DOI: 10.3745/KIPSTD.2009.16-D.1.73

응용 오픈소스 소프트웨어 특징에 적합한 논리적 품질평가 모델에 관한 연구

김 지 혁[†] · 류 성 열[††]

요 약

오픈소스 소프트웨어는 응용 소프트웨어 개발의 60% 이상을 차지하고 있으나 이를 활용하기 위한 객관적인 품질 모델이나 지표에 대한 연구가 초보적인 수준에 머물고 있으며, 이를 평가하기 위한 정량적인 평가 기법에 대한 연구도 거의 이루어지고 있지 않다.

본 연구는 응용 소프트웨어 개발 및 유지보수에 활용할 수 있는 품질평가 모델을 제시하고, 제시된 품질평가 모델에서 활용할 수 있는 정량적 품질 평가 기법을 제시한다. 제안한 오픈소스 소프트웨어 품질 모델은 문헌 자료를 기반으로 품질 특성을 도출하고, 이를 ISO 9126의 품질 특성과 비교 검토하여 6개의 주특성과 12개의 부특성과 이를 정량적으로 측정할 수 있는 12개의 평가 메트릭으로 구성된 '논리적 오픈소스 소프트웨어 품질 모델'을 제안하였다. 제안한 품질 평가 모델의 효율성을 검증하기 위하여 오픈소스 소프트웨어 커뮤니티에 있는 상위 Rank 5개의 프로젝트 관리 시스템(PMS) 관련 소프트웨어를 대상으로 제안한 품질평가을 적용하여 그 가능성을 입증하였다.

키워드 : 오픈소스 소프트웨어, 품질평가 모델

A Study on a Logical Quality Evaluation Model based on Application Open Source Software Characteristics

Kim Ji Hyeok[†] · Rhew Sung Yul[††]

ABSTRACT

Open Source Software has over 60 percent of application software development but previous studies of objective quality model and characteristic to utilize Open Source Software appeared to be low and there are few studies regarding quantitative evaluation methods to evaluate Open Source Software.

To solve these problems, in this paper, we propose a quality evaluation model, "Logical Open Source Software quality model", which is able to utilize for developing and maintaining application software and quantitative quality evaluation method that can utilize in the proposed model. The proposed Open Source Software quality model derives quality characteristics based on literature and it forms six main-features and twelve sub-characteristics by comparing with the quality characteristic of ISO/IEC 9126 and twelve evaluation metrics that can measure the metrics and the characteristics quantitatively. To verify efficiency of the proposed quality evaluation model, we apply the proposed quality evaluation to top 5 project management system (PMS) software in open source software community and prove its availability.

Keywords : Open Source Software, Quality Evaluation Model

1. 서 론

IT 기술, 환경, 비즈니스의 변화 가속화 되면서 기업은 저비용, 고효율, 빠른 소프트웨어 개발을 원하고 있다. 이러한 이유로 오픈소스 소프트웨어(OSS: Open Source Software)를 사용한 빠른 개발 방식이 많은 관심을 받고 있으며, OSS를 사용하는 기업, 기관, 조직도 늘어나고 있는 추세이다.

초기 OSS의 사용은 시스템 소프트웨어(OS, DB)에 국한되어 있던 반면에 현재는 응용 소프트웨어 개발의 60%이상이 OSS를 활용하고 있다. 그러나 현재 OSS 활용을 위한 객관적인 품질 모델이나 지표에 대한 연구가 초보적인 단계이며, 이를 평가하기 위한 정량적인 평가 기법에 대한 연구는 전무하다.

본 연구는 응용 소프트웨어 개발 및 유지보수에 활용할 수 있는 품질평가 모델을 제시하고, 제시된 품질평가 모델

※이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-005-J03803).
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2008년 8월 14일
수 정 일 : 1차 2008년 10월 20일
심사완료 : 2008년 10월 26일

에서 활용할 수 있는 정량적 품질 평가 기법을 제시한다. 제안한 OSS 품질 모델은 문헌 자료를 기반으로 품질 특성을 도출하고, 이를 ISO 9126의 품질 특성과 비교 검토하여 6개의 주특성과 12개의 부특성으로 구성된 '논리적 OSS 품질 모델'을 제안한다. 제안한 품질 평가 모델의 유효성을 검증하기 위하여 OSS 커뮤니티에 있는 상위 Rank 5개의 프로젝트 관리 시스템(PMS: Project Management System) 관련 소프트웨어를 대상으로 수행한다.

본 연구는 다음과 같은 학문적/실용적 측면에서 OSS의 품질 모델을 연구하고자 한다. 첫째, OSS 품질에 대한 정량적인 연구이다. 둘째, OSS를 활용한 유지보수시에 품질에 대한 기준을 제공한다. 셋째, OSS 제공자에 품질에 대한 지침을 제공한다. 넷째, OSS 선정시에 기본 자료로써 활용한다.

2. 관련 연구

2.1 품질 속성에 관한 연구

ISO/IEC 9216은 단일 소프트웨어를 평가하기 위한 표준 품질 모델이다. ISO/IEC 9126은 소프트웨어 품질을 평가하기 위하여 내부·외부·사용에서의 품질에 대한 메트릭을 정의한다. 내부 메트릭은 개발 단계에 있는 소프트웨어 제품(명세서, 원시코드)에 적용할 수 있고, 외부 메트릭은 실행 가능한 소프트웨어, 즉 완제품에 해당하는 소프트웨어에 적용할 수 있다. 그리고 실제 운영 중인 소프트웨어에 대해서는 사용 메트릭을 적용할 수 있다[1]. 실제 OSS에 ISO 9126의 품질 모델을 적용할 수 있는 범위는 외부 메트릭[2]과 사용 메트릭[3]이다. 그러나 ISO/IEC 9126은 OSS의 특징을 반영하지 않은 범용적인 품질 모델이기 때문에 OSS의 특징을 반영한 품질평가 모델이 필요하다.

Wheeler[6]의 연구에서는 사용자의 요구에 따라 <표 1>과 같이 OSS의 15개의 중요 속성을 도출하였다.

Meng Huang[10]은 OSS 기반 개발 프로세스를 제안하였고, 선정 단계에서 OSS를 평가하기 위한 평가 기준을 마련하기 위해 <표 2>와 같은 OSS 속성을 제시하였다. 이 연구는 COTS 평가 프로세스에 근거하고 있으며, OSS의 품질 속성으로 안정성, 코드 이해성과 수정성, 합법성 등을 추가했다.

Kenwood[11]는 OSS 및 COTS의 도입을 비교하기 위한 분류 체계를 제안하였다. 이 연구에서는 품질 요소에 대한 분류 체계를 위해 품질 속성을 <표 3>과 같이 구분하였다.

Wheeler, Meng, Kenwood 연구는 평가를 위한 절차를

〈표 1〉 Meng Huang의 OSS 품질 속성

정확성	성능	컴포넌트간 호환성
유연성	이해가능성	분석가능성
가용성	사용용이성	버전간 호환성
설치용이성	성숙도	변경가능성
보안성	합법성	아키텍처 안정성
이식성	비용	개발도구 사용성
교육지원	개발지원	

〈표 2〉 D. A. Wheeler의 OSS 품질 속성

기능성	보안성	유지보수성/지속성
비용	저작권	유연성/수정가능성
신뢰성	기술성	시장점유율
지원	사용성	상호운용성
성능	정책	규모허용성

〈표 3〉 Kenwood의 OSS 품질 속성

변경 가능성	서비스와 지원
가용성/신뢰성	보안
상호운용성	난이도
확장성	분할의 위험
유연성	응용성
수명	성능

기반으로 품질 속성을 제시하였고, OSS 특징을 고려하여 중요한 품질 속성을 식별하였으나, 주특성/부특성에 대한 구분이 모호하고, 품질 속성에 대한 세부 메트릭을 제안하지 않았다.

2.2 OSS 평가에 관한 연구

일본 IPA(Information-Technology Promotion Agency)에서 OSS에 대한 성능과 신뢰성을 평가하여 공개하고 있다. 주요 내용은 여러 Linux 버전 사이의 상용 서버 및 OSS 서버 사이의 성능 및 신뢰성 평가를 위하여, BMT(Bench Marking Test) 절차를 통해, 만족스러운 결과를 얻을 때까지 반복하는 14 단계의 절차 및 결과를 공개하고 있다[5]. 그러나 이 방법은 OSS의 특성 및 품질 평가를 수행하는 것이 아니라, 소프트웨어 사이의 성능 평가를 기준으로 하는 것이다. 성능 뿐 아닌 다른 품질 속성에 대한 평가도 동시에 고려되어야 한다.

대부분의 오픈소스 커뮤니티(e.g., sourceforge.net, freshmeat.net 등)에서 OSS의 품질 평가 기준을 제공하지 않는다. 또한 품질 평가 기준을 제공하더라도, 표준화된 품질 모델에 의한 평가가 아니라, 사용자가 작성한 글을 기반을 평가하고 있다. 그리하여 실제 OSS를 사용할 때에 사용자는 다른 사용자가 작성한 글을 기반으로 OSS를 사용할 지를 판단하지만, 대부분의 글이 OSS의 주제나 기능과 관련된 내용이며, 이는 객관적인 품질 평가를 위한 기준 및 척도가 되지 못하고 있다.

3. OSS의 특징

본 장은 OSS 특징을 반영한 품질 속성을 도출하기 위한 사전단계로써 기존 연구를 기반으로 OSS의 특징을 식별한다[4, 6~9]. OSS의 큰 특징은소스 공개, 자발적 참여, 일찍/

자주 배포, 대규모 참여자, 커뮤니케이션과 피드백, 요구사항의 빠른 변화, 비즈니스 모델이 상업용 소프트웨어와 구분되는 것이다. 다음은 식별한 OSS 특징에 대한 설명이다.

- **소스 공개** : 소스가 공개되어 있는 것이 다른 소프트웨어에 비해 큰 특징이 될 수 있다. 또한 공개된 소스를 사용자가 임의로 바꿀 수 있음을 의미한다. 이러한 소스가 공개되어 있지만, 아직 OSS의 기타 산출물은 미약한 편이다.

- **자발적 참여** : OSS는 자발적인 참여와 기여로 만들어지는 소프트웨어 있다. 이러한 자발적인 참여는 개발자가 자신이 원하는 것을 자신에게 가장 효율적인 방법으로 개발한다는 것이다. 이를 통해 OSS의 품질을 향상시키는 요인이며, OSS를 개발하는 원동력이 된다.

- **일찍, 자주 배포** : OSS는 제품의 모든 기능이 완료될 때까지 또는 특정 모듈의 모든 기능이 완료된 후 배포하는 스타일이 아니다. Raymond[9]는 "토발즈의 개발 스타일에서 일찍, 자주 배포하는 스타일을 지향하고 있다."고 말한다. 이러한 방식에서 중요한 것은 작동 가능한 제품이 배포되었다는 것이다. 이러한 제품은 외부 메트릭으로 평가할 수 있다는 것을 말한다. 또한 일찍, 자주 배포 된다는 뜻은 OSS는 빠르게 변한다는 뜻이다. 이를 위해 OSS를 빠르게 평가할 수 있는 방법이 요구된다.

- **대규모 참여자에 의한 개발, 테스트, 디버그** : OSS의 특징 중 하나는 프로젝트 참여자의 수가 매우 대규모라는 것이다. 이러한 대규모 개발과 테스팅 및 디버깅은 병행적 개발과 디버깅, 버그 발견이 일어나는 것이다. 이러한 병행적 개발에 대한 이점을 얻기 위해 평가 요소를 식별해야 한다.

- **커뮤니케이션과 피드백** : OSS 프로젝트는 주로 온라인에서 커뮤니케이션이 이루어진다. 개발자들이 만든 프로그램을 온라인 리파지토리에 올리면, 사용자는 이 프로그램을 사용하고 버그 및 개선사항을 온라인에서 작성하여 개발자에게 피드백을 주게 된다. 이러한 온라인에서 피드백은 개별단위로써 개발자와 사용자의 커뮤니케이션이며, 전체단위로써 OSS의 세부 방향을 이끄는 수단이 된다.

- **요구사항의 빠른 변화** : 대규모 참여자와 커뮤니케이션 특징에 따라 요구사항도 빠르게 증가하고 변화하게 된다. 하지만 이러한 요구사항은 어떠한 일정한 패턴을 가지기 보다는 OSS의 진행 상황에 따라 무작위로 제시되며, 이를 프로젝트 리더가 통제하여 OSS에 반영하게 된다. 따라서 이러한 요구사항의 변화를 반영한 평가 모델이 작성되어야 한다.

- **OSS 비즈니스 모델** : OSS의 비즈니스 모델은 상업 소프트웨어와 다르다. OSS는 소스와 제품의 사용에 대해선 무료이다. 하지만, OSS가 전체적으로 무료는 아니다. OSS 자체 제품으로는 비용을 내기 힘들기 때문에, 이를 고려한 품질 평가 주특성을 식별하여야 할 것이다.

4. OSS의 품질 속성

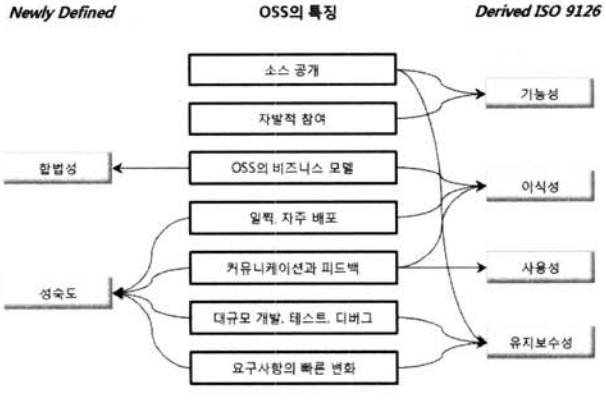
OSS의 품질을 평가하기 위한 구성으로써 주특성과 부특성을 도출한다.

4.1 OSS 특징을 고려한 품질 속성의 식별

이 절에선 3장에서 식별한 OSS의 특징을 기반으로 품질 속성을 식별한다. (그림 1)은 각 특징으로부터 도출된 주특성이다.

- **기능성** : 기능성은 소프트웨어가 제공하는 기능에 대한 역량을 나타낸다. 소스 공개 특징에 따라 목표 어플리케이션과 후보 OSS 사이의 기능/비기능에 대한 일치를 확인할 수 있으며, 소스는 지속적으로 자발적 참여에 의해 기능/비기능의 개선을 지속할 것이다. OSS의 특징을 고려하였을 때, 목표 어플리케이션의 요구 및 기능/비기능과 OSS가 제공하는 기능/비기능의 일치 및 적합 정도를 기준으로 판단할 수 있다. 따라서 기능성에 대한 부특성으로는 기능 일치성, 비기능 일치성, 적합성이 포함된다. 기능 일치성과 비기능 일치성은 목표 어플리케이션과 OSS가 가진 기능/비기능 속성을 비교하여 측정하고, 적합성은 OSS에서 명시된 기능이 목표 어플리케이션에서 적합하게 동작하는지를 판단하는 것으로 측정한다.

- **이식성** : 이식성은 같은 OSS의 하위 버전과의 적응성과, 다른 환경에서의 설치성에 대한 역량을 말한다. OSS의 비즈니스 모델은 상업 모델과 다르기 때문에 상위 버전이 하위 버전을 반드시 호환해야할 필요는



(그림 1) OSS 특징과 주특성 사이의 맵핑

없다. 또한 일찍 자주 배포되기 때문에 버전은 지속적으로 변경되며, 이러한 변경된 버전에 따라, 커뮤니케이션과 피드백도 발생할 것이다. OSS는 자주 배포되고, 환경이 다른 여러 사용자에 의해 대규모 테스트가 수행되며, 요구사항의 변화가 빠르기 때문에 이러한 적응성과과 설치성은 평가 품질로 중요한 요소가 된다. 이식성에는 설치의 용이함과 OSS가 목표 어플리케이션에 맞게 적응하는 적응 정도로 측정할 수 있다. 이러한 설치성은 설치를 쉽게, 성공적으로 수행하는 정도를 기준으로 평가한다. 적응성은 OSS가 목표 어플리케이션으로 바뀌지기 위해 기존 구조와 오퍼레이션을 얼마나 변경해야 하는지를 평가하는 것이다.

- **유지보수성** : 유지보수성은 후보 OSS를 이용한 목표 어플리케이션의 버그를 개선 및 예방하고, 향후 변화에 대처할 수 있는 역량을 말한다. OSS의 특성상 대규모 인원에 의한 테스트 및 디버그를 수행하기 때문에, 개발자가 디버그를 못할 수도 있음을 의미하고, 하나의 기능에 대해서도 개발자가 다를 수 있음을 의미한다. 또한 OSS는 요구사항이 자주 변하기 때문에 변화에 따른 테스트를 수행한다. 요구사항의 변화가 테스트에 많은 영향을 미칠 수 있다. 유지보수를 위해서는 기존 코드를 분석하고 변경하며, 변경한 것에 대해 테스트할 수 있어야 한다. 분석의 정도는 어떠한 요구사항(결함, 변경, 개선)이 미치는 구현 영역을 식별하는 역량이고, 변경은 영향을 미치는 구현 영역을 요구사항에 맞게 변경할 수 있는 역량이며, 시험성은 변경한 영역에 대해 올바로 동작하고, 다른 부분에 영향을 미치지 않는지 시험할 수 있는 역량을 말한다.

- **사용성** : 사용성은 OSS를 운영할 때, 사용자가 원하는 대로 운용하기 편하게 동작하게 하는 역량을 말한다. 자발적 참여와 커뮤니케이션/피드백에 의해 사용성은 점차 개선될 것이다. 그러나 지속적인 변화는 기존 사용자의 사용성에 큰 영향을 미칠 수 있다. 그러므로 변화에 따른 OSS의 사용성은 중요 품질 속성이 된다. 사용성은 OSS에 대한 이해와 운용으로 나뉜다. OSS의 특징 중 요구사항의 빠른 변화로 인해, 각 버전에 따라 운용 방법이 달라질 수 있다. 그러므로 OSS의 이해와 운용은 OSS를 선정하는 중요 고려사항이 된다. OSS의 사용성을 측정하는 것은 OSS를 목표 어플리케이션에 적용하기 이전에 사용하여 평가한다. 이는 다분히 주관적일 수 있음으로 다수의 의견을 모으는 것이 중요하다.

- **합법성** : 코드는 한 사람의 사상을 컴퓨터가 알아들을 수 있게 표현한 언어이다. 이러한 사상은 저작권으로 보호된다. 하지만 OSS의 경우, 코드를 다수의 이득과 자신의 만족 및 기타적인 이유로 코드를 공개하고 있다. 이렇게 공개된 코드는 비즈니스 모델에 따라 이윤을 남기기 위해 조직이 사용하는 것은 법적으로 큰 영향을 받을 수 있다. 또한 현재 적용한 OSS의 라이선스가 미래에도 계속 유지되지 않을 수 있다. 이러한 라이선스의 변경 확률도 합법성을 평가하는데 영향을 미친다. 비즈니스 모델 뿐 아니라, 소스 공개에 따라 소스를 사용하는 때, 영향을 미칠 수 있다. 또한 자발적인 참여는 오픈 소스를 이용하여 상업적 모델을 만드는데 큰 영향을 줄 수도 있다.

〈표 4〉 주특성별 식별된 부특성

주특성	부특성
기능성	기능 일치성
	비기능 일치성
	적합성
이식성	설치성
	적용성
유지보수성	분석가능성
	변경가능성
	시험성
사용성	이해가능성
	운용성
합법성	법적 허용성
성숙성	프로젝트 커뮤니티 성숙성

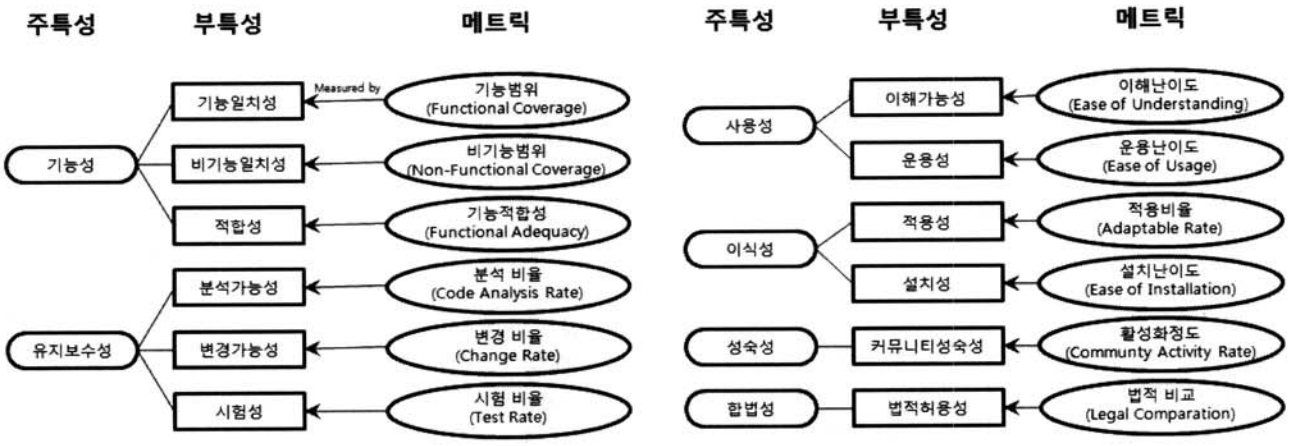
- **성숙성** : 성숙성은 소프트웨어의 역량의 질과 양, 그리고 안정성과 같은 OSS의 총체적인 역량을 뜻한다. 이러한 성숙성은 자주 배포됨에 따라 점차 성숙성이 향상되고, 배포된 제품의 커뮤니케이션/피드백으로 다음 방향을 설정하고, 대규모 디버깅과 테스트에 따라 성숙성은 점차 증진될 것이다. 그러므로 OSS를 평가를 위해 고려되어야 할 중요한 요소이다. 그러나 OSS의 성숙성을 직접적으로 평가하기는 힘들기 때문에 간접적으로 OSS 프로젝트 커뮤니티를 살펴보는 것으로 대신할 수 있다. 커뮤니티 성숙성은 커뮤니티의 활용성을 평가하여 개발자와 사용자 사이의 커뮤니케이션과 피드백을 살펴보는 것이 있다.

4.2 주특성별 부특성 식별

OSS 선정을 위한 품질 모델을 만들기 위해, 식별한 주특성과 OSS의 특징 사이의 관계를 연구함으로써 OSS의 특징이 주특성에 미치는 요인을 찾을 수 있었으며, 요인을 기반으로 부특성을 식별하였다. 주특성별 부특성을 식별한 결과는 <표 4>와 같다.

5. OSS 기반 응용 소프트웨어를 위한 메트릭

이 장은 4장에서 정의한 주특성의 부특성 및 메트릭을 정의한 것이다. 각 부특성의 정의 및 메트릭의 공식과 범위, 결과의 해석방법을 정의한다. (그림 2)는 주특성과 부특성



(그림 2) 주특성, 부특성, 메트릭 사이의 관계

메트릭과의 관계이다.

5.1 기능 일치성

기능 일치성은 기능 범위(Functional Coverage, FC) 메트릭을 통해 측정할 수 있다. 이 메트릭은 목표 어플리케이션이 가지는 기능의 수와 후보 OSS가 제공한 기능의 수의 일치된 수를 비교하여 측정한다. FC의 측정 공식은 다음과 같다.

$$FC = \frac{\text{목표 어플리케이션에 적합한 OSS 기능의 수 (X)}}{\text{목표 어플리케이션의 전체 기능 수 (Y)}}$$

FC의 수치 범위는 0~1이다. FC가 1에 가까울수록 높은 기능 일치성을 나타낸다. FC가 높다는 것은 목표 어플리케이션에 적합한 OSS의 기능의 수가 많다는 뜻이다.

5.2 비기능 일치성

비기능 일치성은 아키텍처 유사도(Architectural Commonality, AC)와 비기능 범위(Non-Functional Coverage, NFC) 메트릭으로 측정할 수 있다. 아키텍처 유사도의 경우 OSS 소프트웨어가 산출물이 체계적으로 정리되어 있지 않아[8], AC를 측정하기 어렵다. 하지만 목표 어플리케이션의 비기능 요소를 기술하였다면, 목표 어플리케이션의 전체 비기능수와 OSS의 비기능수를 비교할 수 있다(비기능 수 비교, Non-Functional Rate, NFR). NFC는 측정할 수 있다. 또한 비기능 요소의 경우 기준이 기술되어 있을 경우, OSS의 비기능 요소의 수치를 측정하고 기술된 목표 어플리케이션의 비기능 요소의 수치의 비율(비기능 기준 비율, Non-Functional Criteria Rate, NFCR)을 비교하여 NFC를 구하게 된다.

$$N = \text{목표 어플리케이션에 적합한 비기능의 수}$$

$$NFR = \frac{N}{\text{목표 어플리케이션의 전체 비기능 수}}$$

$$NFCR = \left(\sum_{i=1}^{N} \frac{\text{후보 OSS i번째 비기능 요소의 측정 수치}}{\text{목표 어플리케이션의 비기능 요소의 기준}} \right) / N$$

$$NFC = NFR * NFCR$$

NFC의 범위는 0~1이다. 1에 가까울수록 높은 비기능 일치성을 나타낸다. NFR이 높다는 뜻은 후보 OSS의 비기능이 목표 어플리케이션의 비기능의 수가 비슷하다는 뜻이다. NFCR은 비슷한 비기능에 대해 비기능의 정도가 비슷하다는 뜻이다. 결국 NFC가 높다는 뜻은 비기능 일치성이 높다는 뜻이 된다.

5.3 적합성

적합성은 명시된 기능이 제대로 작동하는지 평가하는 것이다. OSS가 일반적으로 체계적인 명세서가 없음을 고려할 때, OSS의 적합성을 판단할 수 있는 메트릭은 기능적합성(Functional Adequacy, FA) 메트릭을 통해 측정할 수 있다. 이는 목표 어플리케이션에 적합한 전체 기능을 대상으로 하며, 이에 대해 평가하여 올바로 작동하였는지 평가하는 방식이다.

$$FA = 1 - \frac{\text{평가시 문제가 발견된 기능의 수}}{\text{목표 어플리케이션에 적합한 전체 기능의 수}}$$

FA는 0~1의 범위를 가지며, 1에 가까울수록 적합성이 높다는 뜻이다. FA가 높다는 뜻은 목표 어플리케이션에서도 OSS에서의 기능이 올바르게 작동함을 의미한다.

5.4 법적 허용성

법적 허용성은 목표 어플리케이션의 비즈니스 모델이 OSS의 라이선스에 적합한지를 판단하는 것이다. 또한 이러한 법적 허용성은 차후 OSS의 라이선스가 바뀌거나 변경될 가능성을 고려하여야 한다.

법적 허용성은 법적 비교(Legal Compare, LC) 메트릭을 통해 측정할 수 있다. OSS 라이선스 변경 확률(Licence Change Probability, LCP)은 0~1의 범위를 가진다.

LCP = OSS 라이선스 변경확률
LC = 1 - ({ OSS의 라이선스 ∈ 목표 어플리케이션 비즈니스 모델 ? 1 : 0 } × LCP)

LC의 범위는 0~1까지이고, 1에 가까울수록 법적 허용성

이 좋다는 뜻이다. LC가 높을수록 목표 어플리케이션이 OSS의 법적 사용이 가능하며 이후에도 이러한 법적 사용이 지속될 것임을 의미한다.

5.5 커뮤니티 성숙성

커뮤니티 성숙성은 커뮤니티의 활성화 정도(Community Activity, CA) 메트릭을 통해 측정할 수 있다. 커뮤니티의 활성화 정도는 커뮤니티에서 OSS에 대한 커뮤니케이션(버그 보고서, 수정 보고서, 배포 보고서)의 개수를 통해 알 수 있으며, 이러한 활성화 정도는 시간에 따라 변할 수 있기 때문에, 최근 N주기의 커뮤니케이션의 수와 현재의 커뮤니케이션의 수를 비교하여 결정하는 N점 평균법을 이용한다. N과 주기(일, 주, 달, 년 등)는 목표 어플리케이션의 특성이나 비즈니스 목표에 따라 결정한다.

$$CA = \frac{\text{최근 1 주기의 커뮤니케이션 개수}}{\text{N 주기 동안의 커뮤니케이션 개수}}$$

CA는 0~1 사이의 값을 가지며, CA가 높을수록 최근 커뮤니케이션의 활성화가 높고, CA가 1에 가까울수록 커뮤니티 성숙성은 높다는 뜻이다.

5.6 설치성

설치성은 설치 난이도(Ease of Installation, EI)를 통해 측정할 수 있다. 설치 난이도는 설치 조작 난이도(Ease of Install Configuration, EIC)와 설치 성공률(Successful Install Rate, SIR) 메트릭을 통해 측정할 수 있다. 이것은 OSS의 사용자가 OSS를 얼마나 쉽게 설치하는가를 측정하는 것이다.

$$EIC = 1 - \frac{\text{설치 절차 단계 수}}{\text{사용자 입력 회수}}$$

$$SIR = 1 - \frac{\text{성공한 회수}}{\text{설치를 시도한 회수}}$$

$$EI = EIC \times SIR$$

EI는 0~1 이상의 값이 나온다. EI가 1에 가까울수록 설치성이 높다고 해석할 수 있다. EIC가 높으면 각 단계마다 사용자의 조작이 쉬움을 의미하고, SIR이 높으면 설치가 성공하기 쉽다는 뜻이다.

5.7 적용성

적용성은 적용 비율(Adaptable Rate, ADR) 메트릭으로 측정할 수 있다. OSS를 적용하기 위해, OSS 구현을 기반으로 고려해야 할 요소는 크게 OSS의 데이터 구조, 오퍼레이션의 구조로 나눌 수 있다. 구조적 적용성(Adaptability of Data Structure, ADS)과 기능적 적용성 (Adaptability of Functional Signature, AFS)을 평가한다.

$$ADS = 1 - \frac{\text{적용시 변경된 데이터 구조의 수}}{\text{OSS 전체 데이터 구조의 수}}$$

$$AFS = 1 - \frac{\text{적용시 변경된 오퍼레이션의 수}}{\text{OSS 전체 오퍼레이션의 수}}$$

W_{ADS} = OSS의 데이터 구조에 대한 가중치
W_{AFS} = OSS 오퍼레이션 구조에 대한 가중치
단, $W_{ADS} + W_{AFS}$의 합은 1이다.

$$ADR = ADS \times W_{ADS} + AFS \times W_{AFS}$$

W 값은 목표 OSS의 특징과 조직의 결정에 따라 정할 수 있는 가중치이다.

ADR은 0~1 범위를 가진다. 1에 가까울수록 높은 적응성을 가진다고 측정할 수 있다. ADS가 높음은 OSS의 데이터 구조가 목표 어플리케이션과 유사함을 의미하고 AFS가 높음은 OSS의 오퍼레이션의 구조가 목표 어플리케이션과 유사함을 의미한다. ADR이 높다는 뜻은 OSS가 목표 어플리케이션에 적용하기 쉬움을 의미한다.

5.8 분석가능성

분석 가능성은 OSS를 목표 소프트웨어로 이식 또는 적용할 때, 발생하는 문제 및 변경에 대해 기존 OSS의 코드를 구역을 식별하는 것이다. 분석 가능성은 분석 비율(Analysis Rate, AR) 메트릭을 통해 측정할 수 있다. AE는 분석 영역 비율(Analysis Area Rate, AAR)과 분석 난이도(Easy of Analysis, EA)에 영향을 받는다.

$$AAR = \frac{\text{요구사항에 관계된 코드 영역의 범위}}{\text{식별하기 위해 분석한 영역의 범위}}$$

$$EA = \frac{\text{코드 영역이 식별된 요구사항의 개수}}{\text{전체 요구사항의 개수}}$$

W_{AAR} = 영역 비율에 대한 가중치
W_{EA} = 분석 난이도에 대한 가중치
단, $W_{AAR} + W_{EA}$의 합은 1이다.

$$AR = AAR \times W_{AAR} + EA \times W_{EA}$$

W 값은 목표 OSS의 특징과 조직의 결정에 따라 정할 수 있는 가중치이다.

CAE는 0~1 사이의 수치를 가진다. 수치는 1에 가까울수록 높은 것으로 평가한다. AAR은 분석 영역을 쉽게 찾을 수 있음을 의미하며, EA는 찾고자 하는 요구사항을 쉽게 찾을 수 있음을 의미하고, AR이 높다는 것은 요구사항을 찾기 위해 적은 코드 영역을 검색하고 쉽게 찾을 수 있음을 의미한다.

5.9 변경가능성

변경 가능성은 식별된 코드 영역을 목표 소프트웨어에 적합하게 변경하는 역량이다. 변경 가능성은 변경 비율(Change Rate, CR) 메트릭을 통해 특정할 수 있다. CE는 변경 영역 비율 (Change Area Rate, CAR)과 변경 난이도(Easy of Change, EC)에 영향을 받는다.

$$CAR = 1 - \frac{\text{변경된 코드 영역의 범위}}{\text{요구사항에 관계된 코드 영역의 범위}}$$

$$EC = \frac{\text{변경 완료한 요구사항의 개수}}{\text{코드 영역이 식별된 요구사항의 개수}}$$

W_{CAR} = 영역 비율에 대한 가중치
W_{CA} = 분석 난이도에 대한 가중치
단, $W_{CAR} + W_{EC}$의 합은 1이다.

$$CR = CAR \times W_{CAR} + EC \times W_{EC}$$

W 값은 목표 OSS의 특징과 조직의 결정에 따라 정할 수 있는 가중치이다.

CR는 0~1 이상의 수치를 가진다. 수치는 1에 가까울수록 변경가능성이 높다고 평가한다. CAR은 높다는 것은 변경을 위해 적은 코드 영역만을 수정해도 됨을 의미하며, EC가 높다는 것은 요구사항에 따른 코드 영역의 변화가 쉽다는 것을 의미하며, CR이 높다는 뜻은 요구사항을 반영하기 위해 변경 영역이 작고 쉬움을 의미한다.

5.10 시험성

시험성은 시험 비율(Test Rate, TR) 메트릭을 통해 측정할 수 있다. TR은 1회 테스트를 수행할 때의 난이도(Ease of Testing Difficult, ETD)와 1개의 요구사항을 완료하기 위해 수행한 테스트 회수의 비율 (Failure Detection Test Rate, FDTR)에 영향을 받는다.

$$ETD = \frac{\text{시험을 수행하기 위한 단계 수}}{\text{시험을 수행하기 위한 사용자 조작 횟수}}$$

$$TR = \frac{ETD}{FDTR}$$

TR은 0~1의 수치를 가진다. 수치가 클수록 시험성이 높은 것으로 평가한다. ETD가 높다는 것은 1회의 테스트를 쉽게 수행할 수 있음을 의미하고, FDTR이 적다는 것은 하나의 테스트를 위해 적은 수의 테스트를 수행해도 된다는 뜻이다. TR이 높다는 뜻은 테스트가 쉽고 적은 횟수만 수행해도 됨을 의미한다.

이러한 TR은 자동화 테스트의 경우, 테스트 수를 테스트 시간으로 바꾸어 계산하여도 동일한 값이 나올 수 있다.

5.11 운용성

운용성은 운용난이도(Ease of Usage, EUS) 메트릭을 통해 평가할 수 있다. 이러한 운용성 평가는 실제 목표 어플리케이션을 만들 때, 적용할 OSS를 직접 사용하여 측정할 수 있다. OE는 사용자의 불만 사항 비율(Uncomfortable Rate, UR)과 잘못 조작하였을 때 것을 되돌리기(Undo Account, UA)와 운용에 대한 커스터마이즈 범위 (Customize Coverage)에 영향을 받는다.

$$UR = 1 - \frac{\text{불편한 기능 수}}{\text{전체 기능 수}}$$

$$UA = \frac{\text{되돌리기 가능한 기능 수}}{\text{전체 기능 수}}$$

$$CC = \frac{\text{커스터마이즈 가능한 기능 수}}{\text{전체 기능 수}}$$

W_{UR} = 불만사항 비율에 대한 가중치
W_{UA} = 영역 비율에 대한 가중치
W_{CC} = 분석 난이도에 대한 가중치
단, $W_{UR} + W_{UA} + W_{CC}$의 합은 1이다.

$$EUS = UR \times W_{UR} + UA \times W_{UA} + CC \times W_{CC}$$

W 값은 목표 OSS의 특징과 조직의 결정에 따라 정할 수 있는 가중치이다.

EUS는 0~1 사이의 값을 가진다. OA는 1에 가까울수록 운용성이 좋다고 평가할 수 있다. UR이 높다는 것은 사용자가 불편하게 생각하는 기능이 적음을 의미하고, UA가 높다는 뜻은 잘못된 기능을 수행하였을 때, 되돌릴 수 있는 가능성이 높다는 것을 의미하며, CC가 높다는 것은 기능을 사용자가 조작하기 쉽게 커스터마이즈할 수 있음을 의미한다. EUS가 높다는 것은 결국 OSS의 운용을 쉽게 할 수 있음을 의미한다.

5.12 이해가능성

이해가능성은 이해난이도(Ease of Understanding, EUN)의 메트릭으로 측정할 수 있다. 이해 비율은 기능의 명칭을 보고 기능을 예측할 수 있는 난이도(Ease of Functional Understandability, EFU), 발생한 메시지를 이해하는 비율(Ease of Understanding Message, EUM), 작업을 완료할 때까지의 최적의 조작 순서에 대한 비율(Work Path Rate, WPR)로 측정한다.

$$EFU = \frac{\text{예측 가능한 기능 명칭의 수}}{\text{전체 기능 명칭의 수}}$$

$$EUM = \frac{\text{이해하는 메시지의 수}}{\text{발생한 전체 메시지의 수}}$$

$$WPR = \frac{\text{업무 수행을 위한 최적 조작 수}}{\text{업무를 위해 사용자가 수행한 조작 수}}$$

W_{EFU} = 기능 명칭 이해난이도에 대한 가중치
W_{EUM} = 메시지 이해난이도에 대한 가중치
W_{WPR} = 조작 순서에 대한 가중치
단, $W_{EFY} + W_{EUM} + W_{WPR}$의 합은 1이다.

$$EUN = EFU \times W_{EFU} + EUM \times W_{EUM} + WPR \times W_{WPR}$$

W 값은 목표 OSS의 특징과 조직의 결정에 따라 정할 수 있는 가중치이다.

EUN는 0~1 사이의 값을 가진다. EU는 1에 가까울수록 이해가능성이 좋다고 평가할 수 있다. EFU가 높다는 것은 기능의 명칭을 보고 쉽게 기능을 예측할 수 있음을 의미하며, EUM이 높다는 것은 OSS에서 발생하는 메시지를 사용자가 쉽게 알아볼 수 있음을 의미한다. WPR이 높다는 것은 OSS의 기능을 제공하기 위해 의도대로 사용자가 이해하고

조작하고 있음을 의미한다. EUN이 높다는 것은 OSS가 제공하는 것을 사용자가 쉽게 이해하고 사용할 수 있음을 의미한다.

6. 품질평가 모델의 적정성 검증

6.1 기존 OSS 품질속성과 비교

본 연구에서 제시한 품질속성과 관련연구에서 도출된 품질속성과의 비교를 통해 도출된 품질속성의 타당성을 비교하였다. 기존 OSS 품질속성과의 비교는 <표 5>와 같은 결론을 도출했다.

〈표 5〉 기존 OSS 품질 속성과 제안한 품질 속성의 비교

OSS 품질 속성	Wheeler	Meng	Kenwood	본 연구
기능성		○		○
- 기능적합성	○		○	○
- 비기능적합성				○
- 정확성	○	○		○
- 상호운용성	○		○	
- 보안성	○	○	○	
신뢰성			○	
사용성	○			○
- 이해가능성		○		○
- 운용성		○	○	○
지원	○		○	
- 교육지원		○		
- 개발지원		○		
- 개발도구		○		
성능	○	○	○	
유지보수성	○			○
- 분석가능성		○		○
- 변경가능성	○	○	○	○
- 견고성	○	○		
- 시험가능성				○
이식성		○		○
- 유연성	○	○	○	○
- 설치성		○		○
- 컴포넌트 호환성		○		
- 버전 호환성		○		
- 확장성	○		○	
- 분할의 위험			○	
성숙성		○		○
- 커뮤니티				○
- 시장 점유	○			
- 수명			○	
- 기술성	○			
- 응용성			○	
합법성	○	○		○
비용	○	○		
정책	○			

본 연구와 기존 연구와의 차이점은 품질속성을 도출하기 위해 OSS 특징을 기반으로 하였으며, 품질속성을 측정하기 위한 정량적인 메트릭을 제공했다는 것이다. 기존 연구에서 추가된 부특성은 시험성이고, 도출된 다른 품질속성들은 기존의 연구에서 식별한 속성과 다르지는 않았다. 그러나 기존연구에서는 성능/보안/지원이라는 품질 속성을 중요한 품질 속성으로 인식하였지만, 직접적인 OSS 특징과의 연관성이 없으며 기본적인 소프트웨어 품질 속성의 측면으로써 판단되어 배제되었다.

6.2 전체적 품질의 계산

제안된 품질평가 모델을 사용하여, OSS 응용 어플리케이션 선정을 위한 전반적인 품질을 다음과 같이 계산할 수 있다.

S_{ij}는 품질평가 모델에서 정의된 i번째 주특성 / j번째 부특성을 지칭한다. 예를 들어, S_{32}는 [그림 2]에 따라 상용성 / 운용성이다. 공식에서 n은 i번째 주특성의 부특성의 개수를 의미한다.

목표 소프트웨어와 비즈니스의 목적에 따라 각 특성의 값을 계산하기 위해 다른 가중치를 부특성에 할당할 수 있다. W_{ij}의 경우, i번째 주특성 / j번째 부특성의 가중치를 의미하며, 이러한 가중치는 목표 어플리케이션과 조직에 따라 조정할 수 있는 수치이다. 만약 전체적 품질의 총합인 Q의 값의 범위는 0~V이다. 여기서 V는 W_{ij}의 총합이 된다. Q의 수치가 높으면, 목표 어플리케이션에 적용에 대한 OSS의 품질이 우수하다는 것을 의미한다.

$$Q = \sum_{i=1}^{6} \left(\sum_{j=1}^{n} S_{ij} \times W_{ij} \right)$$

6.3 품질평가 모델의 적용

품질평가 모델을 적용하기 위해 도메인을 PMS로 한정하였다. PMS의 목표 어플리케이션의 기능은 PMBOK[12]를 기준으로(Y) 44개의 기능, 각 기능별 연관 및 기능별 제약조건 및 필수요소와 수행과정, 산출물을 식별하고, 이를 기반으로 목표 어플리케이션에 대한 명세서를 작성하였다.

후보 OSS는 SourceForge.net에서 추출하였다. 도메인에 따라 Office&Business/Project Management 분류에서 PMS와 관련된 5개의 OSS를 식별하였다. SourceForge .net의 경우 Rank를 제시하고 있는데 이러한 Rank는 서비스 회수, 참여 인원, 커뮤니티 활성화 정도, 프로젝트 시작 일자, 최근 배포 일자, 다운로드 회수를 종합하여 산출하고 있다. 이렇게 산출된 SourceForge.net의 Rank를 기준으로 상위 5개의 OSS를 식별하였다. "Onepoint Project", "dotProject", "]project-open[", "OpenProj - Project Management", "GanttProject"를 후보 OSS로 선정하였다. <표 6>에서는 각 프로젝트에 대해서, A, B, C, D, E로 맵핑하였다.

FC / NFC의 경우 명세서에 명시한 기능/비기능 요소를 근거로 평가하였다. FA는 PMBOK의 프로세스를 기반으로

〈표 6〉 후보 OSS에 대한 메트릭 계산 결과

구분		후보 OSS					W
		A	B	C	D	E	
메트릭	FC	0.43	0.39	0.32	0.57	0.50	22
	NFC	0.73	0.55	0.61	0.60	0.47	16
	FA	0.44	0.38	0.72	0.53	0.49	6
	LC	1.00	1.00	1.00	0.33	0.66	12
	CA	0.31	0.27	0.42	0.19	0.22	4
	EI	0.81	0.86	0.73	1.00	1.00	7
	ADR	0.13	0.32	0.28	0.33	0.35	5
	AR	0.21	0.17	0.28	0.11	0.14	6
	CR	0.11	0.09	0.22	0.16	0.08	4
	TR	0.76	0.69	0.71	0.58	0.62	8
	EUS	0.51	0.34	0.46	0.74	0.66	3
	EUN	0.62	0.83	0.89	0.75	0.60	7
Q		56.59	52.05	55.40	55.98	54.19	100

정의역과 치역을 구하고, 이의 범위와 OSS의 정의역과 치역의 범위를 비교하여 결정하였다.

LC의 경우 후보 OSS를 기준으로 만든 소프트웨어를 수집하여 변경 확률을 구하였다.

CA 계산 시, 주기는 1달로 하였으며, N은 3으로 하였다.

ADR의 데이터구조는 PMBOK의 입력물의 구조를 기반으로 하였으며, 오퍼레이션은 기능을 기준으로 세부프로세스의 흐름으로 판단하였다.

AR은 5개의 모든 후보 OSS가 가진 기능을 찾는 것으로 판단하였다. CR은 이렇게 찾은 기능 중에 명세서에 없는 요소를 추가하는 것으로 수행하였으며, TR은 이렇게 추가한 요소가 작동하는지 테스트를 수행하여 산출하였다.

OA는 25명을 대상으로 후보 OSS를 사용하여 일정한 목적을 달성하는 방식으로 불만사항을 수집하였으며, 나머지 부특성은 명세서를 기준으로 식별한 기능에 대해 되돌리기와 커스터마이즈 여부를 조사하였다.

EU는 OA를 수행한 인원에 대해 총 4회의 미션을 주고 각 미션마다 발생하는 메시지와 조작순서 등을 기록하여 계산하였다.

각 메트릭에 사용된 W의 값은 목표 어플리케이션과 비즈니스 목적을 위해 부특성 중 가장 중요시 여기는 것은 FC, NFC, LC로 선정하였으며, 전체의 합을 100 점(%)을 기준으로 각 메트릭에 W 값을 부여하였다. 5개의 OSS 후보에 대해 메트릭을 계산한 결과는 <표 6>과 같다.

6.4 품질평가 모델의 적용 결과

<표 6>의 품질평가 모델의 결과를 기준으로 목표 어플리케이션에 적합한 OSS 후보는 A, D, C, E, B 순서가 되었다. 기능적합성(FC)만 고려하여 보았을 때, D가 가장 적용하기 좋을 수 있지만, 다른 품질을 고려한 결과 이와 같은 결과를 도출할 수 있었다. 결과는 부특성에도 중요한 영향을 받지만 W 값에도 많은 영향을 받게 된다.

SourceForge.net의 경우 대상 OSS의 Rank가 D, E, C, A, B로 차이가 나는 이유는 다음과 같다. 첫째, SourceForge의 경우 OSS를 평가하는데 커뮤니티 활성화 정도와 다운로드 수, 최근 배포 일자가 큰 영향을 미친다. 또한 기능이 많고 오래된 OSS일수록 Rank가 높게 나온다. 하지만 본 연구의 경우, 커뮤니티 활성화 정도와 최근 배포 일자를 반영하긴 하지만, 다운로드 수나 커뮤니티가 오래된 정도를 반영하진 않다. 대신 OSS 커뮤니티가 아닌 OSS 자체의 속성을 반영하게 된다. 예를 들어, SourceForge.net의 경우 목표 어플리케이션과 비교가 아니며 단지 커뮤니티의 운영에 따라 조작이 가능하다. 그러나 본 연구의 경우, 목표 어플리케이션과 비교를 통해, OSS 자체의 특징과 커뮤니티 모두 영향을 받기 때문에 SourceForge.net에서 반영하지 못하는 분석가능성, 기능/비기능 일치성 등과 같은 속성을 평가할 수 있다.

7. 결 론

본 연구는 기존의 OSS의 품질속성이나 지표에 대한 연구를 정량적인 측정이 가능한 논리적인 OSS 품질평가 모델을 제안하였다. 또한 제안한 품질평가 모델은 ISO 9126 을 기반으로 OSS 특징을 반영하여 도출하였다.

본 연구를 위해 OSS의 특징을 식별하였으며, 이를 기반으로 OSS 품질모델의 주특성을 도출하였고, 주특성을 측정하기 위한 부특성과 메트릭을 제안하였다. 최종적으로 제안한 품질모델의 검증을 위해 기존 OSS 품질속성과의 비교 및 실제 품질모델의 활용방안, 사례연구를 통한 실제 OSS의 품질평가를 수행하였다.

본 연구는 학문적, 실용적 측면에서 제안한 OSS 품질모델의 목적은 다음과 같다. 첫째, OSS 품질에 대한 정량적인 연구이다. 둘째, OSS를 활용한 유지보수시에 품질에 대한 기준을 제공한다. 셋째, OSS 제공자에 품질에 대한 지침을 제공한다. 넷째, OSS 선정시에 기본 자료로써 활용한다.

본 연구는 OSS 초기 연구에서 품질에 대한 중요성을 인지하고 이에 대한 시작으로써 OSS 특징을 반영하여 품질모델을 제안하였다. 그러나 향후 실제 적용 기업과의 연계를 통해 실용적인 검증연구와 보완적인 품질속성 연구 및 메트릭에 대한 연구를 수행해야 한다.

참 고 문 헌

[1] 'Software Engineering-Product Quality-Part 1 : Quality Model,' ISO/IEC 9126-1, 2001.

[2] 'Software Engineering-Product Quality-Part 2 : External Metrics,' ISO/IEC 9126-2, 2001.

[3] 'Software Engineering-Product Quality-Part 4 : Use in Metrics,' ISO/IEC 9126-3, 2001.

[4] T.R. Madanmohan and Rahul D., "Open Source Reuse in Commercial Firms," IEEE Computer Society, Vol.21, No.6, pp.62-69, 2004.
[5] IPA, http://www.ipa.go.jp/software/open/forum/development/index.html, 2005.
[6] David a Wheeler, "How to Evaluation Open Source Software Programs," http://www.dwheeler.com/oss-fs-eval.html, 2008.
[7] Stepan Koch, "Agile Principles and Open Source Software Development : A Theoretical and Empirical Discussion", XP 2004, pp.85-93, 2004.
[8] McConnell, S., "Open-Source Methodology: Ready for Prime Time?" IEEE Software, pp.6-8, 1999.
[9] Raymond E.S,. 'The Cathedral and the Bazaar,' O'Reilly, 1999.
[10] Meng Huang, Liguang Yang, and Ye Yang, "A Development Process for Building OSS-Based Applications," LNCS 3840, Vol.3840, pp.122-135. 2005.
[11] Carolyn A. Kenwood, 'A Business Case Study of Open Source Software,' The MITRE Corporation, 2001.
[12] PMI, 'Project Management Body of Knowledge, 3rd Ed.,' Project Management Institute INC, 2004.

김 지 혁

e-mail : jhkim78@ssu.ac.kr
2003년 숭실대학교 컴퓨터학부(학사)
2005년 숭실대학교 컴퓨터학과(공학석사)
2005년~현 재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 소프트웨어 유지보수, 소프트웨어 재사용, 서비스 지향 아키텍처, 오픈소스 소프트웨어, 소프트웨어 품질 보증

류 성 열

e-mail : syrhew@ssu.ac.kr
1981년~현 재 숭실대학교 컴퓨터학부 교수
1982년~1995년 숭실대학교 전자계산연구소 및 중앙전자계산소 소장
1997년~1998년 George Mason University 객원교수
1998년~2001년 숭실대학교 정보과학대학원 원장
2004년~현 재 한국품질재단 운영위원회 위원장
2006년~현 재 공정거래위원회 성과관리위원회위원
2008년~현 재 정보통신연구진흥원 비상임이사
관심분야 : 소프트웨어 유지보수, 소프트웨어 재사용, 오픈소프 소프트웨어