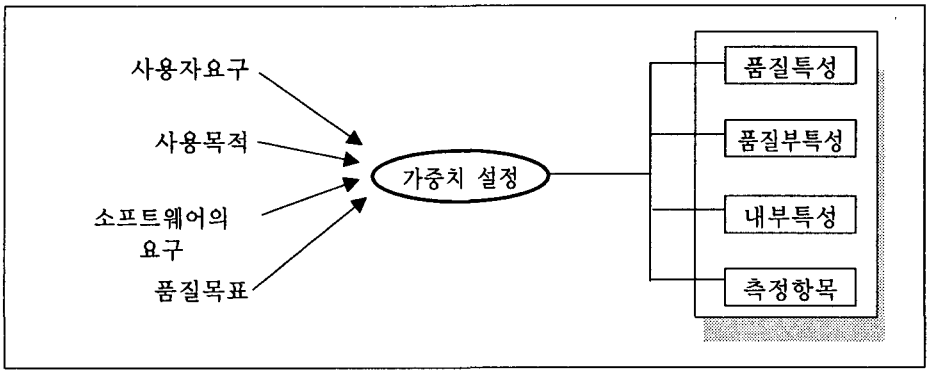
〈그림 4〉 Planning phase activities of quality

4. 품질목표 설정

품질목표란 개발하고자 하는 소프트웨어가 목표로 하는 품질 특성을 말하며, 개발 제품 품질평가 척도의 최상위 레벨인 품질 인자 중 적용가능한 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 파악할 수 있다.

일반적인 시스템 특성들에 대해 강조되는 품질 인자들을 보면 다음과 같다.

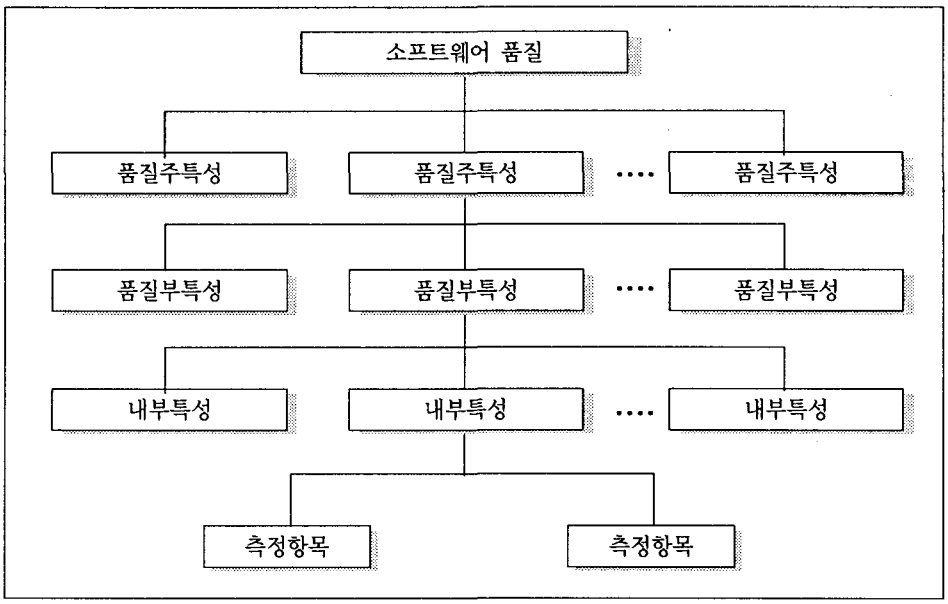
- 시스템 생명주기가 길면 유지보수성이 강조되어야 한다.
- 시스템 하드웨어의 확장, 변경 가능성이 높으면 이식성이 중요하다.
- 사용자의 요구가 확고히 정의되어 있지 않고 업무의 변화 또는 확장가능성이 있으면 유연성을 가지도록 개발하여야한다.
- 네트워크 구성계획이 있으면 상호 운용성 개념이 중요하다.
- 시스템의 변경 가능성이 높으면서 주요 기능이 장기간 필요한 경우에는 주요 기능을 수행하는 모듈들에 대한 재사용성이 요구된다.



〈그림 5〉 Weight decision

5. 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 〈그림 6〉과 같이 부여해야 한다.



〈그림 6〉 Evaluation Metrics of Software Qualiy

6. 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 Rig. 6과 같이 품질 하위특성에 영향을 미치며 항상 일류적이지 않기 때문에 앞에서 언급한 가중치 설정을 신중하게 해야 된다.

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

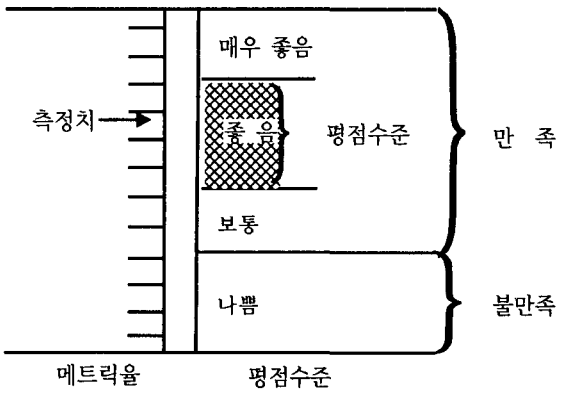
〈표 2〉 Metrics of Time Behaviour

속성집합	정 의	측 정 식
1. 턴어라운드 시간(Tt)	처리요구가 발생된 시점에서 처리결과가 얻어지기까지의 경과시간	· Tt = $P_ot - P_rt$ P_ot : 처리결과 산출시간 P_rt : 처리요구 발생시간
2. 응답시간(Rt)	컴퓨터 시스템에 대하여 조회 또는 요구 종료에서 응답초기까지의 결과시간	· Rt = $R_st - R_et$ R_st : 컴퓨터 시스템에서 응답 개시시간 R_et : 컴퓨터 시스템으로의 요구 종료시간
3. CPU 경과시간 (Ct)	프로그램의 시행개시에서 종료까지 필요한 경과시간	· Ct = $R_et - R_st$ R_et : 프로그램 실행종료시간 R_st : 프로그램 실행개시시간
4. CPU 실행시간 (CRt)	프로금의 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 종료처리까지 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)	단위시간 내에 컴퓨터 시스템에 의해 수행된 업무량	· Pa = 컴퓨터 시스템에서 처리된 업무량 /소요시간
10. 트랜젝션 처리건수(TPn)	단위 시간당 처리한 트랙젝션의 건수	· TPn = 트랜젝션 처리건수/소요시간

7. 평점수준 설정(rating level definition)

매트릭스를 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 매트릭(metric scale)으로 사상된다.



〈그림 7〉 Meaxured value and rated level

〈그림 7〉은 측정치와 평점수준을 나타낸 것으로 평점수준의 범위가 클수록 정확하게 평가할 수 있으나 상대적으로 노력이 많이 요구된다.

8. 품질 측정(Quality Measurement)

소프트웨어 제품 생산물 또는 개발 과정상 얻어지는 명세서 등과 같은 기초 데이터를 가지고 실제로 측정하는 단계이다. 〈표 3〉은 소프트웨어 계발단계별 산출물 및 주요 품질관리 평가항목을 나타낸 것이다.

가령 시스템 설계단계에서 모듈화 및 복잡도에 관한 품질특성을 측정한다고 가정하면 다음과 같이 정량화할 수 있다. 먼저 모듈화에 대한 측정기법으로는 McCabe [15]의 사이클로매틱수(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 출현횟수
- N_2: 프로그램에서의 피연산자들의 총 출현횟수를 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. 개발노력 E의 값이 1,000정도이면 모듈의 구성이 복잡한 경우로 모듈을 분할하거나 모듈의 구현에 사용된 연산자의 수인 n_1을 줄여야 한다. E의 값이

$E \leq 1{,}000$ 이면 우수,

$1{,}000 < E \leq 10{,}000$ 이면 보통,

$E > 10{,}000$ 되면 불량이다.

〈표 3〉 Evaluation viewpoints of software products

개발단계	개발제품	주요 평가관점
요구 분석	· 요구분석 명세서	· 요구정의 일관성/완전성 · 요구사항의 정확도 · 기능/성능 요구의 타당성 · 시스템 요구의 추적성
	· 시스템시험계획서(STP)	· 시험평가 계획의 충분성
시스템 설계	· 시스템설계서(SDS)	· 설계의 일관도 및 완전도 · 모듈화, 단순화, 독립화 · 소프트웨어 구조의 표현, 이해도 및 일치성 · 코드부여방법의 적절성 · 운용 용이성의 고려여부
	· 통합시험계획서(ITP)	· 데이터 표현의 공통도 · 시험평가 계획의 충분성 · 오류허용 정도 · 처리절차의 정확도 · 액세스 관리도 · 요구사항에 대한 추적성
프로그래밍	· 프로그램	· 시스템 설계서에 대한 추적성 · 코딩표준의 준수여부 · 상세설계에 대한 일관도 및 완전도
통합 시험	· 통합시험 명세서(ITS)	· ITP 및 SDS에 대한 추적성 · 시험사례의 충분성 · 인터페이스의 정확도 · 처리기능의 정확도
	· 통합시험 보고서(ITR)	· 시험결과의 신빙성 및 ITS에의 추적성 · ITS와 ITR의 일관도 및 완전도
시스템 시험	· 시스템시험 명세서(STS)	· RAS 및 STP에 대한 추적성 · 시험 사례의 충분성 · 처리기능의 완전도 · 액세스의 통제 및 기록 · 운용 용이도
	· 시스템시험 보고서(STR)	· 오류에 대처하는 능력 · 처리효율 · 시험결과의 신빙성 및 STS에의 추적성 · 데이터베이스의 일치성, 무결성, 보안성 · STS와 STR의 일관도 및 완전도

〈그림 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 ;
```

위 프로그램에서 n_1= 17,, n_2= 5,

N_1= 34, N_2= 12가 된다.

프로그램 단어 : n = 17 + 5 = 22

프로그램 길이 : N = 34 + 12 = 46

프로그램 부피 : V = 46 log2 22 = 205.2

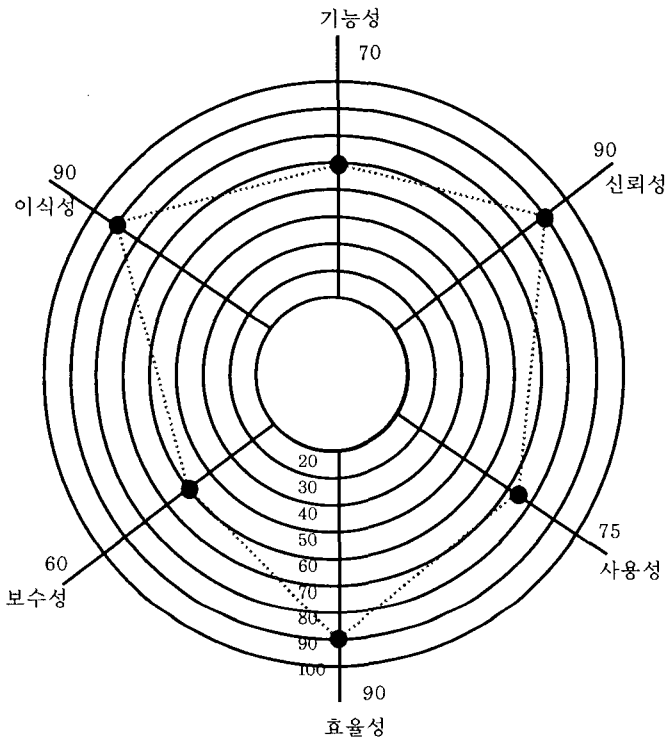
개발노력 : E = V · D

V = 205.2

D = 17×5 / 2×12 ≒ 3.54

E + 205.2×3.54 = 726.41

이 되므로 양호한 프로그램이라고 판정을 내릴 수 있다.

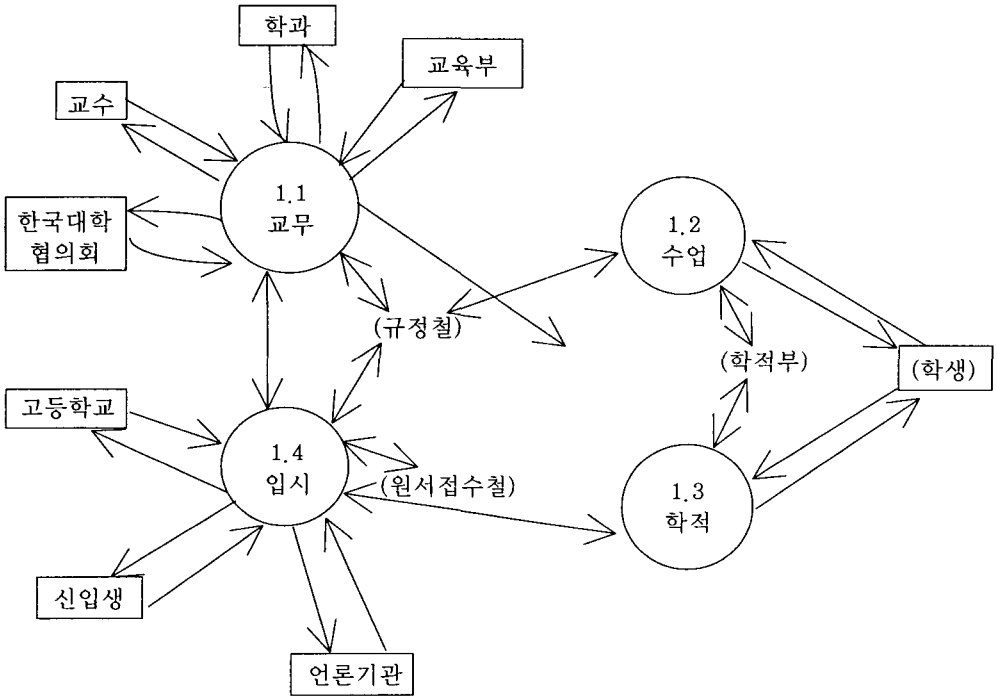


〈그림 8〉 Radarchart of software quality characteristics

Ⅳ. 품질측정과 평가 적용 사례

1. 품질측정

품질측정 대상 시스템은 대학의 주요 학사업무 중 교무, 수업, 학적 및 입시업무 4종으로 하였으며 이는 2,3년제 대학 대부분이 학칙과 학사업무가 동일하기 때문이다. 이러한 학사업무를 자료흐름도로 나타내면 아래와 같다. 각 업무의 주요 내용을 보면 교무업무는 교원임용, 승진, 연구실적 및 출장등주로 인사관리 사항이며, 수업업무는 시간표, 교과과정, 출석부관리 등이다. 학적업무는 재적, 재학, 휴학, 졸업 등 주로 제증명 발급업무이며, 입시관리업무는 입시지원자, 전형방법, 합격자 발표 등 주로 입시전형과 관련된 업무이다.



〈그림 9〉 학사업무 자료흐름도

개발된 〈표 3〉 품질측정표 양식에 따라 개발 단계별 산출물을 대상으로 품질측정을 실시한다. 품질측정자는 실무경험과 전산업무에 경험이 풍부한 요원으로서 분석 및 설계단계는 현업업무에 정통하면서 전산지식을 겸비한자로 코딩및 테스팅/유지보수단계는 전산업무에 정통하면서 현업의 업무절차와 규정을 잘 아는 사람으로 측정하게 했다.

〈표 4〉 설계단계 업무별 내부특성 측정값

코드	내부특성	교 무	수 업	학 적	입 시
C01	완전성	80.0	80.0	90.0	80.0
C02	추적가능성	80.0	85.0	85.0	80.0
C03	일관성	77.1	80.0	80.0	77.1
C04	자기기술성	70.0	70.0	80.0	70.0
C05	무모순성	90.0	80.0	80.0	80.0
C08	통신절차공통성	80.0	90.0	90.0	80.0

C09	엑세스가능성	80.0	80.0	80.0	70.0
C10	엑세스제어성	80.0	70.0	80.0	90.0
C11	엑세스감사성	70.0	70.0	90.0	80.0
C12	견고성	80.0	80.0	80.0	80.0
C13	무결성	70.0	70.0	90.0	80.0
C14	모듈성	66.7	80.0	76.7	76.7
C15	단순성	72.0	76.0	76.0	76.0
C16	계측성	80.0	80.0	80.0	80.0
C18	통일성	86.7	93.3	86.7	86.7
C19	표현성	90.0	90.0	80.0	80.0
C20	계층성	80.0	80.0	80.0	80.0
C21	설명성	80.0	90.0	90.0	90.0
C24	주목성	90.0	80.0	90.0	80.0
C27	간결성	80.0	80.0	80.0	80.0
C28	선택성	70.0	90.0	90.0	90.0

2. 품질평가

가중치 생성을 위한 설문지 조사를 위해서 조사대상은 소프트웨어 개발 특수성을 감안하여 전산전문가 및 프로그램개발에 참여한 개발자로 한정하였다. 가중치 생성의 객관성을 위해서 5개 대학 전산소장에게 주품질특성및 부품질특성의 이원비교 설문을 의뢰하였고 내부특성 이원비교값은 S대학의 개발자 3명에게 의뢰 조사하였다. 조사분야는 소프트웨어 개발 전 단계에 걸쳐서 했으며, 위에서 나타낸 것과 같이 총 105종에 대해 실시했다. 조사기간은 1차로 2004년 6월21일부터 7월 20일까지 30일간 실시하였고 2차 실시는 2005년 3월 한 달간 실시하였다. 통계처리를 위해서 Expert Choice, Microsoft Excel및 SPSS를 이용하였다.

설문지에서 얻어진 주품질특성의 이원비교 자료는 Expert Choice그림의 〈Questionnaire〉 기능에 따라 입력하면 된다. 입력자료 값은 5개 대학의 설계단계의 주품질특성 6개 항목의 이원비교값을 입력하면 된다. 이러한 값이 입력이 되면 개발된 이론적 근거에 따라 가중치가 생성된다. 이때에 생성된 주품질특성 및 부품질특성의 가중치를 정리하고 정규화시킨 값을 산출하며 최종적으로 평가 결과 값을

산출한다.

〈표 4〉 설계단계 업무별 주품질특성 결과 값

구 분	교 무	수 업	학 적	입 시
기능성	17.627	15.493	18.914	17.407
신뢰성	20.488	21.455	22.642	21.471
사용성	12.914	13.185	12.947	12.773
효율성	3.669	4.400	4.219	4.219
보수성	12.329	13.111	13.243	12.756
이식성	9.610	10.732	10.744	9.853
계	76.637	78.376	82.709	78.479

3. 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 해결방안이나 기법 등을 제안함으로써 소프트웨어의 품질 향상을 기하는 것이 바람직하다.

V. 결 론

국제표준인 ISO/IEC9126을 이용하여 소프트웨어 품질을 정량적으로 측정하고 평가할 수 있는 기법을 개발한 것으로, 본 논문에서 연구된 내용을 요약하여 결론을 내리면 다음과 같다.

첫째, 소프트웨어 품질을 정량적으로 나타낼 수 있는 품질측정표를 총 113종 개발하였다. 개발단계별로 보면 분석단계 23종, 설계단계 21종, 코딩단계 38종 및 테스팅/유지보수단계 31종이다. 둘째, 가중치 생성을 위한 설문지를 총 105종 개발하여, 대학의 전산전문가들로부터 조사를 통하여 객관적인 가중치를 도출하였다. 이 가중치는 정규화 절차를 거쳐 최종 품질평가에 활용되었으며 또한 경제적으로 품질

특성을 선정할 수 있는 방법을 제시하였다. 셋째, 개발된 품질측정 및 평가기법을 실업무에 적용하여 품질판정 및 품질개선안을 권고하였다. 적용사례를 통하여 대학의 학사업무 프로그램 중 학적업무는 2등급이상으로 양호한 편이나, 수업, 입시 및 교무업무는 3등급으로 사용자(고객) 만족을 위해서 유지보수가 필요하다고 판단된다. 특히 유지보수비용을 줄일수 있는 길은 TQM 의 기본개념인 "제품을 처음부터 잘 만들어라" 또는 "프로세스 과정에 품질을 주입하라"라는 원칙하에 소프트웨어를 개발해야한다. 넷째 소프트웨어의 특성상 'soft' 한 보이지 않는 부문을 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.

참고문헌

1. 시스템 공학연구소(1997), 소프트웨어 품질평가 기술개발(1), 정보통신부
2. 정호원·황선명(1997), "소프트웨어 프로세스 심사의 이해 : SPICE 중심으로", 정보과학회지, 제17권 제1호(통권 제116호), pp.6~12.
3. 이종무·정호원(1997), "AHP를 이용한 소프트웨어 내부 품질특성의 선정방법", 정보과학회논문집(B), 제24권 제6호 pp.640~649.
4. 장영숙·권영식(1999), "설계단계에서의 소프트웨어 품질측정과평가기법 개발", 한국품질경영학회지, 제27권 제3호
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 Committe 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 Joural*, Vol,19, No.4.
8. U.S. Depart. of Defense(1985), DOD-STD-2167, Denfense system Software Development, Washington, DC.
9. U.S. Depart. of Defense(1985), DOD-STD-2168, 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), ソフウェア, 品質管理, トフイッ, 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.(1976), "A complexity Measure", IEEE Trans. on Software Eng., Vol. SE-2, No.4.

16. Halstead M.H.(1997), Elements of software science, New York : Elsevier North-Holland.

Abstract

A Quantitative Measurement and Evaluation of Software Product Quality

Im, Dae-heug

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 con promote the quality of products. To achieved that goal, we con provide a suitable the technique and method of software quality control.