□ 研究論文 □

韓國軍事運營分析學會誌
第 22 卷, 第 2 號, 1996. 12. 31

소프트웨어 품질측정을 위한 소프트웨어 품질메트릭 방법론과 적용 연구 (A Study on Software Quality Metric Methodology and Application for Software Quality Measurement)

이 성 기*

Abstract

Research issues in software engineering in recent may be object oriented methodology and software quality. Since Halstead has proposed metric-software science in 1977, software quality area has been studied in steady but inactively until 1980s. As international standards such as ISO 9000-3, 9126 were enacted in 1990s early, interest in software quality is increased but many problems such as how to validate metric, measure quality or apply metric are remained.

This paper proposes software quality metric methodology which software developer or project manager can use in measuring quality and validating metric during software development. The methodology is classified by several phases: establishment of quality requirement, identification of quality metric, data collection, metric implementation, metric validation. In order to show its applicability, test program, metrics and data are applied to each phase of the methodology.

Consideration of this methodology as a methodology for software quality measurement similar to development methodology for software development is needed.

Key word : software quality, measurement, metric methodology, metric validation

* 국방정보체계연구소

1. 서론

최근 소프트웨어 공학분야에서는 객체지향 개발방법론과 품질측정 및 관리방법에 관한 연구가 활발하다. 객체지향 개발방법론은 실세계와 유사한 모델링, 재사용과 유지보수성이 향상된다는 기술적인 측면과 전통적으로 적용해오던 구조적 방법론에서 벗어나고자 하는 개발자, 관리자의 의지가 상응하여 최근 2-3년간 크게 관심의 대상이 되고 있다. 한편, 품질측정 및 관리방법은 이 분야에 대한 연구의 실효성이 인지되지 못한 상황에서 1980년대 말이후 침체되었으나, 1991년에 소프트웨어 품질관리(ISO 9000-3), 소프트웨어 품질특성(ISO 9126)에 관한 국제표준이 제정되면서 다시 관심이 증대되고 있다. 그러나 소프트웨어 품질에 관한 연구는 객체지향 개발방법론에 비해 연구의 대중화가 느리고 실효성과 필요성에 대한 의문이 남아 있다.

본 연구는 소프트웨어 품질에 관한 연구가 일반화되지 못하고 몇몇 학자나 연구자의 관심대상에 그치는 근본적인 이유중 하나가 소프트웨어 품질측정에 대한 불필요성보다는 소프트웨어 품질을 어떻게 측정하며, 측정결과를 어떻게 활용할 수 있는 것인가 등과 같은 기본원리나 활용방법에 대한 부재에 기인한다고 판단하였다. 실제로 기존의 연구는 주로 측정방법과 결과분석 위주로 수행되었고 측정활동을 수명주기관점의 방법론으로 발전시키지는 못하였다. 결과적으로 소프트웨어 개발을 위해서는 개발자나 관리자가 적용할 수 있는 개발방법론이 있으나 품질관점에서는 적용할 수 있는 방법론이 부재한 실정이다.

이를 개선하기 위해 본 연구에서는 기존의 연구결과를 토대로 소프트웨어 개발자나 관리자가 품질측정활동시 적용할 수 있는 소프트웨어 품질매트릭 방법론을 정립하고 그 방법론의 적용성을 중점적으로 연구하였다. 방법론은 품질에 대한 요구사항 설정, 품질매트릭 식별, 자료수집, 매트릭 구현(계산), 매트릭 검증(Validation) 단계로 구분하였으며, 적용성을 보이기 위해 가상의 소프트웨어와 자료를 이용하여 품질매트릭 방법론을 적용시켜 보았다.

2장에서 소프트웨어 품질측정의 기본개념을 설명하고, 3장에서 품질매트릭의 기본구조(Framework)에 대해 기술한다. 4장에서는 소프트웨어 품질매트릭 방법론에 대해 논의하고, 5장에서는 방법론의 적용방법을 설명한다.

2. 소프트웨어 품질측정 기본개념

2.1 측정이론

우선 소프트웨어 품질매트릭에 관련된 기본이론과 개념을 살펴 본다.

측정(Measures)는 어떤 측정대상의 속성에 대해 객관적인 수치를 부여하려는 활동으로 정의된다. 이러한 측정은 이론적으로 다음과 같은 특성을 내포한다.

• 관계(Relation) : 어떤 이해된 혹은 관찰한 속성에 대한 특성을 결정하고, 특성들을 개체들간의 관련성으로 정의한다. 예를 들면 개체 A는 개체 B보다 크다(뜨겁다, 무겁다 등).

• 표현(Relation) : 위에서 파악한 개체과 개체간의 관계를 적절한 수와 기호로 표현한다. 예를 들면 위의 'A는 B보다 크다'를 'A > B'로 표현한다.

• 유일성(Uniqueness) : 만약 위의 표현이 여러 개 존재한다면 그 표현들(함수관계)간에는 어떤 관련성이 존재한다. 즉 하나의 표현은 여러 형태가 있을 수 있지만 근본적인 의미는 고유하다.

측정의 형태는 그 측정하려는 속성들간의 관련성에 따라 두가지로 구분된다.

• 직접측정(Direct Measurement) : 어떤 한 속성을 다른 속성들과 독립적으로(연관없이) 측정하는 활동. 예를 들면, 오류수, 크기 등

• 간접측정(Indirect Measurement) : 어떤 한 속성을 하나이상의 다른 속성들에 연관지어서 측정하는 활동. 예를 들면 노력(E) = 5.2 * 프로그램크기(S). 여기서 E는 간접측정값, S는 직접측정값

측정은 대개 매트릭(Metric)을 이용한다. 매트릭은 측정하려는 속성에 값을 부여하는 일종의 계산공식(Formula)라 할 수 있다.

해 품질을 평가할 수 밖에 없다. 또한 품질 측정을 위해서는 각 속성에 대해 매트릭이 설정되어야 하며 매트릭들은 각 속성을 측정하기에 적합한 것이어야 한다.

소프트웨어 품질매트릭은 소프트웨어 혹은 소프트웨어를 구성하는 요소들 -모듈, 자료, 문서 등을 입력으로 하여 그 소프트웨어가 어떤 속성을 갖는지를 (수치)값으로 계산하는 역할을 수행한다.

매트릭 적용을 위해서는 매트릭 적용대상을 미리 설정해야 한다. 소프트웨어 품질측정을 위한 매트릭 적용대상은 크게 개발과정(Process), 산출물(Product), 자원(Resource) 등으로 구분할 수 있다. 다음 표는 이들에 대한 세부측정대상과 속성들을 예시한 것이다. 측정대상중 개발과정, 산출물은 기술적인 관점이고, 자원은 사업관리관점이다.

측정 대상		측정 속성
산출물	명세	크기, 재사용, 모듈화, 중복성, 기능, 구문의 정확성
	설계	크기, 재사용, 모듈화, 응집성, 기능
	코드	크기, 재사용, 모듈화, 응집성, 기능, 복잡성, 구조화
	시험자료	크기, 시험범위
개발과정	요구명세	시간, 노력, 요구변경수
	상세설계	시간, 노력, 명세오류수
	시험	시험, 노력, 오류수
자원 (사업관리측면)	참여요원	나이, 임금
	팀구성	크기, 의사소통수준, 구조
	장비	가격, 규모, 속도(성능), 기억용량
	사무실(환경)	면적, 온습도, 빛

2.2 소프트웨어 품질관련 기본개념

소프트웨어 품질은 소프트웨어가 원하는 속성들을 갖고 있는 정도라고 정의된다. 여기서 속성들은 명확히 정의되어야 한다. 그렇지 않으면 직관에 의

소프트웨어 품질측정에 매트릭을 활용하는 이유는 소프트웨어 개발주기동안에 소프트웨어 품질의 요구사항이 만족되고 있는지를 좀더 객관적으로 평가하기 위한 것이다. 또한 매트릭을 활용하면 정

량적인 기준을 제공하기 때문에 주관적인 요소를 가능한 배제할 수 있고, 품질을 정량적으로 가시화할 수 있어 개발자나 관리자가 유용한 정보를 얻을 수 있다. 개발자, 관리자가 측정활동시 매트릭 활용으로 얻을 수 있는 잇점과 매트릭이 필요한 시기를 구체적으로 살펴보면 다음과 같다.

- 매트릭 활용 잇점

• 품질 목표를 달성하는데 도움을 준다.

• 개발초기에 소프트웨어에 대한 품질요구사항을 설정할 수 있다.

• 소프트웨어 인수기준과 조건을 설정할 수 있다.

• 설정된 요구사항에 대해 실현된 품질수준을 평가할 수 있다.

• 소프트웨어에 내재되어 있을 기능이상이나 문제점들을 탐색할 수 있다.

• 최종 소프트웨어 품질수준을 예측할 수 있다.

• 소프트웨어 변경에 따른 품질의 변화를 감시할 수 있다.

• 산출물개선에 따른 변경용이성을 평가할 수 있다.

• 매트릭을 정규화, 등급화, 조정, 검증할 수 있다.

- 매트릭이 필요한 시기

• 관리자 입장

. 개발비용 산정시

. 산출물의 품질을 알고자 할때

. 사업목표를 설정하고자 할때

. 개발방법, 도구유용성을 평가하고자 할때

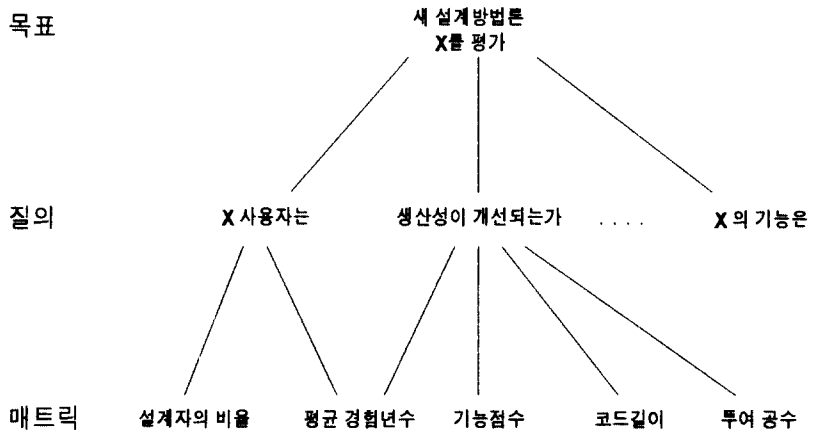
• 개발자 입장

. 개발이 진행되면서 품질을 알고자 할때

. 품질요구사항을 명세화하고자 할 때

. 인증목적으로 개발과정, 산출물을 평가하고자 할 때

. 개발될 소프트웨어가 갖는 품질속성을 예측하고자 할때



<그림-1> 목표/질의/매트릭 모형

3. 소프트웨어 품질메트릭 기본구조

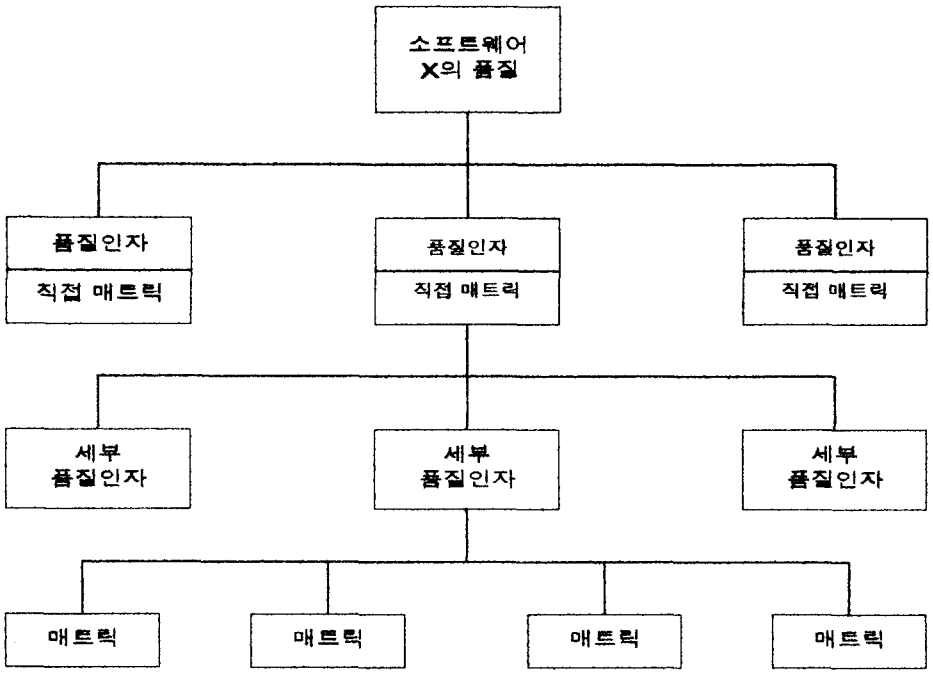
소프트웨어 품질메트릭을 효과적으로 적용하기 위해서는 메트릭적용을 위한 기본틀이 필요하다. 이에 관련된 연구의 하나로 바실리(Basili), 롬바크(Rombach)는 <그림-1>과 같은 소프트웨어 품질메트릭 적용을 위한 목표/질의/메트릭(GQM: Goal/Question/Metric)모형을 제안하였다.

이 모형 자체는 간단하지만 품질메트릭 적용을 위한 기본개념과 구조, 절차를 제시하고 있다. 이 모형을 해석하고, 체계적으로 정리하면 일반적으로 활용할 수 있는 소프트웨어 품질메트릭의 기본구조를 유도할 수 있다. 우선 메트릭을 적용하는 목적를 정하고, 기술적 관점에서 품질의 목표를 설정한 후 이를 달성하기 위해 세부적으로 도달해야 하는 세부 품질속성과 각 속성을 측정하는 메트릭을 정한다. <그림-2>는 이러한 소프트웨어 품질메트릭 기본구조를 나타낸 것이다.

그림에서 알 수 있듯이 소프트웨어 품질메트릭 기본구조는 크게 품질인자, 세부품질인자, 메트릭으로 구성된다.

- 품질인자(Quality Factor) : 관리자, 사용자입장에서 정한 품질요구로서 각 인자에는 인자를 정량적으로 표현하기 위한 직접메트릭이 선택된다. 예를 들면 신뢰성 품질인자 경우, 직접메트릭으로 고장간 평균시간(MTTF:Mean Time To Failure)을 설정할 수 있다.

- 세부품질인자(Sub-Quality Factor) : 하나의



<그림-2> 소프트웨어 품질메트릭 기본구조

품질인자를 측정가능한 품질속성들로 분할한 것으로 하나의 세부품질인자가 여러 품질인자에 관련될 수 있다.

- 매트릭(Metrics) : 매트릭은 개발주기동안 산출물과 개발과정을 측정하기 위해 세부품질인자를 분할한 것이다. 대개 최상위 계층인 품질인자수준에서 매트릭은 활용하기 어렵다. 왜냐하면 측정에 이용할 기초자료를 수집하기 어렵고 노력이 많이 들기 때문이다. 따라서 대개 최하위수준에서 식별된 매트릭을 이용해서 품질인자를 예측하기도 한다.

위 기본구조에서 품질인자와 세부품질인자로는 국제표준 ISO 9126의 내용을 수용하는 것이 바람직하다. <표-1>은 국제표준으로 제정된 소프트웨어 품질인자와 세부품질인자를 요약한 것이다. 이 표준에서 각 세부 품질인자에 대한 매트릭은 제시하지 않았다. 유럽의 소프트웨어 인증프로그램(SCOPE)등에서 이에 대해 연구하고 있으나 아직 정립되지 않았다. 매트릭은 품질측정을 하는 기관이나 부서의 환경에 맞게 정의하여 설정하도록 권고하고 있다..

<그림-2>에서 보인 소프트웨어 품질매트릭 기본구조는 품질측정에 관하여 관리자, 분석자, 설계자, 프로그래머, 시험요원, 유지보수요원 등 품질관련 요원간에 객관성있는 의사소통 수단으로 활용될 수 있다.

<표-1> ISO 9126의 품질인자 및 세부품질인자

품질인자(QF)	세부 품질인자(Sub-QF)	정 의	매트릭(M)
기능성	적절성(Suitability)	요구된 기능에의 적합여부	
	정밀성(Accuracy)	결과 및 효과의 정확성	
	상호운용성(Interoperabili.)	타 체계와의 상호연동성	
	순응성(Compliance)	표준,관례,규정 준수 여부	
	보안성(Security)	불법접근 방지 능력	
신뢰성	성숙도(Maturity)	결함에 의한 고장빈도	
	고장내성(Fault Tolerance)	오류시 일정성능 유지능력	
	회복성(Recoverability)	고장시 본래 성능으로 회복	
사용성	이해용이성(Understanabil.)	체계 개념이해, 적용 노력	
	학습용이성(Learnability)	체계 이용에 소요된 노력	
	운용성(Operability)	체계운용에 소요된 노력	
효율성	시간동작(Time Behavior)	기능수행의 시간(응답,처리)	
	자원동작(Resource Behav.)	기능수행의 소요자원량	
유지보수	분석용이성(Analysability)	결함,오류,수정 식별 노력	
	변경용이성(Changeability)	수정,결함제거,변경 노력	
	안정성(Stability)	수정에 의한 비예측 위험	
	시험용이성(Testability)	체계 검증 노력	
이식성	적응용이성(Adaptability)	환경전환시 소요노력	
	설치용이성(Installability)	특정 환경에 설치하는 노력	
	적합성(Conformance)	이식성관련 표준적응 노력	
	대체용이성(Replaceability)	타 체계로의 대체가능성	

4. 소프트웨어 품질매트릭 방법론

소프트웨어 품질측정에서 매트릭 적용이 매우 중요하다. 특정 품질인자를 어떤 매트릭으로 측정하는가는 측정활동 전반에 영향을 주고 그 결과에 대해서도 의미를 부여한다. 가령 한 품질인자를 그것과는 관련없는 매트릭으로 측정하였다면 그 측정결과는 의미가 없게 되고 측정을 위해 소요된 시간과 비용이 낭비된다. 따라서 효과적인 측정활동을 위해서는 유효한 매트릭의 설정, 측정결과의 타당성 입증이 필요한데 이를 위해서는 측정하려는 품질인자를 매트릭과 연계시키고, 적용가능한 매트릭 식별, 식별된 매트릭을 구현하기 위해 필요한 자료수집, 매트릭 구현, 구현결과의 분석, 매트릭의 검증 등과 같은 일련의 절차를 체계적으로 정리하여 적용해야 한다. 이러한 과정들은 하나의 수명주기의 틀로 체계화할 수 있고 품질매트릭 방법론으로 정립할 수 있다. 여기서 이에 대해 구체적으로 논의한다.

4.1 품질메트릭 수명주기

소프트웨어 품질매트릭 방법론은 소프트웨어 품질요구사항을 설정하고, 소프트웨어 개발과정과 산출물에 대해 적용할 품질매트릭을 식별, 구현, 분석, 검증하기 위한 체계적인 접근방법이다. 방법론은 수명주기로 나타낼 수 있으며, 수명주기의 각 단계는 다음과 같이 구분될 수 있다.

- 소프트웨어 품질요구사항 설정 : 소프트웨어 개발초기 혹은 변경시 원하는 품질인자들을 선택하고, 우선순위를 부여하고, 정량화한다. 품질요구사항들은 소프트웨어 개발시 지침이 되고 사업관리에 활용된다. 또한 인도시에는 개발된 소프트웨어가 명시된 품질요구사항을 만족하는지를 평가하는데 이용된다.

- 소프트웨어 품질매트릭 식별 : 앞에서 정의한 소프트웨어 품질매트릭 기본구조를 이용하여 관련 매트릭들을 선택하고 선택한 매트릭을 어느 단계에서 적용할 것인지를 결정한다.

- 자료수집 : 식별된 매트릭을 구현할 수 있는 도구를 도입하거나 개발하고, 매트릭 적용에 필요한 기초자료(표본자료)를 모은다.

- 소프트웨어 품질매트릭 계산 : 소프트웨어 개발주기 각 단계에서 매트릭을 계산한다.

- 소프트웨어 품질매트릭 결과를 분석 : 매트릭 적용결과들을 분석하고 문서화한다. 분석결과는 개발과정을 조정통제하고 최종 산출물을 평가하는데 이용된다.

- 소프트웨어 품질매트릭 검증 : 매트릭 결과와 품질인자의 직접매트릭값간의 상호관련성을 분석하여 매트릭이 정확하게 품질인자들을 예측하였는지를 검증한다.

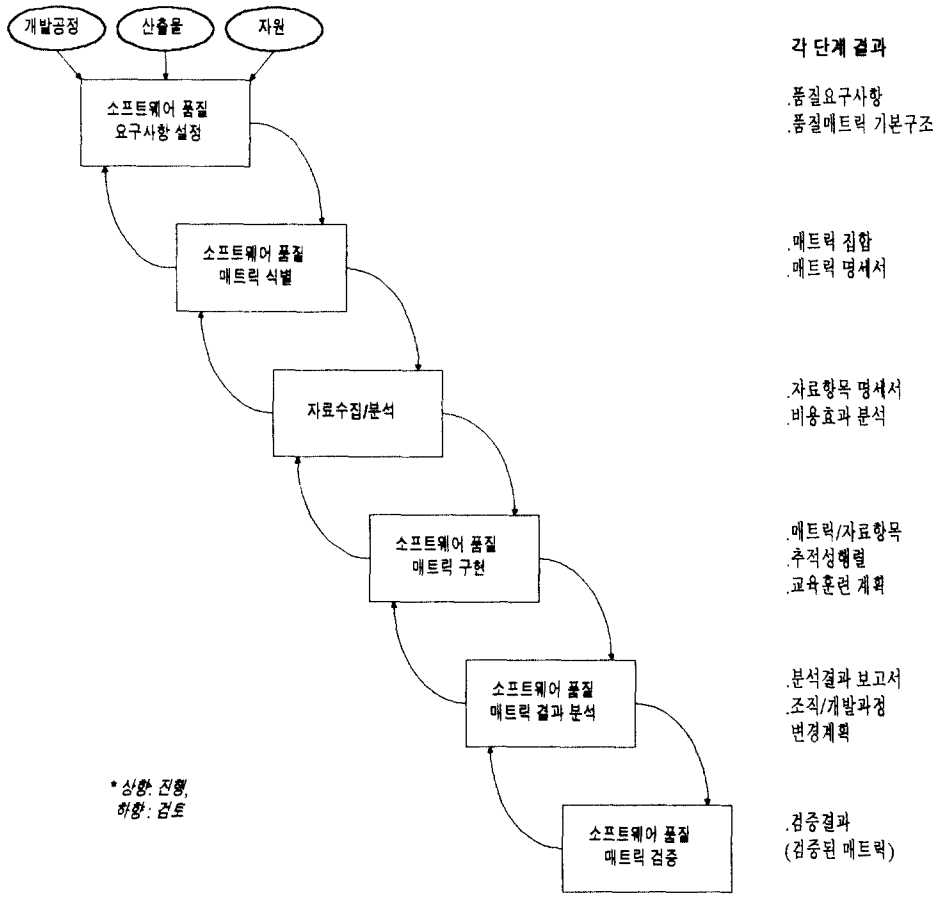
위의 각 단계들을 진행하면서 보다 정확한 매트릭 적용을 위해 필요시 이전단계의 결과를 재검토한다. <그림-3>은 품질매트릭 수명주기의 각 단계와 각 단계에서의 주요 산출물을 나타낸 것이다.

4.2 각 단계에서의 활동

소프트웨어 품질매트릭 수명주기의 각 단계를 좀더 구체적으로 살펴 본다.

(1) 소프트웨어 품질요구사항 설정

소프트웨어 품질요구는 최종 산출물이 만족해야



<그림-3> 소프트웨어 품질매트릭 방법론 단계 및 문서화

하는 목표값(예를 들면, 소프트웨어 신뢰성에 관한 MTTF값)이거나 품질인자에 설정된 직접매트릭 값(예를 들면, 코드 오류수)으로 설정된다. 이 단계에서 수행되어야 할 세부활동은 다음과 같다.

- 가능한 품질요구사항들을 모두 목록화
  . 경험, 표준, 규정 등을 이용하여 가능한 모든 품질요구사항들을 추출한다.
  . ISO 9126에서 정한 품질인자/세부품질인자들을 참조한다.
  . 품질에 영향을 줄수 있는 제한사항들-일정, 예산 등-을 고려한다.
  . 품질인자와 직접매트릭 추출에 중점을 둔다.
- 최종 품질요구사항 결정
  . 중요성에 따라 위에서 목록화한 요구사항들에 우선순위를 부여한다.
  . 위의 품질요구사항을 기술적으로 분석가능한지, 합당한지, 구현가능한지를 검토한다.
  . 최종 품질요구사항을 결정한다.
- 각 품질인자를 정량화
  . 위 단계에서 최종선택한 각 품질인자에 대해

직접매트릭을 설정하고, 품질목표값을 매트릭값으로 나타낸다. 이때 매트릭값은 그 품질요구를 정량적으로 표현한 것이다. 예로, '높은 효율성' 이라는 품질요구사항을 정한 경우, 이를 '할당된 자원에 대한 실제 자원의 이용율이 90% 이상' 이라고 정량화하여 나타낸다.

(2) 소프트웨어 품질매트릭 식별

이 단계에서는 앞에서 정의한 소프트웨어 품질매트릭 기본구조에 맞춰 품질인자, 세부품질인자, 매트릭을 정한다. 우선 최종선택한 품질인자에 대해 세부 품질인자들을 설정하고 각 세부품질인자에 대해 측정가능한 매트릭들을 설정한다. 각 매트릭에 대해서는 목표값, 임계값 등을 부여한다. 이외에 자세한 매트릭 명세는 <표-2>와 같은 매트릭 명세서로 정리하는 것이 바람직하다.

식별한 매트릭들에 대해 다음과 같은 요소들-구현에 소요되는 비용, 적용효과 등-을 추가로 고려하여 현실적으로 유용한 매트릭을 설정할 수 있도록 한다.

- 구현에 소요되는 비용
  - 매트릭 이용 비용 : 자료수집, 매트릭값 계산, 매트릭결과 분석 및 보고에 관련된 비용
  - 소프트웨어 개발과정의 변경에 따른 비용 : 매트릭이 개발과정에 변경을 초래할 경우에 개발과정 변경에 드는 비용
  - 조직구조 변경에 따른 비용 : 매트릭이 조직의 변경을 초래할 경우에 조직 변경에 드는 비용
  - 장비 비용 : 매트릭 구현에 필요한 장비 확보에 드는 비용
  - 교육훈련 비용 : 품질보증/조정팀, 개발팀이 매트릭 사용, 자료수집 기술을 습득하는데

<표-2> 매트릭 명세서 형태

항 목	명 세
이름	매트릭 이름
비용	매트릭을 적용하는데 소요되는 비용
효과	매트릭적용에 따른 효과
영향	매트릭이 프로젝트, 소프트웨어의 품질에 미치는 영향
목표값	품질목표를 만족하기 위해 달성해야 하는 매트릭 값
품질인자	이 매트릭에 관련된 품질인자
도구	자료의 수집/저장, 매트릭 계산, 결과분석에 이용되는 SW,HW
적용	매트릭의 이용방법과 적용분야
자료항목	매트릭값을 계산하는데 필요한 자료항목들(입력값)
계산방법	매트릭 값을 계산하는 방법
해석	매트릭 계산결과에 대한 해석
고려사항	매트릭 가정, 적절성(자료가 수집될 수 있는가, 이 분야에 매트릭이 적절한가 등에 대한 설명)
교육훈련 소요	매트릭을 구현하고 사용하는데 요구되는 교육훈련
예	매트릭을 적용하는 예
검증이력	매트릭이 사용된 프로젝트, 만족했던 검증기준
참고사항	매트릭 이해, 구현에 대한 상세한 자료들

소요되는 비용

- 특정 매트릭을 적용할 경우에 예상되는 효과
  - 소프트웨어 개발팀에게 품질목표를 인지시키는데 기여한다.
  - 소프트웨어 품질을 향상시키기 위한 재검토 시점을 알려준다.
  - 소프트웨어 품질을 정량화함으로써 고객의 만족을 제고시킨다.
  - 소프트웨어 품질을 판단하기 위한 정량적인 기준을 제공한다.
  - 매트릭값에 근거하여 개발과정을 개선함으 할 수 있다.

위의 사항들에 대한 분석결과를 토대로 품질측정에 적용할 최종 매트릭들을 추출한다.

(3) 자료수집과 분석

각 최종매트릭에 대해서 매트릭 구현에 필요한 자료들을 정의한다. 이를 위해 자료수집절차를 정하고, 자료수집도구 활용이 가능할 경우 도구활용 시기와 방법을 명확하게 정한다. 매트릭의 정확성과 실효성은 자료에 좌우되기 때문에 매트릭 구현을 위해 어떤 자료항목들이 필요한지를 완전하면서 정확하게 정의해야 한다. 각 자료항목에 대해서도 <표-3>와 같은 자료항목 명세서로 정리하는 것이 바람직하다.

자료항목을 정의한 후에는 특정 소프트웨어를 선택하여 정의한 자료항목에 대해 자료들을 수집해보고 자료들이 올바르게 수집되는지, 매트릭 계산이 가능한지를 시험하고 필요시 매트릭 명세서와 자료항목 명세서를 개선한다. 위와 같은 시험, 개선과정을 거쳐 확정된 수집대상자료에 대해 실자료를 수집한다.

<표-3> 자료항목 명세서 형태

항 목	명 세
이름	자료항목 이름
관련매트릭	자료항목에 관된 매트릭들
정의	자료항목에 대한 명확한 정의
소스	자료가 발생하는 위치
수집자	자료 수집에 책임있는 사람, 부서 등
시기	자료가 수집되어야 하는 개발주기상의 시기
절차	자료를 수집하는데 이용되는 방법
저장	자료가 저장되는 위치
표현	자료가 표현되는 방식 (정확도, 형식 등)
표본추출	수집되어야 할 자료를 선택하는 방법 등
확인	수집된 자료의 검증 방식
대안	보다 나은 자료수집방법을 제시
무결성	자료항목을 변경해야 하는 조건, 변경책임자

(4) 소프트웨어 품질매트릭 구현(계산)

매트릭 명세서와 자료항목 명세서에 따라 소프트웨어 개발주기상의 특정 시점에서 수집된 자료들을 이용하여 매트릭을 계산한다.

(5) 소프트웨어 매트릭 결과 분석

매트릭 구현(계산) 결과를 해석한다. 특히 수집

된 자료를 이용하여 계산한 실제 매트릭값과 미리 설정한 목표값간의 차이를 분석한다. 실제 값이 임계구간을 벗어나면 그 소프트웨어는 품질목표 혹은 품질요구사항을 만족하지 못한 것으로 소프트웨어가 바람직하지 못한 품질속성 예를 들면, 과도한 복잡성, 부족한 문서화수준, 추적성 부족 등과 같은 성질을 갖고 있음을 나타낸다. 이러한 분석결과는 측정대상인 소프트웨어의 취약한 요소들을 알려줌과 동시에 품질향상을 위해 취할 수 있는 방안-예를 들면, 재설계, 모듈화 강화 등-을 알려주거나 유추할 수 있게 한다.

매트릭 결과는 품질인자값 예측에도 이용된다. 예를 들어 신뢰성을 측정할 경우 개발과정 혹은 산출물에 대해 직접적으로 신뢰성을 측정하기는 쉽지 않다. 왜냐하면 신뢰성계산에 필요한 자료를 수집하기 어렵기 때문이다. 이 경우 신뢰성에 대한 직접매트릭(예:오류의 발생횟수)과 관련성이 있는 다른 매트릭(예:Cyclomatic No.)을 통해 간접적으로 신뢰성을 예측해 볼 수 있다. 여기서 오류수나 Cyclomatic No.는 개발과정, 산출물로부터 바로 얻을 수 있는 값이다.

이와 같이 매트릭을 이용하면 개발 초기단계에서 개발 후반의 품질을 예측할 수 있을 뿐 아니라, 품질목표값과 비교(즉 결과값이 임계범위내에 있으면 소프트웨어가 원하는 품질을 갖는 것이고 임계범위를 벗어나면 품질이 만족스럽지 못한 것이다.)를 통해 소프트웨어가 품질요구사항을 준수하는지를 확인할 수 있다.

(6) 소프트웨어 품질매트릭 검증

(가) 정의 및 목적

매트릭 검증은 실제 품질인자값(직접매트릭값)과 예측에 이용된 매트릭값사이의 통계적인 중요성을 결정하는 작업이다. 만약 통계적 중요성이 있다면, 그 매트릭은 해당 품질인자 측정에 사용할 수 있는 검증된 매트릭(Valided Metric)이다.

매트릭을 검증하는 목적은 특정 품질인자값을 예측할 수 있는 매트릭을 식별하려는 것이다. 품질인자값을 알수 없는 경우에 예측이 필요하게 되는데, 예측을 위해서는직접매트릭과 예측에 이용된 매트릭과의 연관성을 분석하여 연관성이 있을 경우 그 매트릭은 검증된 매트릭으로서 해당 품질인자를 평가할 수 있게 된다.

검증된 매트릭이 필요한 시기는 개발초기에 개발후반의 개발과정과 산출물에 대해 품질을 평가 혹은 예측하고자 할 경우 혹은 특정 개발주기에서 품질인자값을 평가하고자 할 경우 등이다.

(나) 검증요건

앞서 논의한 바와 같이 어떤 매트릭이 검증된 것이라면, 그 매트릭은 특정한 품질인자와 높은 연관성을 갖고 있음을 의미한다. 매트릭의 검증은 쉬운 것이 아니나 대개 다음과 같은 요건의 만족여부로 판단할 수 있다.

- 상호연관성(Correlation)
- 추적성(Tracking)
- 일관성(Consistency)
- 예측성(Predictability)
- 식별성(Discriminative Power)
- 신뢰성(Reliability)

여기서 검증요건에 대해 좀더 구체적으로 살펴본다. 이를 위해 우선 검증기준을 아래와 같이 설

정하였다.

V : 선형 상관계수(R)의 제곱 B : 순위 상관계수 E : 예측 오류률 S : 성공률

• 상호연관성 : 이 요건은 품질인자(직접매트릭)와 특정 매트릭간의 관련성을 평가하는 것이다. 만약 관련성이 존재한다면 그 매트릭값으로 품질인자를 평가 혹은 예측할 수 있다. 상호연관성을 보이기 위해 먼저 품질인자값의 편차와 매트릭 값의 편차를 계산하여 선형상관계수 R의 제곱을 구한다. 만약 그 값이 검증기준 V보다 클 경우(R^2 > V)에 상호연관성이 존재한다고 해석한다. 예로 복잡성(Complexity) 매트릭과 신뢰성(Reliability) 품질인자의 선형상관계수 R이 0.8인 경우 이 값의 제곱은 0.64이다. 이때 먼저 설정한 검증기준(V값)이 0.7이었다면 그 매트릭은 검증기준을 만족하지 않는다. 이는 복잡성 매트릭과 신뢰성이 서로 연관성이 높지 않으므로 복잡성 매트릭은 신뢰성에 대해 검증된 매트릭이 아니다. 이는 신뢰성을 복잡성 매트릭으로 평가 혹은 예측할 수 없다는 것을 의미한다.

• 추적성 : 이 요건은 매트릭이 품질인자의 변화를 추적할 수 있는지를 평가하는 것이다. 가령 매트릭 M이 품질인자 F에 대해 추적성이 있다면, 품질인자값이 시점 t1, t2에서 F1, F2로 각각 변경될때, M값도 M1에서 M2로 변경되어야 한다. 이를 시험하기 위해 품질인자값과 매트릭값을 하나의 쌍(Pair)으로 하여 순위상관계수를 구하고 이 계수의 절대값이 위 표에서 설정한 검증기준 B를 넘는지 확인한다. 예를 들면, 복잡성 매트릭이 신뢰성 품질인자를 추적할 수 있다고 하면 신뢰성값의 변화가 복잡성 매트릭값의 변화로 연계되어야 하고 이것이 수치적으로 표현되어야 한다. 신뢰성의 직접매트릭의 하나인 MTTF값이 t1에 1000에서, t2에 1500으로 변할때, 복잡성 매트릭값이 같은 시각 t1, t2에서 8에서 6으로 개선되었다면 신뢰성이 개선됨에 따라 복잡성도 개선되었으므로 복잡성 매트릭은 신뢰성 품질인자에 대해 추적성이 있다고 할 수 있다.

• 일관성 : 이 요건은 동일 소프트웨어를 대상으로 품질인자값의 순위와 매트릭값의 순위가 일관성이 있는지를 평가하는 것이다. 가령 산출물 혹은 개발과정 1,2,..,n에 대해 품질인자값 F1, F2,.. Fn이 F1>F2>...>Fn의 관계를 갖을때, 이에 대응하는 매트릭값들도 M1>M2>..>Mn의 관계를 갖는지를 평가한다. 이를 시험하기 위해 품질인자값과 매트릭값간의 순위상관계수를 계산한다. 계산결과 계수의 절대값이 위에서 정한 검증기준 B를 넘으면 일관성이 있는 것이다. 예로 소프트웨어 X, Y, Z의 신뢰성(MTTF)값이 각각 1000, 1500, 900이고, 이에 해당하는 복잡성 매트릭값이 5, 3, 7이라면, 신뢰성 순위 Y, X, Z와 복잡성 매트릭 순위 Y, X, Z가 같으므로 복잡성 매트릭이 일관성을 만족한다. 따라서 복잡성 매트릭이 신뢰성을 일관성있게 나타낸다고 판단할 수 있다. 여기서 순위는 좀더 주의하거나 관심을 두어야 하는 소프트웨어를 식별하는 데도 이용될 수 있다.

• 예측성: 이 요건은 특정 매트릭이 얼마나 정확히 품질인자값을 예측할 수 있는지를 평가한다. 만약 t1에서 산출물이나 개발과정에 대해 t2의 품질인자를 예측한 경우, 예측오류률은 t2에서의 실제

품질인자값(At2)에서 예측값(Pt2)을 빼고 이를 다시 실제 값으로 나누어 구할 수 있다. 즉 (At2-Pt2)/At2 이다. 예측성은 예측오류률을 위에서 정한 검증기준 E값과 비교하여 판단할 수 있다. 예를 들어 소프트웨어 신뢰성을 MTTF를 이용하여 예측할 경우 MTTF가 t2에서 1200시간으로 예측되고, 실제값은 1000시간이라면 예측오류률은 20%이다. 검증기준을 예측오류률 25%로 설정했다면 이때의 예측값은 수용가능하다.

• 식별성 : 이 요건은 특정 매트릭이 낮은 품질의 소프트웨어와 높은 품질의 소프트웨어를 구분할 수 있는지를 평가하는 것이다. 이를 계산하기 위해 통계적인 방법(Mann-Whitney Test, Chi-Square Test등)을 이용할 수 있다. 예를 들어, 복잡성 매트릭의 값이 10보다 큰 소프트웨어 모듈들의 신뢰성 직접매트릭 MTTF가 1000, 10보다 같거나 작은 소프트웨어 모듈들의 MTTF가 2000이고 이들이 통계적으로 서로 구분된다면 복잡성 매트릭은 신뢰성에 대해 식별성을 갖는다고 할 수 있다.

• 신뢰성 : 이 요건은 특정 매트릭의 검증을 충분한 응용과 소프트웨어를 대상으로 시험하였는지를 평가하는 것이다. 완전히 검증된 매트릭은 상호연관성, 추적성, 일관성, 예측성, 식별성을 갖는데 이때 검증매트릭 적용시 예측성공률은 S이다. 즉 검증된 매트릭이라 하더라도 품질인자값을 예측하는데는 1-S 만큼의 오류를 포함하고 있다. 예를 들어 복잡성 매트릭의 예측성에 대해 검증할 경우, 100개의 소프트웨어 모듈이 있고 원하는 성공률 S가 80%이라 하면, 복잡성 매트릭은 적어도 80개의 소프트웨어 모듈에 대해서는 정확히 품질인자값을 예측한다고 할 수 있다.

(다) 검증절차

매트릭을 검증하는 절차는 다음과 같다. 매트릭 검증에 필요한 자료가 매트릭 데이타베이스에 저장되어 있다고 가정한다.

① 품질인자값의 표본 식별

검증을 위해 매트릭 데이타베이스로부터 품질인자값의 표본들을 추출한다.

② 매트릭 값의 표본 식별

검증할 매트릭에 대해 매트릭값을 매트릭 데이타베이스로부터 추출한다.

③ 통계적 분석

검증할 매트릭에 대해 위의 검증요건들이 만족되는지를 통계적 방법을 이용하여 시험한다.

④ 결과의 문서화

검증하기 위한 사전정보와 검증과정에서 산출되는 산출물을 문서화한다. 매트릭값, 검증기준, 검증과정에서 산출된 시험결과 등을 문서화한다.

5. 방법론의 적용

본 장에서는 4장에서 살펴 본 소프트웨어 품질매트릭 방법론을 적용하는 방법을 논의한다.

소프트웨어 품질매트릭 방법론을 적용하기 위해서는 기본적으로 품질측정대상 소프트웨어, 매트릭 계산을 위한 자료 등이 선별, 수집되어야 하나 현실적으로 이들을 준비하는 것은 어렵다. 왜냐하면 소프트웨어의 품질을 측정하는 경우가 거의 없었을 뿐아니라 필요한 자료들은 대개 체계적인 품질관리활동에 의해 수집되기때문에 이러한 기반이

<표-4> 감시/판별 서브시스템에 대한 소프트웨어 품질인자

소프트웨어 품질인자 / 소프트웨어 기능	효율성	무결성	신뢰성	생존성	편의성	정확성	유지용이성	확인능력	확장성	상호운용성	유연성	재사용성
감시/판별 서브시스템												
- 자료수집/추출	×		×	×						×	×	
- 동적인 그래픽 출력	×		×		×							
- 목표물 인식	×		×									
- 위협 탐지/식별	×		×									
- 위협 전시	×		×		×	×	×				×	
- 위협 응전 지원	×	×	×		×	×	×	×			×	×

없는 현상황에서 실자료를 얻기는 쉽지 않다. 이에 본 연구에서는 가상의 소프트웨어, 자료를 대상으로 4장에서 제시된 방법론을 적용시켜 보았다.

5.1 가상의 측정대상 소프트웨어

임무긴요 소프트웨어(Mission Critical Software)로 분류되는 무기체계 삽입 소프트웨어(Embeded software)는 정확하고, 신뢰할 수 있는 것이어야 하기 때문에 그 품질이 매우 중요하다.

본 고에서는 삽입소프트웨인 군 지휘통제 시스템중에서 레이다 소프트웨어를 가상의 소프트웨어로 하여 품질매트릭 방법론 적용성을 논의하고자 한다.

이 소프트웨어는 5가지의 기능을 수행하는 서브시스템으로 구분되어 있다. 괄호안은 각 서브시스템의 명령문수, 모듈수이다.

- 감시/판별 서브시스템 (1600, 24)
- 위협 평가 서브시스템 (2400, 48)
- 무기 탑재 및 조정 서브시스템 (3000, 53)
- 전투요원의 전투 서브시스템 (4400, 67)
- 통신 서브시스템 (2000, 50)

여기서는 위 서브시스템중 감시/판별 서브시스템에 대해서 논의한다.

5.2 방법론의 수명주기별 세부활동

(1) 소프트웨어 품질 요구사항 설정

우선 감시/판별 서브시스템에 대해 고려할 수 있는 품질요구사항들을 식별한다. 품질인자를 추출하기 위해 감시/판별 서브시스템을 좀더 세부기능으로 구분하고 각 세부기능을 품질인자와의 연관시킨다. <표-4>는 이의 결과를 나타낸 것이다.

위의 결과를 다시 여러 관점 즉 획득(구매) 관점, 운용관점 등에서 검토하고 관련자들의 의견을 반영한다. 또한 품질인자들간의 상호관련성, 중복성 등을 분석하여 좀더 중요하고 필수적인 품질인자들을 식별한다.

필수적인 품질인자를 식별한 결과 감시/판별 서브시스템의 품질이 효율성, 신뢰성, 편의성, 정확성, 유지용이성에 따라 좌우된다고 검토되었다면, 이들이 감시/판별 서브시스템과 관련된 최종 품질

인자이다. <표-5>는 최종선별된 품질인자들을 나타낸 것이다.

소프트웨어 품질요구사항은 최종선발된 품질인자에 부여하는데, 이를 위해 각 품질인자에 대해 직접매트릭과 목표값을 설정한다. <표-6>은 최종 품질인자중 신뢰성에 대해 직접매트릭과 목표값을 설정한 것이다.

<표-7>은 매트릭 설정과 매트릭의 적용단계를 보인 것으로 세부품질인자중 모듈성(Moduality), 간편성(Simplicity)에 대한 매트릭들을 M1, M2, M3, S1, S2, S3, S4, S5으로 설정하고 이들이 어느단계에서 활용되는지를 예시한 것이다. 각 매트릭에 대해서는 <표-8>과 같이 매트릭명세서로 정의하고 구현하기전에 매트릭 구현에 따르는 비용

<표-5> 감시/판별 서브시스템에 대한 최종 소프트웨어 품질인자

소프트웨어 품질인자 / 소프트웨어 기능	효율성	무결성	신뢰성	생존성	편의성	정확성	유지용이성	확인능력	확장성	상호운용성	유연성	재사용성
감시/판별 서브시스템	A	B	A	B	A	A	A	B	B	B	B	B

(2) 소프트웨어 품질매트릭 식별

여기서는 소프트웨어 품질매트릭 기본구조에 맞춰 앞에서 선택한 품질인자에 대해 세부품질인자를 정하고, 각 세부품질인자에 대해 매트릭들을 설정한다.

즉 매트릭 이용비용, 소프트웨어 개발과정 변경비용, 조직구조 변경비용, 장비확보 비용, 교육훈련 비용 등을 추정하고 얻게 되는 이익들을 분석한다. 분석결과에 따라 매트릭을 추가, 삭제 또는 변경하여 최종적으로 적용할 매트릭들을 설정한다.

<표-6> 신뢰성 품질인자 목표값(예)

품질인자	매트릭	목표값
신뢰성	오류수/명령문	< 0.5
신뢰성	오류수/모듈	< 1.0

매트릭은 각 세부품질인자에 대해 측정할 수 있는 사항들을 질의문으로 기술한후 각 질의문을 매트릭으로 전환하여 설정할 수 있다. 매트릭을 설정한 후에는 매트릭을 개발단계의 어느 단계에서 적용할 것인지를 결정한다.

<표-7> 감시/판별 서브시스템에 대한 품질인자/세부품질인자/매트릭

품질인자	세부품질인자	개발단계별 매트릭			
		분석	예비설계	상세설계	코딩
효율성	시스템 사용용이성				
신뢰성	정확성				
	고장관리				
	간편성	S1	S2	S3, ***S4, S5***	
편의성	운용용이성				
	교육훈련 요구정도				
정확성	완전성				
	일관성				
	추적성				
유지용이성	일관성				
	가시성				
	모듈성	M1	M1, M2	M2, M3	M2, M3
	간편성	S1	S2	S3, S4, S5	S3, S4, S5

<표-8> 감시/판별 서브시스템에 대한 매트릭 명세서(예)

항 목	명 세
매트릭 이름	모듈화 구현(모듈화하여 구현하였는가) **M1**
효과	구조적 설계기법이 사용되었는지를 점검할 수 있음
영향	이 매트릭값이 만족되지 않은 모듈은 검토되고 수정되어야함
목표값	1
관련품질인자	유지용이성
계산방법	주관적 판단에 따름. 결과: 예, 아니오
결과해석	모듈성을 예측하고, 문제요소를 지적하는데 이용

항 목	명 세
매트릭 이름	모듈화 설계(모듈화하여 설계하였는가) **M2**
효과	모듈의 응집력을 향상시키는 효과발생
영향	평균 응집력이 0.6보다 작으면, 모듈성을 향상시켜야 함
목표값	평균 응집력이 0.6보다 커야 함
관련품질인자	유지용이성
적용	소프트웨어 모듈화 정도를 결정하는데 이용
자료항목	소프트웨어 모듈의 응집력 값
계산방법	측정대상 소프트웨어의 응집력을 결정하고 평균을 구함
결과해석	평균응집력이 0.6보다 작은 소프트웨어는 조사해야 함
예	모듈:응집력 (예) A:0.5, B:0.6, C:0.5, D:0.7, E:0.7 -> 평균응집력:0.6

항 목	명 세
매트릭 이름	설계구조(Call/Return) **M3**
효과	개발이 정상적으로 되었는지 확인하는 효과
영향	값이 0이면, 해당 모듈 검토해야 함
목표값	1
관련품질인자	신뢰성, 유지용이성
자료/항목	소프트웨어 모듈간의 제어구조를 표현한 문서
계산방법	Call/Return이 제대로 되어 있는지 확인. 결과 예, 아니오
결과해석	결과가 0이면, 설계가 잘못된 것임

항 목	명 세
매트릭 이름	설계구조 (구조적으로 설계된 문서가 있는가) **S1**
효과	구조적으로 설계된 문서가 있다면, 문서는 개발에 유용함
목표값	1
관련품질인자	신뢰성, 유지용이성
자료항목	소프트웨어 기능들의 문서(도표)
계산방법	구조적으로 문서화되었는지를 검토하여 예, 아니오로 판단
결과해석	주관적이므로, 결과는 평가요원의 전문성에 의존

항 목	명 세
매트릭 이름	설계구조 (프로그래밍 스타일을 표준화하였는가) **S2**
영향	개발초기에 적용되면 영향 큼
목표값	1
관련품질인자	신뢰성, 유지용이성
계산방법	표준화되었는지를 확인. 예, 아니오로 판단
결과해석	표준화는 바람직함. 수학적으로 확인하기는 어려움

항 목	명 세
매트릭 이름	코드의 단순성(제어흐름이 단순한가) **S3**
영향	값이 0이면, 문제있는 소프트웨어 모듈이 정정되어야 함
목표값	1
관련품질인자	신뢰성, 유지용이성
자료항목	소프트웨어 모듈의 제어구조를 표현한 문서
계산방법	제어구조가 계층적인지에 따라 예, 아니오로 판단
결과해석	값이 0(아니오)면, 설계품질이 낮은 것임

항 목	명 세
매트릭 이름	Cyclomatic Number S4
영향	매트릭값이 너무 높으면, 소프트웨어를 좀더 나누어야 함
목표값	3이하
관련품질인자	신뢰성, 유지용이성
적용	소프트웨어의 복잡성을 예측하는데 이용
자료항목	소프트웨어 모듈의 방향그래프에서 가지, 노드수
계산방법	E - N + 2 (모듈을 방향그래프로로 표현했을때, E:edges수, N:노드수)
결과해석	매트릭값이 너무 높으면 모듈이 너무 크거나 복잡한 것임
예	E=12, N=10 이면, 매트릭값은 12-10+2 = 4

항 목	명 세
매트릭 이름	Size(한 모듈의 명령문 수) S5
영향	매트릭값이 너무 높으면, 소프트웨어를 좀더 나누어야 함
목표값	13이하
관련품질인자	신뢰성, 유지용이성
적용	소프트웨어의 크기와 복잡성을 예측하는데 이용
계산방법	지문을 제외한 한 모듈내의 명령문 수
결과해석	매트릭값이 너무 높으면 모듈이 너무 크거나 복잡한 것임
예	실제 명령문수는 자동으로 산출가능함

(3) 자료수집

위에서 식별한 매트릭들을 구현하기 위해 필요로 하는 자료들을 수집한다. 수집할 자료들은 주로 <표-8>에서 기술한 자료항목들이고 각 자료항목은 <표-3>에 보인 자료명세서로 정리한다. 예로 매트릭 M2를 계산하기 위해 필요한 자료항목을 <표-9>와 같이 자료명세서로 정리한다.

<표-9> 응집력 M2 계산을 위한 자료명세서(예)

항 목	명 세
자료항목 이름	모듈의 응집력 값
관련 매트릭	M2 모듈화 설계
정의	응집력은 모듈내의 기능적 관련성의 강도에 대한 측정치
참고할 문서	상세 설계문서
수집자	품질보증요원
수집시기	예비설계단계 종료직후
절차(방법)	각 모듈을 검토하여 결정

수집할 자료항목을 정의한후에는 매트릭 구현비

용, 매트릭 구현가능성, 자료항목 수집절차 등을 고려하여 자료를 수집한다. 수집된 자료는 매트릭 데이터베이스에 저장한다.

(4) 소프트웨어 품질매트릭 구현(계산)

매트릭 구현지원도구를 확보하고 앞에서 수집된 자료들을 이용하여 매트릭을 계산한다. 이때 매트릭 계산은 수집된 자료에서 통계적으로 추출한 표본자료를 이용한다.

(5) 소프트웨어 매트릭 결과분석

각 매트릭에 대해 매트릭 계산결과를 분석한다. 소프트웨어는 매트릭값으로 평가된다. 예로 예비설계단계에서 매트릭 S4, S5에 대한 계산결과 S4 > 3, S5 > 13인 모듈들은 S4가 3미만, S5가 13미만인 모듈에 비하여 오류가 발생할 가능성이 크고 품질이 낮다고 평가되며, 필요하다면 복잡성과 크기를 줄이기 위해 재설계할 것을 권고할 수 있다.

끝으로 품질요구사항이 만족되었는지를 판단한다. 이를 위해 각 품질인자의 목표값을 만족하는지를 분석한다. 예로 <표-6>의 목표값을 이용하여 신뢰성이 만족되는지를 확인한다. 가상 소프트웨어의 감시/판별 서브시스템에서의 오류수가 64이라면 오류수/명령문은 0.04(목표값은 <0.5), 오류수/모듈는 2.7(목표값은 <1.0)이다. 이런 경우에 감시/판별 서브시스템은 원하는 수준의 품질을 갖고 있지 않다고 판단한다.

(6) 소프트웨어 품질매트릭 검증

매트릭 검증은 앞에서 설정한 매트릭들이 품질인자의 평가 혹은 예측에 타당하게 사용될 수 있는지를 확인하는 단계로서 특정 매트릭이 직접매트릭과 검증요건을 만족하는지를 보이는 것이다. 이를 위해 검증기준하에 검증요건들이 만족되는지를 검증절차에 따라 확인한다. 검증시험은 통계적인 분석방법을 이용하는 것이 효과적이다.

본 고에서는 품질인자 신뢰성의 직접매트릭을 오류수로 하고, 앞에서 설정한 매트릭중 복잡성 매트릭 S4(Cyclomatic Number)와 크기 매트릭 S5(명령문수)가 오류수와 검증요건을 만족하는지를 검증해 본다. 검증기준을 아래와 같이 설정하였다.

- V : 0.7 (선형상관계수(L)의 제곱)
- B : 0.7 (순위상관계수(R))
- E : 25% (예측 오류률)
- S : 80% (성공률)

검증결과 S4, S5가 오류수와 검증요건을 만족한다면 S4, S5를 신뢰성(오류수) 측정에 활용할 수 있다.

(가) 품질인자 관련자료(표본) 식별

우선 매트릭 데이타베이스로부터 품질인자 신뢰성(오류수)에 연관있는 자료들을 추출한다. 오류의 유무가 표본추출의 기준이 될 수 있으므로 오류없는 모듈과 오류있는 모듈로 구분하여 표본을 추출한다.

(나) 매트릭 표본 식별

위에서 추출한 모듈들에 대해 S4(Cyclomatic수)와 S5(명령문수)를 산출한다. <표-10>은 수집된 표본의 예이다.

(다) 통계적 분석

<표-10.a> 자료수집결과 - 오류없는 모듈들의 매트릭값

모듈	Cyclomatic No.(S4)	명령문 수(S5)	오류수	모듈	Cyclomatic No.(S4)	명령문 수(S5)	오류수
1	3	6	0	7	2	9	0
2	4	8	0	8	1	4	0
3	3	11	0	9	5	12	0
4	1	7	0	10	3	15	0
5	2	5	0	11	2	6	0
6	6	16	0	12	4	9	0

<표-10.b> 자료수집결과 - 오류있는 모듈들의 매트릭값

모듈	Cyclomatic No.(S4)	명령문 수(S5)	오류수	모듈	Cyclomatic No.(S4)	명령문 수(S5)	오류수
13	2	12	3	19	7	45	5
14	4	24	5	20	3	30	4
15	5	22	6	21	9	68	10
16	3	15	2	22	5	53	8
17	1	17	2	23	13	79	14
18	6	36	9	24	2	19	4

<표-10>의 자료와 앞에서 설정한 검정기준을 이용하여 검증요건을 시험한다. 시험은 Cyclomatic 수와 명령문수가 오류수에 대해 상호연관성(Correlation), 추적성(Tracking), 일관성(Consistency), 예측성(Predictability), 식별능력(Discriminative Power)를 만족하는지 통계적으로 시험하는 것이다. 본 고에서는 시험방법을 예시하는 수준에서 논의하였고 신뢰도, 유의수준, 표본추출방법 등에 대해서는 고려하지 않았다.

• 상호연관성 : 상호연관성을 보이기 위해서는 오류수와 Cyclomatic수의 선형상관계수, 오류수와 명령문수의 선형상관계수를 계산하여 $L^2 > 0.7$이 만족되는지를 확인한다. <표-10>을 이용한 시험결과는 다음과 같다.

시험결과 오류가 있는 모듈만 대상으로 할 경우에 $L^2 > 0.7$ 이므로 Cyclomatic수와 명령문수가 오류수와 상호연관이 있으나, 모든 모듈을 대상으로 할 때는 명령문수와 오류수만이 선형관계가 있음을 알 수 있다.

• 추적성 : 추적성은 Cyclomatic수, 명령문수의

구 분	오류가 있는 모듈		모든 모듈	
	Cyclomatic No.	명령문 수	Cyclomatic No.	명령문 수
오류수와의 선형 상관관계	L = 0.908	L = 0.897	L = 0.787	L = 0.932

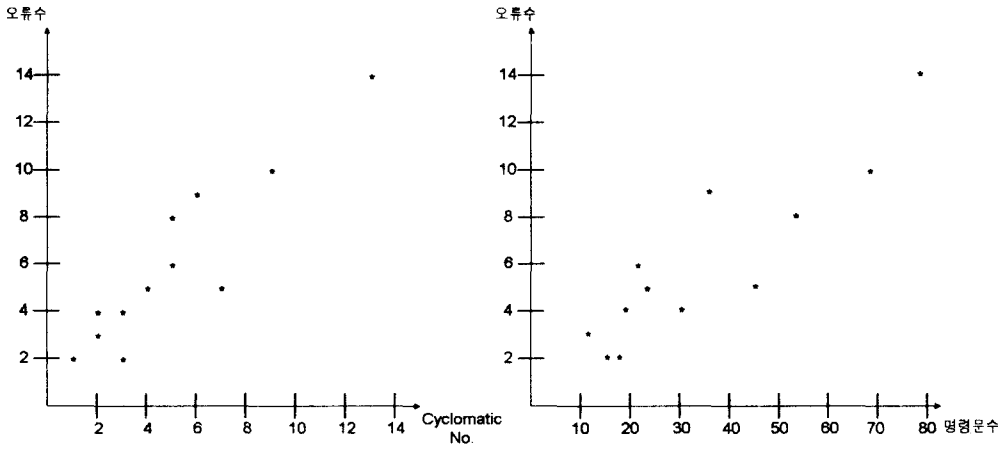
변화가 오류수에 영향을 미치는지를 보이는 것이다. 이를 보이기 위해 Cyclomatic수와 오류수와의 순위상관계수(Spearmen's rank correlation)를 계산하여 그 결과가 앞에서 설정한 검증기준 즉 B=0.7보다 큰지를 시험한다. 시험결과는 다음과 같다.

구 분	오류가 있는 모듈	
	Cyclomatic No.	비 고
오류수와의 순위상관관계	r = 0.897	r > 0.7

시험결과 r의 절대값이 0.7보다 크므로 추적성이 만족된다. 이는 Cyclomatic수가 오류수의 변화를 충분히 추적함을 의미한다.

• 일관성 : 일관성은 Cyclomatic수, 명령문수의 변화가 오류수를 일관성있게 변화시키는지를 보이는 것이다. 추적성 시험방법과 같이 Cyclomatic수와 오류수와의 순위상관계수를 계산하여 그 결과가 앞에서 설정한 검증기준 즉 B>0.7가 만족되는지를 시험한다. 시험결과에서 r의 값이 0.7보다 크므로 일관성이 만족된다. 이는 Cyclomatic수가 오류수의 변화를 일관성있게 나타냄을 의미한다.

• 예측성 : 예측성은 Cyclomatic수, 명령문수를 오류수와 각각 2차좌표상에 상관도(Scatter Diagram)을 그리고, 이들의 분포를 대표하는 회귀모형(Regression Model)을 구한후, 이 모형이 예측한 값과 실제값의 차이를 비교한다. 이로부터 예측오류률과 성공률을 구하여 검증기준인 예측오류률 E < 25%가 만족되는지를 검토한다. 먼저 Cyclomatic수, 명령문수를 오류수에 대해 각각 상관도를 그리면 <그림-4>과 같다.



<그림-4> 오류수-Cyclomatic, 오류수-명령문수 상관도

상관도에서 오류수(E)와 Cyclomatic수(C)와의 회귀모형을 최소자승법(Least Square Method)에 의해 구하면 E=0.9766C+1.1172이다. 이 모형으로 오류수를 예측하고, 예측오류수를 실제 오류수와

비교하면 다음과 같다.

구성요소중 감시/판별 서브시스템의 신뢰성를 설

Cyclomatic No.	모형 예측오류수	실제 오류수	예측오류률	예측성공(<25%)
2	3.0704	3	0.02346	o
4	5.0236	5	0.00472	o
5	6.0002	6	0.00003	o
3	4.0470	2	1.02350	
1	2.0938	2	0.04690	o
6	6.9768	9	0.22480	o
7	7.9534	5	0.59068	
3	4.0470	4	0.01175	o
9	9.9066	10	0.00934	o
5	6.0002	8	0.24998	o
13	13.813	14	0.01338	o
2	3.0704	4	0.23240	o

시험결과 예측오류률 25%이내로 약 83%(10/12)의 예측성공률을 나타내므로 검증기준을 만족하였다. 이는 Cyclomatic수가 오류수에 대해 예측성이 만족됨을 의미한다. 따라서 위 회귀모형을 이용하여 Cyclomatic수에 따른 오류수 즉 신뢰성을 예측할 수 있다. 명령문수 매트릭에 대해서도 위와 같은 방법으로 예측성을 시험한다.

• 식별능력 : 식별능력은 성질이 서로 다른 그룹들을 구분해낼 수 있는 능력을 말한다. 위 예에서는 Cyclomatic수에 대해 오류가 있는 모듈들과 오류가 없는 모듈들이 서로 명확히 구분되는지를 시험한다. 시험은 주로 통계적 방법을 이용한다.

• 신뢰성 : 위의 예측성 시험결과 예측성공률이 83%(검증기준은 S>80%)이므로 검증요건을 만족한다. 따라서 Cyclomatic수로 오류수(신뢰성)을 신뢰성있게 측정할 수 있다. 신뢰성을 보장하기 위해서는 충분히 많은 자료를 이용하여 시험해야 한다.

지금까지 시험한 결과로 볼때 대체로 매트릭 Cyclomatic수, 명령문수는 오류수와 검증요건을 만족하는 검증된 매트릭으로 판단된다. 위의 검증결과로 Cyclomatic수, 명령문수는 가상 소프트웨어 계, 코딩단계에서 평가 혹은 예측하는데 사용할 수 있다. 다른 품질인자에 대해서도 이러한 방법과 절차를 반복적으로 적용함으로써 소프트웨어 품질을 측정할 수 있다,

본 고에서 논의한 소프트웨어 품질매트릭 방법론을 적용함으로써 얻을 수 있는 기대효과는 다음과 같다.

• 보다 객관적이고 정량적으로 소프트웨어 품질을 측정할 수 있다.

• 체계적으로 측정활동을 할 수 있게 된다.

• 단계별 측정활동에 대해 문서화가 가능하다.

• 품질을 고려한 개발이 추진되므로 전체적으로 소프트웨어 품질이 향상된다.

6. 결론

본 고는 소프트웨어 개발자나 관리자가 품질측정활동시 실제 적용할 수 있는 방법론을 정립하고자 하였다.

소프트웨어 품질매트릭 방법론은 수명주기관점

에서 제시하였다. 수명주기는 크게 품질요구사항 정립, 품질매트릭 식별, 자료수집, 매트릭 구현, 결과분석, 매트릭 검증단계로 구분하였으며 각 단계별로 수행될 세부활동들을 체계적으로 논의하였다. 또한 방법론의 적용방법을 보이기 위해 가상의 소프트웨어, 매트릭자료를 이용하여 각 단계를 추적함으로써 방법론이 실제 적용가능함을 보였다.

본 방법론은 소프트웨어 개발방법론과 같은 수준으로 받아 들여져야 하고 소프트웨어 개발자나 관리자는 개발과 함께 소프트웨어 품질매트릭 방법론을 병행할 필요가 있다.

향후에 연구되어야 할 주요 연구내용은 본 방법론을 좀더 체계화하고 상세화하는 연구, 각 단계에서의 수행활동을 지원 품질측정 시스템개발, 방법론을 지원하는 도구(예: 자료수집과정, 검증과정)의 개발 등이다. 이외에도 매트릭을 식별하는 연구(최근 객체지향 소프트웨어에 대한 매트릭을 식별하는 연구가 수행되고 있음), 매트릭 표준화 연구, 현업에 있는 소프트웨어 개발자나 관리자가 쉽게 적용할 수 있도록 지침을 작성하는 연구 등이 수행되어야 할 것으로 판단된다.

참 고 문 헌

[1] H.L. Hausen, N.Cacutalua, D. Welzel, A Method for Software Assessment and Certification-The Informal Model, SCOPE Project, 1992.

[2] ISO, ISO 9126: Information Technology -Software Product Evaluation-Quality Characteristics and Guidelines for Their Use, 1991.

[3] ISO, ISO 9000-3: Quality Management and Quality Assurance Standards Part 3, 1991.

[4] K.H. Moller and D.J. Paulish, Software Metrics-A Practitioner's guided to improved product development, IEEE Press, 1993.

[5] Norman E. Fenton, Software Metrics-A Rigorous Approach, Chapman & Hall, 1991.

[6] Raymond A. Paul, Metrics to Improve the US Army Software Development Process-1st International Software Metrics Symposium, pp 40-50, 1993.

[7] S.K. Lee, Software Quality Engineering Workbook, 1994.