소프트웨어의 보수성 측정을 위한 메트릭스의 제안과 평가

양해술*

Proposal and Evaluation of Metrics for Measurement of Software Maintenance

Hae-Sool Yang

〈요 약〉

소프트웨어에 대한 사용자의 인식이 높아감에 따라 다양한 기능의 고품질 소프트웨어의 개발이 요구되고 있으며 이러한 요구를 만족시킬 수 있는 방안들이 다양한 관점에서 연구되고 있다. 그중에서도 소프트웨어의 품질평가를 통해 그 결과를 개발자에게 피드백함으로써 소프트웨어의 품질을 향상시키려는 관점에서 품질평가 방법론 및 지원도구에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 소프트웨어의 품질에 관한 국제표준인 ISO/IEC 9126의 기능성, 신뢰성, 사용성, 효율성, 보수성, 이식성 등의 품질특성을 바탕으로 보수성에 대한 품질특성, 부특성, 내부특성의 체계에 따라 메트릭스를 정의하고 내부특성 체계에 맞추어 품질측정표를 개발함으로써 실제 품질향상을 위해 필요한 개발자 관점의 내부특성의 품질평가를 통해 사용자 관점의 외부특성에 대해 평가할 수 있는 체계를 확립하고 실제 프로젝트 업무에 대해 평가한 결과를 제시하였다.

* 한국소프트웨어품질연구소(INSQ) 소장

1. 서 론

컴퓨터가 등장하여 인간의 업무를 효율적으로 지원하게 된 이후로 전산화의 급격한 발전 추세와 더불어 소프트웨어에 대한 수요는 급속히 증가하고 있다. 이러한 이유로 소프트웨어의 수요와 공급에 균형이 깨어져 이른바 소프트웨어 위기라는 용어까지 등장하게 되었고 요구되는 소프트웨어에 대한 신속한 개발과 지원이 절실히 필요한 시점이 되었다. 이와 더불어 한번 개발된 소프트웨어는 사용자의 요구에 따라 오류의 수정이나 기능의 향상을 위해 지속적인 버전업을 반복하고 있으며 이러한 유지보수에 관련된 비용이 차지하는 비율은 초기의 개발 비용을 훨씬 초과하는 수준에 이르고 있다. 즉, 소프트웨어 생명주기 단계 중에서 유지보수 단계가 차지하는 비율이 80%를 넘게 되어 소프트웨어의 품질에 대한 중요성이 더욱 부각되고 있는 실정이다[1, 2, 5].

그러나 현재 국내에서는 아직까지 실용화하기에 적합한 수준의 품질평가 절차나 방법론에 대한 정립이 미흡한 실정이며 국제적으로도 품질특성에 관한 표준 ISO/IEC 9126이 정의되어 있으나 실용화하여 소프트웨어의 품질평가에 적용할 수 있는 세부단계의 수준까지 체계화되어 있지 못하며 현재 품질특성을 세분화하여 정량적으로 측정할 수 있는 실용적인 메트릭스에 관해 연구가 진행되고 있다[10, 11, 13, 14, 15].

국제품질표준 ISO/IEC 9126은 품질특성으로 기능성, 신뢰성, 사용성, 효율성, 보수성, 이식성의 6항목을 정의하고 있다. 본 연구에서는 품질 주특성 중의 하나인 보수성에 초점을 맞추어 보수성을 세분화한 품질부 특성인 해석성, 변경성, 안정성, 시험성에 대한 메트릭스를 정의하고 이러한 메트릭스를 관련된 내부특성과 연계하여 품질측정표를 작성하여 이용하는 효율적인 방법으로 생명주기 단계별 산출물에 대한 보수성 품질을 측정할 수 있는 방안을 기술하였다. 즉, 본 논문의 제2장에서는 관련 연구와 문제점에 대해 기술하였으며, 제3장에서는 보수성을 측정하기 위한 메트릭스와 관련 내부특성에 관한 체크리스트화를 통한 품질측정 방법을 다루었고, 제4장에서는 품질측정표를 이용한 실제 업무의 품질 측정 및 평가 사례를 기술하였다. 끝으로 5장에서는 본 연구의 결과와 앞으로의 연구 과제를 기술하였다.

2. 관련 연구와 문제점

소프트웨어에 관한 품질 의식은 일반적으로 다음과 같은 단계를 거쳐 변화된다[11, 15].

① 품질무의식 시대 : 품질을 의식하지 않고 소스 코드 개발만을 중시
② 품질문제 표면화 시대 : QC 사이클 활동의 목적이나 QC 절차를 이해
③ 체계적 품질관리 시대 : 본래 업무의 개선에 몰두
④ 자발적 품질관리 시대 : 스스로 업무의 개선에 몰두하여 성과를 올리는 시기

그러나 국내에서는 소프트웨어 품질관리체계에 대해 지금까지 생명주기 단계별로 체계적인 지원 시스템이나 품질관리 방법론이 제대로 정립되어 있지 못한 것이 현실이다. 본 장에서는 소프트웨어 생명주기 단계와 관련된 품질 관련 연구결과들을 살펴보고 기존의 연구에서 나타나는 문제점들을 기술하였다.

2.1 소프트웨어의 품질관련 연구

소프트웨어 생명주기 단계와 관련된 품질관리 방법론 및 품질평가 메트릭스에 대한 연구결과는 다음과 같다[13, 14].

- 소프트웨어 제품 평가를 위한 품질특성에 관한 국제 표준으로서 ISO/IEC 9126과 관련된 품질부특성, 내부특성의 구축 및 상호관계 확립
- 사용자 및 관리자 관점의 외부특성으로서 품질특성과 품질부특성, 개발자 관점으로서 내

부특성에 대한 메트릭스를 바탕으로 소프트웨어의 품질향상을 목적으로 하는 메트릭스의 구체화 및 효율적인 평가를 위한 품질측정표 구축에 관한 연구

- 구현단계의 산출물인 프로그램 소스에 대한 품질평가 모델의 구축 및 자동화 도구 개발 추진
- 각 생명주기 단계에서 산출되는 개발산출물에 대한 테스트 및 품질평가 결과 발견되는 장해 데이타를 축적 관리함으로써 품질관리의 효율화를 추진하는 장해관리 시스템에 관한 연구

2.2 소프트웨어 품질관리 분야의 문제점

소프트웨어 생명주기 단계와 관련된 지금까지의 연구 결과들은 아직까지 개선의 여지가 많고 실제 업무에 적용하여 실용화하기에는 미흡한 점이 발견되고 있다. 이러한 품질관리 분야에 대한 연구동향과 문제점들을 품질특성 모델, 메트릭스, 생명주기 공정, 상위공정의 품질평가에 중점을 두고 살펴보면 다음과 같다[12].

(1) 국제표준 및 품질특성에 관한 모델

소프트웨어 제품에 관한 품질특성으로서 국제표준이 제정되어 있으나 광의의 개념적인 정의 수준에서 머물고 있으며 실제 업무에 실용화하여 적용하려면 지속적인 연구를 통해 실용적인 세부사항들을 개발하고 시험 적용을 통해 객관성과 타당성을 검증할 필요가 있다. 현재 실용가능한 수준의 품질특성 모델인 경우에도 프로그램 소스에 대한 복잡성이나 이해용이성, 모듈간의 관련성 등 대부분이 구현단계 산출물인 프로그램 소스에 관련된 일부분의 특성을 평가하는 것이고 소프트웨어 생명주기 전체에 걸쳐 단계별로 체계적인 평가를 통해 실용적인 품질관리를 추진할 수 있는 구체화된 모델은 개발되지 않고 있다.

(2) 메트릭스

소프트웨어 품질특성 모델의 대부분이 구현과 테스트 공정을 고려한 모델이며 국제표준의 품질특성에 관련된 메트릭스의 경우에는 개략적인 수준의 메트릭스만이 정의되어 있으며 실제 생명주기 단계별 개발산출물에 적용하여 평가하기에는 적합하지 않다. 실제 품질특성 모델의 대부분은 다음과 같은 사항에 중점을 두고 있다.

- 프로그램의 시험공수나 버그수에 관련된 것
- 소스레벨의 기능 사이즈, 구조, 이해 용이성 및 복잡성 등의 메트릭스 설정

(3) 초기공정의 품질

소프트웨어의 품질은 생명주기 단계의 초기공정인 분석·설계단계에 크게 좌우되며 이들 초기단계에서 발생된 문제점은 후기단계로 갈수록 더 많은 오류를 유발할 가능성이 있을뿐만 아니라 뒤늦게 발견되었을 경우 후기단계의 오류에 비해 유지보수에 더 많은 비용이 소요된다. 이러한 초기공정의 산출물인 명세서의 품질에 대해서는 DR(Design Review) 등으로 점검하는 경우가 대부분이고 체계적이며 정량적인 품질평가 방법을 적용하고 있지 않다.

(4) 생명주기 각 단계간의 연계

대부분의 품질특성 모델과 관련된 메트릭스는 생명주기 각 단계별 단일공정에서 사용하도록 되어있으며 다른 단계와의 연관관를 고려하고 있지 못할 뿐만 아니라, 특정 공정의 산출물이 다음 공정의 산출물에 미치는 영향에 대한 검토가 충분하지 못하다.

(5) 프로세스의 능력 평가

소프트웨어를 개발하기 위한 프로세스의 능력을 정량적으로 평가하기 위해 필요한 평가 척도와 측정 방법으로 구성된 프로세스 능력평가 메트릭스에 대해 명확히 정의되지 않고 있다. 즉, 지금까지 소프트웨어 프로세스 능력은 소프트웨어

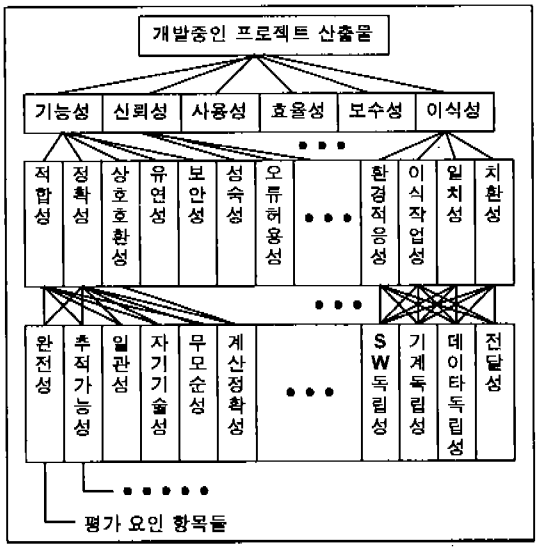
제품의 품질을 평가한 결과를 기초로 간접적으로 평가되어 왔다. 현재, 소프트웨어 프로세스가 가지는 종합적인 능력을 직접적으로 평가, 개선하도록 하는 방법으로 CMM, SPICE, Bootstrap, ISO9000-3의 모델이 활용되고 있다.

3. 보수성 메트릭스와 품질측정표

프로그램의 보수성을 측정하기 위한 메트릭스의 개발과 관련 ISO/IEC 9126의 품질특성 6항목 중 하나인 보수성을 바탕으로 보수성을 세분화한 품질부특성과 관련 내부특성을 체계화할 수 있다.

품질특성은 크게 외부특성과 내부특성으로 분류할 수 있으며, 외부특성은 사용자 관점 또는 제품평가를 의식한 특성이고, 내부특성은 개발자 관점 또는 개발 프로세스의 평가를 의식한 특성이다.

즉, 외부특성이란 소프트웨어의 내부(예를 들면 구조나 명령어 등)를 살펴볼 수 없고, 소프트웨어를 이용하기 위한 이해 정도를 나타내는 소프트웨어 특성이며, 내부특성이란 복잡한 형태의 소프트웨어 내부를 살펴봄으로써 초기에 이해할 수 있는 소프트웨어 특성이다.

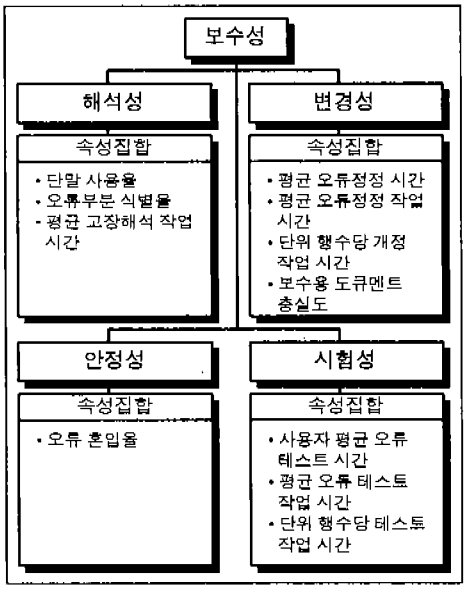


<그림 1> 품질특성, 부특성, 내부특성의 관계

국제표준은 품질특성 6가지의 항목을 세분하여 품질부특성 21항목을 정의하고 또한, 관련 내부특성 40항목을 구성하여 품질부 특성과의 관련성을 정의하고 있다. <그림 1>은 품질특성, 품질부특성, 내부특성간의 관계를 나타내고 있다. 이러한 관계에 따라 품질부 특성의 메트릭스로부터 직접 품질특성을 평가하거나, 내부특성을 평가한 후 <그림 1>의 관계에 따라 상향식으로 품질특성을 산출한다.

3.1 외부특성에 관한 보수성의 메트릭스

ISO/IEC 9126에는 외부특성에 관한 품질특성으로서 기능성, 신뢰성, 사용성, 효율성, 보수성, 이식성을 정의하고 있으며 이들을 좀더 세분하여 품질부특성을 정의하고 있다. 이중에서 본 연구의 주제인 보수성은 <그림 2>와 같이 사양화된 개정을 처리하기 위해 필요로 하는 노력을 나타내는 속성의 집합을 의미하며 그 부특성으로 해석성, 변경성, 안전성, 시험성으로 구성되며 그 정의는 다음과 같다.



<그림 2> 보수성 메트릭스의 구조

① 해석성 : 고장의 원인 분석이나 결함분석 및 개정 가능한 부분의 정도와 사용 시간을 나타내는 소프트웨어 속성
② 변경성 : 개정을 위한 장해제거나 환경 변경에 필요한 노력을 나타내는 소프트웨어 속성
③ 안정성 : 개정에 의해 예기되는 효과의 포함성을 나타내는 속성
④ 시험성 : 소프트웨어의 타당성 검증에 필요한 노력을 나타내는 속성

3.1.1 해석성의 메트릭스

해석성은 고장의 원인이나 결함 분석, 또는 개정 가능한 부분의 정도와 사용 시간을 나타내는 속성의 집합으로 <표 1>과 같이 메트릭스를 정의한다.

<표 1> 해석성의 메트릭스

속성집합	정 의	측정식
1.단말 사용율	일정 시간내에서 단말을 사용한 시간이 차지하는 비율	• Rtu = Tt/Ttp Tt : 단말 사용시간 Ttp : 단말 사용 가능 시간
2.오류 부분 식별율	보수자의 고장해석 결과 평가대상 소프트웨어에 오류가 있다고 판정된 고장 건수에 대하여 엔드유저가 올바르게 오류부분을 식별가능한 고장 건수의 비율	• Rer = S_{ec}/N_d S_{ec} : 오류 정정 시간의 합계 N_d : 고장 건수
3.평균 고장 해석 작업 시간	고장 연락을 받은 보수자가 고장의 원인을 특정하거나, 변경가능한 부분을 수정시 요구되는 평균 작업 시간	• Tea = S_{ea}/N_e S_{ea} : 고장 해석작업 시간의 합계 N_e : 대상 소프트웨어의 고장건수

3.1.2 변경성의 메트릭스

변경성은 개정을 위한 장해 제거 또는 환경 변경에 필요한 노력을 나타내는 속성으로 <표 2>와 같은 메트릭스들을 정의하였다.

<표 2> 변경성의 메트릭스

속성집합	정 의	측정식
1.평균 오류정정 시간	보수자가 고장 연락을 받으면서부터 처리하여 소프트웨어 발송까지의 평균 경과 시간	• Tmec = S_{ec}/N_e S_{ec}:오류 정정 시간의 합계 N_e:고장 건수
2.평균 오류정정 작업시간	엔드 유저에 의해 고장 대책의 경과 시간의 평균 값	• Tmecw = S_{ecp}/N_e S_{ecp}:오류 정정 실시 시간의 합계 N_e:고장 건수
3.단위행수당 개정 작업시간	갱신 또는 추가한 소스 코드의 단위 행수당 개정 작업 시간	• Tul = S_{rt}/N_{cs} S_{rt}:개정 작업 시간의 합계 N_{cs}:변경 소스코드행수
4.보수용 도큐멘트 충실도	보수에 사용가능한 개발용의 도큐멘트가 준비되어 있는 비율	• Ddf = N_{dp}/N_{as} N_{dp}:도큐멘트페이지 수 N_{as}:모든 소스코드행수

3.1.3 안정성의 메트릭스

안정성은 개정에 의해 예기되는 효과의 포함성을 가지는 속성으로 <표 3>과 같은 메트릭스를 정의하였다.

<표 3> 안정성의 메트릭스

속성집합	정 의	측정식
오류 혼입율	소프트웨어의 개정에 의해 새로운 오류가 발생하는 비율	• Rei = N_{ei} / N_{csc} or N_{ei} / N_{ce} N_{ei} : 오류 혼입 건수 N_{csc} : 변경 소스코드 행수 N_{ce} : 정정한 오류 건수

3.1.4 시험성의 메트릭스

시험성은 소프트웨어의 타당성 검증에 필요한 노력을 나타내는 속성으로 <표 4>와 같은 메트릭스들을 정의하였다.

<표 4> 시험성의 메트릭스

속성집합	정 의	측정식
1.사용자 평균 오류 테스트 시간	오류 정정에서 사용자의 확인 테스트에 요구되는 평균 작업 시간	• Tmet = S_{et}/N_e S_{et} : 사용자의 오류 테스트 작업시간의 합계 N_e : 오류 건수
2.평균 오류 테스트작업 시간	오류 정정에서 정정 실시후에 처리되는 확인 테스트에 요구되는 평균 작업 시간	• Tmew = S_{et}/N_e S_{et} : 오류 테스트 작업 시간의 합계 N_e : 오류 건수
3.단위행수당 테스트작업 시간	오류 정정 또는 개조에서 갱신 또는 추가된 소스 코드의 단위 행수당 테스트 작업 시간	• Ttw = S_{tt}/N_{sc} S_{tt} : 테스트 작업 시간의 합계 N_{sc} : 변경 소스코드 행수

품질특성 / 품질부특성 / 내부특성	보수성			
	해석성	변경성	안정성	시험성
1. 완전성				
2. 추적가능성	◎	◎	○	△
3. 일관성	◎	◎	○	○
4. 자기기술성	◎	◎		○
⋮	⋮	⋮	⋮	⋮
14. 모듈성	◎	◎	○	◎
15. 단순성	◎	◎	○	○
16. 계측성	◎	◎	○	◎
⋮	⋮	⋮	⋮	⋮
27. 간결성	◎	◎		
⋮	⋮	⋮	⋮	⋮
35. 확장성	○	○	△	△
36. 제품관리성	△	◎		
37. S/W 독립성	○	○		○
38. 기계 독립성	○	○		○
39. 데이타 독립성				
40. 전달성				

◎ : 강한 관계 ○ : 보통 관계 △ : 약한 관계

<그림 3> 보수성에 대한 외부특성과 내부특성의 관계

3.2 내부특성에 관한 보수성의 메트릭스

본 절에서는 40개의 내부특성 중 품질특성인 보수성에 관련된 항목들을 살펴보았다. 보수성에 관련해서 외부특성과 내부특성과의 관계를 <그림 3>에 나타내었다. 보수성에 관련된 품질부특성인 해석성, 변경성, 안정성, 시험성은 <그림 3>과 같이 각각 추적가능성, 일관성 등의 다수의 내부특성과 관련성을 갖는다. 관련성의 강도는 강한 관계, 보통 관계, 약한 관계로 구분하여 표시하였다.

강도의 강약에 따라 내부특성을 이용하여 품질을 평가하는 과정에서 관련 내부특성의 가중치를 결정하게 된다.

<표 5> 추적가능성에 관한 메트릭스

속성집합	정 의	측정식
1. 기능 전개율	전단계의 설계서에 기술되어 있는 기능이 현단계의 설계서에 전개되어 있는 비율	• Rfd = Ncs/Nps Ncs:현단계의 설계서에 기술되어 있는 기능 항목수 Nps:전단계 설계서에 기술되어 있는 기능항목수
2. 검증 가능성	어떤 공정에서 기술된 사항이 다음 공정에서도 기술되어 있는 것을 체크한 조건 중 실제로 추적 가능한 비율	• Pv = Ntp/Nc Ntp:추적가능 건수 Nc:체크 항목수
3. 요구기능 추적율	사용자의 요구에 대하여 기능사양서의 어느 기능에서 실현되는지 대응 가능한 요구의 비율	• Rdp = Ncc/Nud Ncc:대응된 요구수 Nud:사용자의 모든 요구수
4. 기능/ 프로그램 추적율	기능사양서에 기술한 기능에 대하여 프로그램의 어느 모듈에서 실현되어 있는지 대응 가능한 기능의 비율	• Rfp = Ncf/Naf Ncf:대응된 기능수 Naf:기능사양서의 모든 기능수
5. 데이타 강도	물리 데이타 구조가 논리 데이타 구조와 잘 대응되고 있는지에 대한 비율	• Pd = Nld/Nad Nld:논리적 강도의 데이타 수 Nad:모든 데이타 수

보수성에 관련된 내부특성들에 대해 메트릭스를 정의함으로써 품질평가에 활용할 수 있다.

즉, 보수성에 관련된 품질부특성별로 관련 내부특성들에 대한 메트릭스를 이용하여 품질을 측정하고 관련 강도에 따른 가중치를 적용하여 내부특성을 이용한 간접적인 방법으로 외부특성인 품질특성을 평가하는 것이다. 개발자 중심의 품질평가 메트릭스인 내부특성에 관한 메트릭스 중 추적가능성의 예를 <표 5>에 나타내었다.

3.3 내부특성 메트릭스의 품질측정표

해석성을 측정하기 위해서는 관련된 내부특성에 대한 품질측정표를 이용한다. 관련된 내부특성은 <그림 3>에 나타나 있는것처럼 추적가능성, 일관성, 자기기술성, 모듈성, 단순성, 계측성, 간결성, 확장성, 제품관리성, 소프트웨어 독립성, 기계독립성, 데이타독립성, 전달성의 11가지이다.

이러한 내부특성들에 대한 메트릭스를 바탕으로 상세화, 구체화하여 체크리스트를 구성함으로써 편리한 방법으로 효율적인 품질 측정 및 평가를 할 수 있다. 보수성에 관련된 메트릭스 중에서 추적가능성에 관한 체크리스트를 <표 6>에 나타내었다. 체크리스트의 각 세부항목은 해당 요소데이타로 이루어진 계산식에 의해 평가된다. 요인항목과 세부항목은 중요도에 따라 가중치를 가지며 관련 항목들의 가중치 합은 1이 된다. 세부항목과 해당 가중치를 곱한 결과를 합산한 것이 요인항목에 대한 평가 결과가 되며, 요인항목의 값과 해당 가중치를 곱한 결과를 합산한 것이 내부특성인 해석성의 평가 결과가 된다.

개발 산출물로부터 직접 평가할 수 있는 항목을 요소데이타라고 하며 각 세부항목의 값은 요소데이타로 구성된 수식을 이용하여 산출된다.

추적가능성의 각 세부항목에 관한 요소데이타와 계산식은 <표 7>과 같으며 계산식에 의해 세부항목은 0과 1사이의 값을 가지고 평가 결과를 백분율로 변환하여 세부항목의 평가 결과가 계산된다.

<표 6> 추적가능성의 품질측정표

일관성			
요인 항목	세 부 항 목	평가(예)	비 고
1. 설계서의 표준/규약을 일관성있게 준수하고 있는가?(0.3)	PDi1 : 코딩된 프로그램이 표준/규약에 어느정도 일관성있게 제공되었는가? (0.6)	79.5	요소데이타와 계산식으로 도출(표 7 참조)
	PDi2 : 코딩된 프로그램 표준/규약을 준수하는 기능들이 어느정도 제공되는가? (0.4)	83.2	상 동
	계 : 0.6*79.5 + 0.4*83.2 = 80.98		
2. 코딩된 모듈과 자료의 명칭이 일관성이 있는가?(0.3)	PDi3 : 모듈과 자료에 명칭을 부여하기 위한 표준 지침이 준비되어 있는가? (0.6)	75.4	상 동
	PDi4 : 모듈과 자료의 명칭 부여에 대한 일관성은 어느 정도인가? (0.4)	73.7	상 동
	계 : 0.6*75.4 + 0.4*73.7 = 74.72		
3. 오류통보와 응답 형식에 일관성이 있는가?(0.2)	PDi5 : 오류통보 메시지 관리를 위한 기능이 어느정도 제공되는가? (0.6)	73.8	상 동
	PDi6 : 오류통보에 대한 응답형식의 일관성은 어느정도인가? (0.4)	72.5	상 동
	계 : 0.6*73.8 + 0.4*72.5 = 73.28		
4. 조숙기술 요령에 일관성이 있는가? (0.2)	PDi7 : 주석기술 요령에 대한 표준 지침이 준비되어 있는가? (0.6)	77.2	상 동
	PDi8 : 주석기술의 일관성은 어느정도인가? (0.4)	81.3	상 동
	계 : 0.6*77.2 + 0.4*81.3 = 78.84		
총 계 : 0.3*80.98+0.3*74.72+0.2*73.28+0.2*78.84 = 77.13			

(주) 괄호 안의 수는 각 항목별 가중치

<표 7> 추적가능성의 요소데이타와 계산식

세부항목	요소데이타를 이용한 계산식
PDi1	표준/규약을 일관성있게 지원하는 기능수 / 구현된 기능수
PDi2	표준/규약을 준수하는 기능수 / 구현된 기능수
PDi3	표준지침의 보유 정도
PDi4	일관성있게 명칭이 부여된 기능수 / 구현된 모든 기능수
PDi5	오류통보 메시지 관리 기능수 / 관리에 대한 모든 기능수
PDi6	오류 메시지를 일관성있게 제공하는 기능수 / 오류 메시지를 제공하는 기능수
PDi7	주석 기술에 대한 표준 지침의 제공 여부
PDi8	일관성있게 주석이 된 기능수 / 주석이 된 모든 기능의 수

4. 보수성의 측정과 평가 사례

보수성 측정을 위한 대상 업무는 H사에서 개발하고 있는 프로젝트로서 주요 목적은 회사 전체의 토탈시스템, 고객 목표에 부응하는 현장 중심 시스템, 지역/본사 연결 재무 종합 시스템의 구축 등이다. 따라서 본 장에서는 위와 같은 목적에 부합하면서 보수성에 관한 내부특성 메트릭스의 타당성을 입증하기 위하여 개발 소스 코드를 이용하여 품질을 측정하고 평가한 사례를 기술하였다.

4.1 평가 대상 업무의 구성

개발중인 프로젝트는 수입관리 부문, 지불관리 부문, 자금관리 부문, 회계관리 부문, 고정자산관리 부문 등으로 구성되어 있으며 이러한 각 부문에 대한 업무별 구현단계 산출물에 대한 품질측정과 평가를 수행하였다.

4.2 업무별 평가 결과

평가의 대상이 되는 프로젝트가 재무 업무 전산화에 관한 통합적이고 신뢰성을 요구하는 시스템인 만큼 본 연구에서 제안한 메트릭스와 품질측정표 즉, 범용적인 시스템 품질평가 요소들뿐만 아니라 개발 목적에 적합한 평가요소들을 추가하여 평가하였다. 본 절에서는 H사의 프로젝트에 대해 각 업무별로 보수성 품질측정 결과를 기술하고 문제점과 해결 방안을 예시하였다.

4.2.1 내부특성의 평가 결과

먼저 내부특성에 대한 품질 측정 및 평가를 위해 보수성과 관련된 내부특성에 대한 요소데이타를 측정하고 관련 내부특성 품질측정표의 평가체계에 따라 <표 8>과 같은 측정 결과를 얻을 수 있었다. 측정 결과를 통해 각 업무별 보수성 관련 내부특성에 대한 득점 및 상대적으로 취약한 내부특성을 파악할 수 있다.

특히, 전체 업무에 대해 내부특성인 계측성을 살펴보면 고정자산만인 70점에 이르고 있으며 그 외의 업무들은 매우 취약한 결과를 보이고 있을 뿐만 아니라 전체 내부특성중 가장 낮은 점수대를 보이고 있음을 알 수 있다. 반면 내부특성인 추적가능성의 경우 고정자산 업무를 제외하고 전체적으로 상위 수준에 이르고 있음을 보이고 있다.

<표 8> 보수성 관련 내부특성의 평가 결과

특성	내부특성	회계	지불	수입	세무	고정자산
보수성	추적가능성	86	84	86	74	70
	일관성	71.4	74	78	72.4	80
	자기기술성	78.4	79.6	81.4	77.6	77.4
	모듈성	79.2	81.6	81.8	88.8	83.4
	단순성	70.8	74.6	75.8	76.2	77.6
	계측성	64	64	64	60	70
	간결성	68.8	74	76	78.4	86.4
	확장성	68.6	72.8	68.8	77	77
	제품관리성	73	73	77	77	73
	S/W독립성	62.8	71.6	74.8	68.8	77.6
	기계독립성	64	70	73.6	70.4	77.6

4.2.2 품질부특성과 품질특성의 결과 집계

<표 9>는 보수성 관련 내부특성에 대한 측정결과를 바탕으로 내부특성과 품질부특성의 관계에 따라 품질부특성중 해석성에 대한 평가 결과를 보이고 있다. 각 내부특성에 포함된 수치는 내부특성과 품질부특성의 관련성 정도에 따른 가중치를 나타내고 있으며 가중치와 득점을 이용하여 다음과 같은 계산에 의해 품질부특성의 평가 결과가 산출된다.

$$\text{업무별품질부특성} = \sum_{i=1}^{n}(w_i * v_i)$$

여기서, w_i : 내부특성 i의가중치,

v_i : 내부특성 i의평가값,

n : 세부항목의 수

$$\sum_{i=1}^{n} g_i = 1$$

평가를 통해 각 업무별로 최저점수, 최하점수 및 분포 상황을 알 수 있다. 회계관리 업무의 경우 관련 내부특성 중 추적가능성이 상대적으로 높은 점수를 보이고 있으며 소프트웨어독립성이 최하 점수를 기록했음을 알 수 있다.

<표 9> 품질부특성에 대한 집계표(해석성)

부특성	내부특성	회계	지불	수입	세무	고정자산
해석성	추적가능성(0.1)	86	84	86	74	70
	일관성(0.1)	71.4	74	78	72.4	80
	자기기술성(0.1)	78.4	79.6	81.4	77.6	77.4
	모듈성(0.1)	79.2	81.6	81.8	88.8	83.4
	단순성(0.1)	70.8	74.6	75.8	76.2	77.6
	계측성(0.1)	64	64	64	60	70
	간결성(0.1)	68.8	74	76	78.4	86.4
	확장성(0.08)	68.6	72.8	68.8	77	77
	제품관리성(0.06)	73	73	77	77	73
	S/W독립성(0.08)	62.8	71.6	74.8	68.8	77.6
	기계독립성(0.08)	64	70	73.6	70.4	77.6
	계	71.9	74.7	76.3	77.4	77.4

<표 10>은 품질부특성의 결과를 바탕으로 업무별 품질부특성들의 평가 결과와 품질특성인 보수성의 집계 결과를 보이고 있다. 각 부특성에 포함된 수치는 품질부특성과 품질특성간의 관련성 정도에 따른 가중치이며 가중치와 득점을 이용하여 품질부특성의 경우와 같은 방법으로 계산하여 품질특성의 평가 결과를 산출한다. 결과적으로 고정자산 업무에 대한 개발산출물의 평가 결과가 가장 높고 회계관리 업무가 상대적으로 미약함을 알 수 있다. 회계관리 업무의 평가 결과 중에서도 변경성이 상대적으로 낮은 점수를 보이고 있으며 이러한 결과를 나타낸 원인을 찾기 위해 변경성에 관련된 내부특성들의 점수 분포상황을 분석하면 그 원인이 되는 내부특성의 문제점을 찾을 수 있다.

<표 10> 품질특성에 대한 집계표

특성	부특성	회계	지불	수입	세무	고정자산
보수성	해석성(0.3)	71.9	74.7	76.3	74.7	77.4
	변경성(0.3)	72.2	74.8	76.5	74.9	77.3
	안정성(0.2)	73.7	75.4	76.3	74.6	76.3
	시험성(0.2)	72.1	75.1	76.5	74.6	77.7
	집계	72.4	74.9	76.4	74.7	77.2

본 평가에서는 80점을 합격수준으로 정하고 수준에 미치지 못하는 특성에 대해서는 문제점과 개선방안을 예시하여 품질향상을 도모할 수 있도록 하였다. 품질평가를 통해 전체적으로 품질수준이 다소 미흡하다는 것을 알 수 있었으며 앞으로 반복적인 평가를 통해 평가 기준의 타당성을 검증해야 할 필요가 있다.

4.3 문제점과 개선방안

<표 11> 문제점과 개선 방안의 일부

내부특성	요인항목	세부항목(문제점)	개 선 방 안
일관성	표준/규약의 일관성있는 지원	코딩된 프로그램이 일반적인 구조적 프로그래밍 기법을 준수하지 못하고 있다.	제어흐름의 복잡성을 최소화하도록 goto문을 제거하여 단순화하고 프로그램을 읽기 쉽도록 블록별로 명시적인 구분을 추가한다.
	명칭부여의 일관성	명칭부여를 위한 표준 지침이 미흡하고 일관성이 부족하다.	모듈명, 자료명에 대한 명칭부여 규칙을 문서화하여 명시하고 관련 명칭에 대해 일관성있는 명칭을 부여하도록 수정한다.
	⋮	⋮	⋮
자기기술성	주석기술 지침 준수	주석 기술 지침을 명확히 준수하고 있지 못하며 주석 기술량이 부족하여 프로그램 이해에 어려움이 따른다.	주석 기술 지침에 따라 기술하는 위치, 방법 등을 준수하고 프로그램 및 문서의 이해를 돕도록 충분한 양의 주석을 기술하도록 한다.
	모듈색인 충실도	모듈에 관한 색인이 체계적이지 못하고 누락된 경우가 있어서 필요한 모듈을 검색하기 어렵다.	모듈에 관한 색인을 순서에 맞게 체계적으로 기술하고 프로그램 사양서 등에 기술된 모든 모듈에 대해 색인을 충실히 기술함으로써 필요시 쉽게 참조할 수 있도록 한다.
	⋮	⋮	⋮

제안한 내부특성의 체크리스트에 따라 업무별 보수성을 측정한 결과 각 품질특성별, 업무별 평가 결과가 전체적으로 합격수준으로 설정했던 80점에 미치지 못하는 미흡한 수준임을 알 수 있다.

품질평가는 결과를 산출하는 것 뿐만 아니라 결과에 대한 문제점을 분석하여 개발자에게 제시함으로써 품질을 향상시키는 것이 목적이므로 <표 11>과 같이 내부특성의 메트릭스에 관련해서 평가 대상 소프트웨어에 나타난 문제점과 개선방안을 예시하였다.

5. 결 론

소프트웨어는 요구정의로부터 개발 종료에 이르기까지 제반 사항을 구현하는 것도 중요하지만, 이미 개발이 완료된 소프트웨어에 대한 수정보완의 차원에서 유지보수도 매우 중요하다고 할 수 있다. 더구나 개발이 완료되어 사용되고 있는 소프트웨어라 해도 오류가 발견된다거나 기능을 향상시켜 다음 버전을 개발할 필요가 있는 경우 지속적인 수정보완은 매우 중요한 관건이다.

그러므로 소프트웨어의 유지보수 비용이 전체 생명주기에 소요되는 비용의 80%를 초과하는 현 시점에서 비용을 절감할 수 있는 방안을 강구한다는 차원에서도 유지보수를 용이하게 할 수 있는 방법을 찾아야 할 필요가 있다.

따라서, 본 연구에서는 품질특성인 보수성에 대한 품질평가를 통해 문제점을 발견하여 개발자에게 피드백하고 수정보완함으로써 유지보수에 관한 용이성을 확보할 수 있다는 차원에서 보수성의 품질부특성인 해석성, 변경성, 안정성, 시험성에 관한 메트릭스와 그 관련 내부특성들에 대한 메트릭스를 구축하여 품질평가에 활용할 수 있도록 함으로써 품질향상을 도모하고자 하였다.

또한, 효율적인 품질평가를 위해 개발자의 관점인 품질 내부특성에 관한 메트릭스를 상세화하고 구체화하여 품질측정표를 작성함으로써 용이하게 품질평가를 할 수 있도록 하였으며 구축한 품질측정표를 이용하여 실제 개발 업무에 대한 평가 과정과 결과를 제시하고 평가 결과의 분석을 통해 발견할 수 있는 문제점과 품질 향상을 위한 개선 방안을 제시하였다.

향후 연구 과제로 보수성에 관련된 메트릭스를 이용하여 품질평가를 반복실시함으로써 문제점을 발견하고 개선하는 과정을 통해 메트릭스를 검증해야 하며 또한 메트릭스의 값을 구하는 과정에서 주어지는 가중치에 대한 객관성에 대해서도 연구가 병행되어야 할 것이다.

〈참고문헌〉

[1] Roger S. Pressman, *Software Engineering*, 3rd. Ed., McGraw-Hill International Editions, 1992.

[2] K., John Gough, *Syntax Analysis and Software Tools*, Addison Wesley, 1988.

[3] L. A. Belady, M, M. Lehman, "A model of large program development," IBM System Journal, Vol. 15, No. 3, pp. 225-252, 1976.

[4] Brian W. Kernighan, *The C Programming Language*, 2nd. Ed., Prentice Hall, 1988.

[5] Zuse, H, *Software Complexity : Measure and Methods*, New York, Walter de Grutyer, 1991.

[6] Uwe Meyer, "Techniques for Partial Evaluation of Imperative Languages," Proc. Symposium on Parial Evaluation and Semantics-Based Program Manipulation, pp. 94-105, Jun 1991.

[7] Thadhani, A. J., "Factors Affecting Programmer Productivity during Application Development," IBM Systems Journal, Vol. 23, No. 1, pp. 19-35, 1984.

[8] Ohba, M., "Software Quality = Test Accuracy × Test Coverage," Proc. of 6th ICSE Tokpo, IEEE, pp. 287-293, 1982.

[9] Lowell Jay Arthur, *Measuring Programmer Productivity and Software Quality*, John Wiley & Sons, Inc., 1985.

[10] 吉澤. 東. 片山, ソフトウェアの 品質管理と 生産技術, 日本規格協會, 1988.

[11] 菅野文友, ソフトウェアの品質管理, 日科技連出版, 1987.

[12] 정호원, 양해술, ISO 9000 시리즈와 소프트웨어 품질시스템(上), 하이테크정보, 1993. 3.

[13] 양해술, 소프트웨어의 품질평가 체계 및 자동화 도구의 개발, 한국통신 장기기초연구과제, 최종보고서, 1995.

[14] 양해술, 품질관리 방법론 및 지원도구의 개발, 과학기술처 STEP2000 지원과제 제3차년도 최종보고서, 1997.

[15] 양해술, "소프트웨어 품질측정 기록 및 지원 툴킷 개발", 시스템공학연구소(SERI) 위탁연구과제, 제1차년도 중간보고서, 1997. 6.