소프트웨어 패키지 평가를 위한 평가집합의 생성 및 유지를 위한 메타 모델

오 재 원[†] · 이 종 원[††] · 박 동 철[††] · 이 병 정[†††]
우 치 수[††††] · 김 순 용[†††††] · 송 기 평[††††††]

요 약

오늘날 소프트웨어 산업의 발전은 소프트웨어 패키지 제품들의 양적 팽창을 이루게 하고 있다. 이러한 급속한 소프트웨어 패키지 제품의 증가 추세에 따라서, 사용자가 선택하는 소프트웨어 제품에 대한 품질 인증 요구가 대두되었다. 공산품의 품질 인증과는 달리, 소프트웨어 제품의 경우 아직 인증 역사가 길지 않고 이를 위한 소프트웨어 품질 평가 및 인증 방법 연구는 성숙되지 않았다. 소프트웨어 제품 인증 업무 시 중요한 요소 중의 하나가 평가 집합의 체계적인 생성이다. 평가 집합이란 소프트웨어 제품 유형의 분류에 따라서 소프트웨어 품질 인증을 위한 기준과 메트릭을 명시한 체크리스트를 포함한다. 본 논문에서는 평가 집합의 체계적 생성 및 유지 관리를 위한 메타 모델을 제안한다. 그리고 메타 모델의 유효성을 확인하기 위하여 프로토타입 수준의 평가 집합을 생성한다.

A Metamodel for Creation and Maintenance of Evaluation Set of Software Package Evaluation

Jaewon Oh[†] · Chongwon Lee[††] · Dongchul Park[††] · Byungjeong Lee[†††]
Chisu Wu[††††] · Soon-Yong Kim[†††††] · Gi-Pyeung Song[††††††]

ABSTRACT

Today, the growth of software industry leads to the quantitative expansion of software package products. Due to this rapid increase of software package products, quality certification has been required for software products which users select. Unlike the quality certification of industrial products, the history of software product certification has not been so long. For this reason, software quality evaluation and certification methods have not matured yet. When certifying software products, one of most important factors is the systematic generation of evaluation sets. The evaluation sets include checklists with metrics, and criteria for the software quality certification according to the classification of software product type. This paper presents a metamodel for the systematic generation and maintenance of the evaluation sets. Then, we construct prototype level evaluation sets to show the validity of the metamodel.

키워드 : 소프트웨어 품질 인증(Software Quality Certification), 인증 메타 모델(Metamodel for Software Certification), 소프트웨어 패키지 제품(Software Package Products)

1. 서 론

오늘날 소프트웨어 산업의 발전은 소프트웨어 패키지 제품들의 양적 팽창을 이루게 하고 있다. 이러한 급속한 소프트웨어 패키지 제품의 증가 추세에 따라서, 사용자가 선택하는 소프트웨어 제품에 대한 품질 인증이 필요하게 되었다. 공산품의 품질 인증과는 달리, 소프트웨어 제품의 경우 아직 인증 역사가 길지 않다.

현재 소프트웨어 품질 인증과 관련하여 세 가지 접근 방식이 있다. 즉, 개발 인력을 인증하는 방식, 개발 조직 혹은 프로세스의 능력을 결정하는 방식, 소프트웨어 완제품 자체를 인증하는 방식이 있다[1]. 프로젝트 관리 전문가를 인증하는 기관으로는 PMI(Project Management Institute)[1], 소프트웨어 품질 전문가를 인증하는 기관으로는 ASQ(American Society for Quality)[2] 등이 있다. 개발 조직 혹은 프로세스

† 정 회 원 : 삼성전자 S/W센터 책임연구원
†† 준 회 원 : 서울대학교 대학원 컴퓨터공학부
††† 정 회 원 : 서울시립대학교 컴퓨터과학부 교수
†††† 정 회 원 : 서울대학교 컴퓨터공학부 교수
††††† 정 회 원 : 한국정보통신기술협회 이동통신시험센터 선임연구원
†††††† 정 회 원 : 한국정보통신기술협회 S/W시험센터 선임연구원
논문접수 : 2003년 1월 17일, 심사완료 : 2003년 12월 17일

1) http://www.pmi.org

능력 결정을 위한 모델로는 CMM[2], ISO/IEC 15504[3], ISO 9001[4] 등이 있다. 이 두 번째 접근 방식은 소프트웨어 품질은 제작 과정이 중요하여 프로세스 수행 능력이 소프트웨어 품질을 결정한다고 가정한다. 이에 반해 완제품 인증 방식은 소프트웨어 완제품 자체를 인증하는 것으로 아직까지 국제적인 인증 기구는 없으나 ISO/IEC 9126[5-8]을 기반으로 한 품질 인증 사례들이 이런 접근 방법에 속한다. 미국 NSTL(National Software Testing Labs)[3]과 NTS (National Technical Systems)[4]사 등이 이에 해당한다. Voas는 앞선 두 가지 접근 방법에 비해 소프트웨어 완제품 자체의 시험을 통한 품질의 측정을 더욱 중요시하고 있다[1]. 즉 개발자나 프로세스 수행 능력이 뛰어나더라도 품질이 나쁜 소프트웨어가 만들어질 수 있다고 생각하고 소프트웨어의 내부 및 외부 특성들을 여러 가지 메트릭으로 측정해 보고 품질을 판단하려는 것이다. 그렇지만 대부분의 품질 인증 방법은 위의 세 가지 접근 방식을 병합하는 형태를 띤다.

한편, 여기서 패키지 소프트웨어란 특정 수요자에 의해 주문 생산되는 것이 아니라 불특정 다수에게 판매할 목적으로 생산되는 소프트웨어를 의미하며, 워드프로세서, 스프레드시트나 데이터베이스 프로그램 등이 이에 속한다. [1]은 오늘날의 소프트웨어 개발 추세와 관련하여 개별 소프트웨어 제품 인증이 왜 필요한지를 주장하고 있다. 오늘날의 소프트웨어 개발은 처음부터 고유한 시스템을 생산하는 것에서 재사용 가능한 하위 시스템들을 획득하여 접합한 후 최종 목적 시스템을 구축하는 공정을 사용하는 방법으로 서서히 넘어가고 있다. 하위 시스템은 소프트웨어 컴포넌트나 COTS(Commercial Off The Shelf) 소프트웨어 제품이 해당된다. 그러한 상용 컴포넌트 혹은 COTS 소프트웨어 제품은 소스가 보통 공개되지 않아 품질에 이상은 없는지, 악성 코드는 들어있지 않는지 알 수 없어 검증을 하지 않고 사용할 경우 해당 소프트웨어를 이용하여 구축된 최종 목표 시스템의 품질에 의문이 제기될 수 있다. 이것이 소프트웨어 인증이 개발자나 사용자를 막론하고 왜 중요도가 높아지고 있는가를 말해준다.

또한 개발 프로세스의 능력을 결정하는 방식은 프로세스를 제도화 할만한 주로 규모가 큰 기업에 적합한 방식이다. 일반 소형 소프트웨어와 소형 콤포넌트를 개발하는 소규모 기업들은 CMM등의 성숙도 모델에 의존해 품질 보증을 하지 않고 자체적인 내부 품질 감사를 행한 후에 제품 인도를 한다[9]. 또한 [9]는 프로세스가 결함이 적은 소프트웨어 컴포넌트나 애플리케이션을 반드시 보장하지는 못 한다고 주장한다. 이러한 관점에서 볼 때, 패키지 소프트웨어 제품의 인증을 수행하기 위해서는 제3자에 의한 완제품 인증 방식이 바람직 할 수 있다. 이러한 이유로 본 논문에서는 소프트웨어 완제품의 품질 평가를 통한 인증 방식을 가정한다.

완제품에 기반한 품질 인증을 살펴보면, 소프트웨어는 유형(type)별로 서로 다른 인증 방법(certification methodology)을 요구한다[10]. 또한 사용 환경이나 소프트웨어의 특성에 따라 품질을 평가하기 위한 메트릭의 선택이 달라질 수도 있다[11-13]. 예를 들어, 네트워크 프로그램의 신뢰성과 데이터베이스 프로그램의 신뢰성을 동일한 메트릭 집합을 가지고 평가할 수는 없을 것이다. 따라서 소프트웨어를 좀 더 정확하게 평가하기 위해서는 유형별로 소프트웨어를 체계적으로 분류하고 사용 환경이나 소프트웨어의 특성을 고려할 필요가 있다.

이에 따라, 본 연구의 메타 모델은 품질요소, 소프트웨어 도메인, 소프트웨어 사용 환경으로 이루어져 있다. 이 세 가지 요소들이 소프트웨어 품질 인증의 세 가지 요소가 된다. 그러한 바탕 위에서 새로운 종류의 소프트웨어 품질 인증 방법을 명확한 분류 체계에서 도출할 수 있도록 고안된 모델이다.

본 논문의 구성은 다음과 같다. 제2장에서 소프트웨어 품질 인증과 관련된 현황을 소개한다. 제3장에서는 평가 집합의 체계적 생성 및 관리를 위한 메타 모델을 제안한다. 제4장에서 인증 모델과 프로토타입 수준의 평가 집합 생성을 통해 인증 메타 모델의 실현 가능성에 대해서 논의한다. 끝으로 제5장에서는 결론을 제시한다.

2. 관련 연구

이 장에서는 소프트웨어 품질 관련 표준안과 인증시 필요한 소프트웨어 분류와 관련해서 국내외 기관에서 채택하고 있는 소프트웨어 분류 안에 대해서 알아 보도록 한다. 그 외에, 본 연구와 관련된 기존 연구들에 대한 유사점 및 차이점을 알아본다.

2.1 소프트웨어 품질 평가를 위한 국제 표준

소프트웨어 품질은 소프트웨어가 명시적 요구 사항이나 암시적 요구 사항을 얼마나 충족하는가를 나타내는 소프트웨어 특성들(characteristics)의 총체이다[5]. ISO/IEC 9126에서는 복잡한 소프트웨어 품질을 단일 요소로 정의하지 않고, 기능성, 신뢰성 등 여러 가지 하부 요소들의 결합으로 소프트웨어 품질을 정의하고 있다.

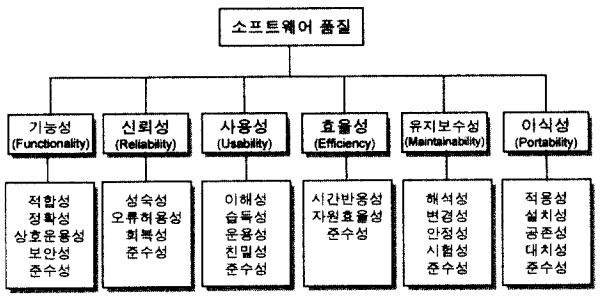
ISO(International Organization for Standardization)와 IEC(International Electrotechnical Commission)에서는 소프트웨어 품질 명세 및 평가를 위한 모델과 이에 대한 평가 프로세스[14-19]를 표준으로 정하고 있으며, 특히 ISO와

2) http://www.asq.org
3) http://www.nstl.com
4) http://www.ntscorp.com

IEC가 공동으로 설립한 JTC1(Joint Technical Committee)에서 표준화를 하는 ISO/IEC 9126에는 품질 모델이 규정되어 있다.

ISO/IEC 9126-1[5]에서는 내부 품질(internal quality), 외부 품질(external quality) 및 사용 품질(quality in use)을 표현하기 위한 품질 모델을 제시하고 있다. ISO/IEC 9126-1에서는 품질을 기술하거나 평가하기 위해 고려해야할 요소들을 특성이라 정의하며 이 특성들 또한 부특성들로 나누어 표현하고 있다((그림 1) 참조). 특성 및 부특성을 측정하는 구체적인 방법들은 ISO/IEC 9126-2[6], ISO/IEC 9126-3[7]과 ISO/IEC 9126-4[8]에서 제시되고 있다.

ISO/IEC 9126-2에서는 외부 품질을 표현하는 방법으로 외부 메트릭을 제시하고 있다. 이와 유사하게 ISO/IEC 9126-3과 ISO/IEC 9126-4에서는 각각 내부 품질과 사용 품질 측정을 위한 내부 메트릭과 사용 메트릭을 제시하고 있다.



(그림) ISO/IEC 9126 품질모델

여기서 내부 메트릭은 요구 사항 명세서, 원시 코드와 같은 실행이 불가능한 결과물에 적용하는 메트릭이다. 외부 메트릭은 소프트웨어를 실제로 수행시켜 품질을 측정하는 방법이다. 사용 메트릭은 사용자의 필요를 만족시키는 정도를 측정한다.

본 논문에서 가정하는 소프트웨어 완제품을 통한 인증 방법에서는 기본적으로 외부 메트릭을 이용하여 인증 유무를 결정한다. 제삼자인 인증기관에서 품질 분석을 위하여 개발 업체로부터 얻을 수 있는 자료는 실행 프로그램과 사용자 매뉴얼 정도이기 때문이다.

ISO/IEC 14598[14-19]은 소프트웨어 제품의 품질을 측정하거나 평가하는데 필요한 방법과 평가 프로세스를 정의하고 있는 표준이다. 이 문서에서는 품질 평가를 위한 일반적인 프로세스를 평가 요구 사항 도출, 평가 명세서 작성, 평가 계획 수립, 평가 수행 및 결과 도출 등의 단계로 구성하고 있다. 그리고 평가 명세서를 작성할 때는 ISO/IEC 9126에서 제시하는 내외부 메트릭을 활용하도록 하고 있다.

ISO/IEC 9126이 소프트웨어 품질에 대한 전반적인 내용을 다루고 있는데 반해 ISO/IEC 12119[20]는 패키지 소프트웨어에 국한하여 이러한 소프트웨어가 갖추어야 할 품질 요구 사항과 이러한 소프트웨어의 품질 평가를 위한 절차를 제시하고 있다. 패키지 소프트웨어 제품은 제품 설명서, 사용자 문서, 프로그램과 데이터로 구성이 되며 ISO/IEC 12119에서는 각 구성 요소별로 갖추어야 할 품질 요구 사항을 제시하고 있다.

2.2 소프트웨어 분류

소프트웨어 제품을 인증하기 위해서 소프트웨어 분류 작업이 선행될 필요가 있다. 왜냐하면 소프트웨어의 종류에 따라 인증에 필요한 평가 항목 혹은 메트릭들이 달라질 수 있기 때문이다[13]. 소프트웨어 분류와 관련한 표준으로 ISO/IEC 12182[21]가 존재하긴 하지만 아직까지 소프트웨어 분류에 대해 명확하게 정립된 하나의 표준이 존재하지는 않는다. ISO/IEC 12182에서는 소프트웨어를 뷰(view)의 관점에서 분류하는 것이 가능하다고 설명하고 있다.

한편, 소프트웨어를 분류하는 방법으로 다음 네 가지 접근 방법이 제시되었다[22].

첫 번째 방법으로 열거식 분류 방법은 십진 분류 방식과 유사하다. 예상되는 모든 소프트웨어를 우선 몇 개의 대그룹으로 제1차 구분을 한다. 이러한 구분을 각 대그룹에 대해 반복하여 제2차 구분을 한다. 이와 같은 구분을 되풀이하면서 차례로 세분하여 나간다. 열거식 분류는 현재 가장 많이 사용되는 방법이다. 예로써 한국정보통신기술협회의 소프트웨어 재사용 분류체계 표준[22], 한국표준산업분류[23], 북미 산업분류체계[24] 등이 이러한 열거식 분류 방법을 사용하여 대상을 분류하고 있다.

두 번째 많이 언급되는 방법으로 패싯(facet) 분류 방법은 분류 대상인 소프트웨어가 가질 수 있는 여러 가지 측면을 분류 키로 사용하는 방식이다. 예를 들면, Ruben Prieto-Diaz는 소프트웨어 부품들을 분류하기 위하여 패싯으로 기능(function), 객체(object), 매체(medium), 시스템 유형(system-style), 기능 영역(function-area), 설치 방법(setting)을 두었다[25]. 새로운 부품을 위해서는 패싯별로 적절한 값을 선택하고 이것들을 조합하면 분류가 된다.

세 번째 방법은 인덱싱(indexing) 기법을 활용하는 것으로 분류 대상의 특성을 정의하는 몇 개의 키워드와 중요도를 정의하는 방법이다.

마지막으로 위에서 나열한 방법들을 혼합하여 사용하는 방법이 있다.

앞서 밝혔듯이 아직까지도 소프트웨어 분류 작업을 위한 체계적인 표준은 존재하지 않으며 각국마다 나름대로 분류 기준을 마련하여 소프트웨어를 분류하고 있는 실정이다. 본 논문이 고려하는 소프트웨어 인증 작업에서도 소프트웨어 분류 작업이 필요하며 사용하는 분류 체계에 따라서 소프트웨어 인증의 내용이 달라지게 된다. 그러나 본 논문의 목적은 소프트웨어를 분류하는데 있는 것이 아니므로 하나의

독자적인 분류 기준을 마련하는데 중점을 두기보다는 기존 연구된 분류 체계를 메타 모델의 설명을 위해 사용할 것이다. 앞으로 본 논문에서는 현재 가장 많이 사용되는 열거식 분류 방법을 이용한 소프트웨어 분류 체계를 가정하여 설명할 것이다.

2.3 기존 소프트웨어 품질 인증 연구 분석

패키지 소프트웨어의 품질 평가 및 인증과 관련하여 몇 가지 유사 연구가 이미 발표되어 있다. 각각의 기존 유사 연구별로 간략하게 분석 한다.

[26]은 모델링 및 시뮬레이션 애플리케이션 인증에 있어 전문가의 평가를 활용할 수 있는 일관성 있는 방법론을 제시하고 있다. 제시된 방법들에 따르면, 인증 대상 소프트웨어 분야의 전문가별로 계층적 인디케이터들을 구성한다. 각 인디케이터는 상대적 가중치 및 점수를 나타내는 측정 항목이라 할 수 있다. 각 개별 전문가는 자신의 전문 영역에 속하는 인디케이터를 심사하게 된다. 각각의 인디케이터들의 심사 결과값들은 모두 최상위 계층에 있는 루트 인디케이터로 취합된다. 취합된 최종값이 인증 항목을 얼마나 만족시키는가를 여러 가지 다른 요소와 함께 고려하여 인증 여부가 결정된다. [26]은 인증 소프트웨어의 유형에 대해 잘 알고 있는 전문가가 심사하는 것을 지향하여 인증 과정이 높은 신뢰성을 추구하도록 하였다. 그러나 인디케이터에 ISO-/IEC 9126과 같은 국제적 기준을 어떻게 적용할 것인가를 명시하지 않았다.

한국전자통신연구원의 한 보고서인 [27]은 국내 패키지 소프트웨어 품질 인증을 위해 평가 항목을 정의하였다. 각 평가 항목은 "평가모듈(Evaluation Module)"의 한 요소로 정의된다. 적용시 사용 가능한 기법, 양식 및 점검표 등은 별도로 정의하여 평가의 객관성을 향상시키도록 했다. [27]의 경우, 매번 개별 소프트웨어 평가 작업시 적용 가능한 평가항목을 새로이 선정하는 작업이 필요하다. 또한 새로운 유형의 소프트웨어가 등장하는 경우 평가항목을 선정하는 데 있어서 체계적이지 못하다.

마찬가지로 한국전자통신연구원의 보고서인 [28]에서는 ProductSuite라고 하는 소프트웨어 제품 품질 평가 지원도구를 제안하고 있다. ProductSuite는 평가 절차나 방법이 아닌, 평가 항목만을 결정한다. ProductSuite는 품질 평가를 위한 각 항목을 선택한 후 체계화된 가중치 부여 방법을 사용한다. 이 방법은 선호도간의 일관성을 가정할 필요가 없어 사용자의 의사결정을 용이하게 한다. 그러나 메트릭 선정시 마다 해당 소프트웨어를 위한 체크리스트를 작성하는 임기 응변식 방법을 사용하고 있다. 이전에 인증한 동일한 유형의 소프트웨어를 인증한다고 하더라도 매번 전체 체크리스트를 작성하므로 비효율적이다.

[10]에서는 소프트웨어의 유형을 8개의 "차원(dimension)"으로 나열하였고, 모든 소프트웨어는 각각의 차원에서 하나씩의 특성을 선택하여 취합한 것이 개별 소프트웨어의 "종(species)"을 이룬다고 주장하였다. 순열 조합에 의해서 가능한 종의 개수는 수천 개가 나올 수 있다. [10]에서 소프트웨어 품질 인증은 오직 그 소프트웨어가 속한 종의 특성에만 국한되어야 한다고 정의하였다. 즉, 실시간 처리 소프트웨어는 일괄 처리 소프트웨어의 특성을 사용하여 평가할 수 없고 인증도 할 수 없다는 것을 의미한다. 그러나 [10]에서는 명확히 어떤 공인 기준에 바탕을 두어야 하는지 명시하지 않는다.

지금까지 나열한 것들을 고려해 보면, 인증에 필요한 기반 구조 위에서 중복 작업을 피할 수 있는 유연성과 계층적 분류가 포함되는 모델이 필요함을 알 수 있다.

3. 소프트웨어 품질 인증을 위한 메타 모델

이 장에서는 소프트웨어 품질 인증과 관련해서 본 논문에서 제안하는 메타 모델에 대해서 알아보도록 한다. 먼저 메타 모델의 필요성과 메타 모델을 구성하는 세 요소의 정의를 통해서 메타 모델의 개념을 정립하고, 메타 모델과 인증 모델, 평가 집합과의 관계와 메타 모델 적용 방법을 알아보도록 한다.

3.1 품질 인증 메타 모델의 정의

메타 모델은 대부분의 소규모 소프트웨어 개발 조직을 위해서는 소프트웨어 완제품 인증 방식이 더 유용하다는 인식에서 출발하였다. 인증 메타 모델이란 인증 기관이 인증 방법, 인증 대상, 인증 범위 등을 결정하기 위해서 참조하는 모델로 평가 집합을 체계적으로 생성하기 위한 방법까지 포함하는 개념이다. 즉, 인증 기관은 인증 서비스 전에 메타 모델로부터 인증 모델을 정립하며 이를 기반으로 인증과 관련된 전반적인 요소를 결정하게 되고 인증 프로세스 전체에 걸쳐 만들어진 인증 모델을 사용하게 된다.

3.1.1 요구 사항

일반적으로 소프트웨어 품질 인증이 갖추어야 할, 주요한 요구 사항은 다음과 같다.

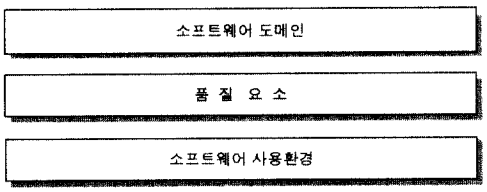
- 비교 가능성 : 인증 받은 소프트웨어 사이의 비교가 가능해야 한다.
- 정보력 : 인증은 소프트웨어를 사용하는 최종 소비자의 선택과 판단을 돕기 위한 것이므로 인증을 통해서 최종 소비자는 구체적이고 명확한 정보를 얻을 수 있어야 한다.
- 신뢰성 : 인증은 소프트웨어가 일정 수준 이상의 품질을 획득하고 있다는 것을 보증하는 것이므로 인증 받은 소프트웨어는 품질에 대해서 소비자에게 확실한 신

뢰를 줄 수 있어야 한다[30].

인증 기관이 위 세 가지 요구 사항을 만족시키면서 인증을 수행하기 위해서는 품질 인증 절차, 인증 방법, 품질 표시 방법 등 여러 가지 요소들에 대한 연구가 필요하다. 그 중에 본 논문에서는 인증 기관의 소프트웨어 품질 인증 행위를 설명하는 기준이 될 수 있으며 또한 인증 기관들이 참조하여 올바른 소프트웨어 품질 인증을 수행하는데 도움을 줄 수 있는 인증 메타 모델을 제안한다. 인증 메타 모델의 세 가지 요소는 인증 관련 정보의 범주를 지정하여 "정보력"에 기여한다. 또한 소프트웨어 분류 체계를 사용함으로써 "비교 가능성"을 높인다. 그러한 명확한 기준 하에서 인증 받은 소프트웨어는 소비자들에게서 "신뢰성"을 확보하는 것이 용이해진다.

3.1.2 메타 모델의 세 가지 요소

인증 메타 모델은 품질요소, 소프트웨어 도메인, 소프트웨어 사용 환경으로 이루어져 있다. 이 세 가지 요소들이 소프트웨어 품질 인증의 세 가지 요소가 된다((그림 2) 참조).



(그림 2) 메타 모델의 세 가지 요소

● 소프트웨어 도메인(Software domain)

소프트웨어 도메인이란 소프트웨어 분류 체계를 말한다. 소프트웨어 품질 인증 과정에서 우선적으로 선행되어야 할 작업은 소프트웨어를 적절한 기준에 따라 분류하는 것이다. 왜냐하면 소프트웨어 품질 측정 및 인증 과정은 해당 소프트웨어의 특성 혹은 종류에 따라서 달라지기 때문이다[10, 13]. 따라서 소프트웨어의 좀 더 정확한 인증을 위해서는 소프트웨어의 분류가 필요하며 이렇게 분류된 소프트웨어 그룹(群)마다 적절한 품질 측정 방법을 마련해야 한다. 또한 소프트웨어 분류를 통해서 한 그룹에 속하는, 인증을 받은 소프트웨어간의 비교가 가능해지면 소비자의 소프트웨어 구매 선택에 도움을 줄 수가 있다. 이를 위해 인증 기관은 하나의 그룹으로 분류된 소프트웨어들이 동일한 품질 요소를 기준으로 하여 비교가 가능하도록 분류 체계를 세워야 한다.

이렇게 체계적이고 효율적인 분류를 위해서 메타 모델의 소프트웨어 도메인이 가져야 할 성질들을 살펴보면 다음과 같다.

범위(Coverage) : 소프트웨어 분류에 있어 가장 기본적이면서도 중요한 속성으로 모든 소프트웨어들은 소프트웨어 분류 체계에 의해 특정 그룹으로 분류가 가능해야 한다. 아울러 하나의 그룹에 속한 소프트웨어끼리는 동일한 기준에 의해 비교가 가능해야 한다.

계층성(Hierarchy) : 제2장에서 살펴보았듯이 소프트웨어를 분류할 때 모든 종류의 소프트웨어를 일렬로 나열하지는 않는다. 비단 소프트웨어 분류뿐만 아니라 도서관에서 책을 분류할 때나 인터넷 쇼핑몰에서 물품들을 분류할 때에도 일렬로 나열하는 것을 보기 힘들다. 이러한 나열식은 소규모 분류에서 많이 쓰이며 실질적으로 대규모 분류에서는 거의 사용되지 않는다고 해도 과언이 아니다. 메타 모델에서는 소프트웨어가 기본적으로 열거식 방법으로 분류가 됨을 가정한다. 즉 대분류와 그 대분류의 상세 분류에 해당하는 중분류로 계층적으로 분류가 된다. 다시 말해서 하위 분류는 상위 분류의 부분 집합이라고 볼 수 있으며, 하위 분류는 상위 분류의 속성을 상속(inheritance)받는다. 주목할 것은 이러한 계층적인 분류 구조가 트리 구조가 아니라 그래프 구조라는 것이다. 즉, 하나의 분류가 여러 상위 분류에 속할 수 있으며, 이를 통해 다중 상속(multiple inheritance)의 개념을 이용할 수 있다.

확장성(Extensibility) : 소프트웨어 공학은 급격하게 발전하는 학문 분야 중의 하나이며 항상 새로운 수많은 소프트웨어가 생산되고 있다. 이렇듯 분류 체계는 고정되어서는 안되며 향후 새로운 소프트웨어의 분류를 위해서 확장성이라는 유연성을 가지고 있어야 한다.

● 품질 요소(Quality factor)

소프트웨어 품질은 하나의 요소에 의해 결정되는 것이 아니라 여러 가지 요소들을 함께 고려하여 결정이 된다. 메타 모델에서는 이러한 품질 평가 및 명세의 기본 단위를 품질 요소로 정의한다. (그림 1)에서 보듯 ISO/IEC 9126은 소프트웨어 품질을 6가지 요소들로 나누고 있으며 각 요소를 다시 하위 품질 요소들로 나누고 있다. 이렇듯 소프트웨어 인증 작업은 기본적으로 (그림 1)과 같은 품질 모델을 기반으로 하여 품질 요소들의 측정을 통하여 이루어진다.

소프트웨어에 대해 품질 요소들에 대한 구체적인 설명 없이 인증 여부만을 표시하는 것은 바람직하지 않다. 왜냐하면 첫 번째로 개별 품질 요소들의 정보를 손실 없이 종합하는 것이 쉽지가 않다. 두 번째 이유는, 품질 요소에 대한 내용을 상세히 알 수가 없으므로 사용자가 자신의 요구사항에 맞는 소프트웨어를 선택하는 것이 쉽지 않기 때문이다. 세 번째 이유는, 소비자마다 소프트웨어 품질 요소에 대한 기대 수준이 다를 수 있기 때문에 인증 기관에서 정한 일률적인 품질 요소 종합 기준을 모든 사용자에게 강요하는 것이 바람직하지 않기 때문이다. 그러므로 소프트웨어 품질 인증은 사용자를 고려하여 개별 품질 요소 단위로 이

루어지거나 전체 품질에 대한 인증이더라도 개별 품질 요소들에 대한 설명을 인증서에 명시할 필요가 있다.

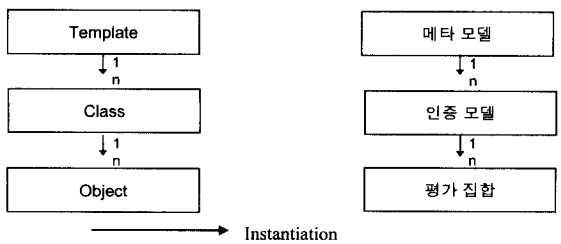
• 소프트웨어 사용 환경(Software environment)

소프트웨어의 품질을 평가할 때 소프트웨어 자체만 고려해서는 안되고 소프트웨어가 최종적으로 사용되는 실제 환경까지 고려해야 한다[11, 12]. 왜냐하면 환경에 따라 품질 측정을 위한 메트릭이 달라질 수 있기 때문이다. 여기에서 말하는 환경이란 단순히 플랫폼만을 지칭하는 것은 아니다. 컴퓨터 시스템과 환경을 포함해서 사용자 계층(user class)이나 운영 모드(operation mode), 응용 영역(application area) 등 소프트웨어가 사용되는 영역의 모든 주변 요소들을 포함하는 포괄적인 개념이다. 또한 소프트웨어 품질 인증은 소프트웨어 사용 환경을 통해서 인증의 유효 범위를 명시해야 한다. 예를 들어 일반 내장형 소프트웨어로서 안정성에 대해서 인증을 받은 소프트웨어를 의료 시스템이나 항공 시스템 등 치명성(criticality)이 높은 시스템에 사용하기에는 적합하지 않을 수 있다. 따라서 소프트웨어 사용 환경에 따라 소프트웨어가 받은 인증의 유효 범위와 한계가 명시되어야 하는 것이다.

3.1.3 메타 모델, 인증 모델과 평가 집합의 관계

이 절에서는 대표적인 객체 지향 언어인 C++에서 사용하는 개념을 이용하여 메타 모델, 인증 모델, 평가 집합간의 관계를 정립하도록 한다.

C++의 템플릿이 클래스를 생성하기 위한 상위 개념의 틀인 것처럼 메타 모델 역시 특정 인증 모델을 생성하기 위한 상위 개념의 틀이라고 할 수 있다. 즉, 메타 모델 자체가 소프트웨어 인증에 사용되는 것은 아니며, 메타 모델은 소프트웨어 인증에 직접적으로 필요한 인증 모델을 생성하기 위해서 참조하는 모델이라고 할 수 있다. 마찬가지로 클래스가 객체를 생성하기 위한 상위 개념의 틀인 것처럼 인증 모델을 통해서 실제 인증에 사용되는 평가 집합이 생성된다. 이러한 계층적 구조를 통하여 평가 집합을 생성하면 소프트웨어 인증 환경이 수시로 변경되더라도 유연한 대처가 가능할 것이다((그림 3) 참조).



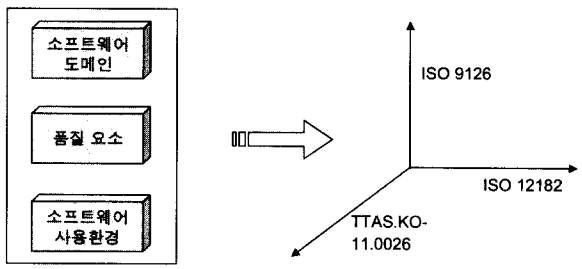
(그림 3) 메타 모델, 인증 모델, 평가 집합 상관도

3.2 품질 인증 메타 모델의 적용 방법

메타 모델의 적용은 크게 인증 모델의 생성과 평가 집합의 생성 두 단계로 나눌 수 있다. 여기에서는 인증 메타 모델을 이용하여 인증 모델과 평가 집합을 생성하는 예제를 통해서 메타 모델의 적용 방법을 알아보고자 한다.

3.2.1 인증 모델의 생성

C++의 템플릿을 이용하여 여러 가지 클래스를 생성할 수 있는 것처럼 메타 모델의 세 요소를 무엇으로 결정하느냐에 따라서 여러 가지 인증 모델이 생성될 수 있다. 여기에서는 소프트웨어 도메인으로 한국정보통신기술협회(TTA)의 소프트웨어 재사용 분류체계표준[22]을 사용하고, 품질 요소로는 ISO/IEC 9126[5], 소프트웨어 사용 환경으로 ISO/IEC 12182[21]를 사용한 인증 모델을 예제로 사용하기로 한다((그림 4) 참조).



(a) 메타 모델 (b) 인증 모델

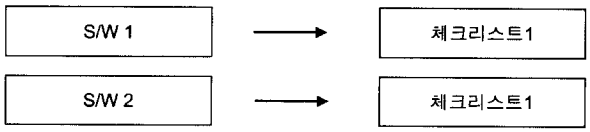
(그림 4) 메타 모델을 통해 생성한 인증 모델

3.2.2 평가 집합 생성

인증 모델이 만들어지면 이 인증 모델을 이용하여 실질적으로 인증에 사용되는 평가 집합을 생성할 수 있다. 여기에서 소프트웨어 혹은 소프트웨어 그룹마다 품질 요소별 메트릭을 선출하고 그 선출된 메트릭의 하한 값을 포함하는 체크리스트를 작성하게 된다.

평가 집합의 생성 방법으로는 크게 다음과 같은 세 가지 방법을 생각해 볼 수 있다.

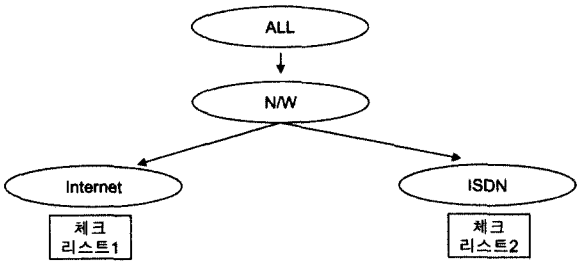
임기 응변식 방법(Ad hoc approach) : 인증을 위해 소프트웨어가 입력될 때마다 해당 소프트웨어를 위한 체크리스트를 작성하는 방식이다. 같은 그룹에 속한 소프트웨어라고 하더라도 매번 리스트를 작성하므로 비효율적이다((그림 5) 참조).



(그림 5) 임기 응변식 방법

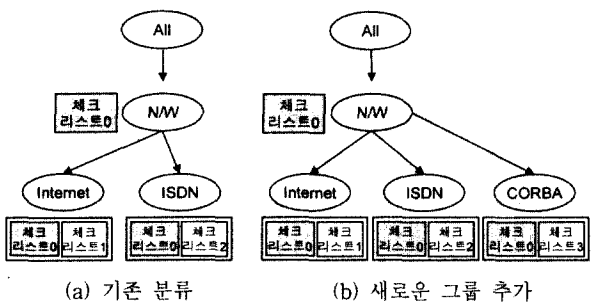
소프트웨어 그룹 기반 방법(Software-group-based approach) : 각 소프트웨어 그룹에 대해 미리 체크리스트를 마련해서 인증을 위해 소프트웨어가 입력되면 해당 그룹의 체크리스트를 참조하는 방식이다((그림 6) 참조). 이 방식은

하나의 상위 그룹에 속하는 하위 그룹들이 상위 그룹과 혹은 하위 그룹끼리 체크리스트 상에 공통적인 부분을 많이 공유하고 있음에도 불필요하게 체크리스트를 각기 생성하여 보유하는 단점을 지니고 있다.



(그림 6) 소프트웨어 그룹 기반 방법

소프트웨어 분류 체계의 계층성을 이용하는 방법(Software-group-based approach with specialization) : 앞서 언급한 바와 같이 소프트웨어 분류 체계가 가져야 할 특성 중의 하나가 계층성이다. 이 기법은 계층성을 잘 반영하여 체크리스트를 체계적으로 유지하고 관리하는 방식으로 기본적으로는 소프트웨어 그룹 기반 방법에 기반을 두고 있지만 다음과 같은 차이점이 있다. 즉, 특정 그룹에 대한 체크리스트는 상위 그룹의 체크리스트와 무관한 것이 아니기 때문에, 이 기법에서는 상위 그룹의 체크리스트에 기반해서 자신만의 필요한 체크리스트를 추가하여 완전한 체크리스트를 작성한다. (그림 7)(a)에서 보듯 네트워크(N/W)의 하위 그룹의 체크리스트는 상위 그룹의 공통 체크리스트(체크리스트 0)에 각 하위 그룹에서 필요한 리스트만 추가함으로써 체크리스트를 작성할 수 있다. 또한 이 방법은 새로운 소프트웨어 그룹이 추가되더라도 어려움 없이 확장을 할 수 있는 장점이 있다. (그림 7)(b)에서 CORBA라는 하위 그룹이 추가되더라도 기존의 체계를 유지하면서 어려움 없이 확장 가능한 것을 볼 수 있다. 따라서 메타 모델에서는 이 세 번째 방법을 평가 집합 생성 방법으로 사용한다.

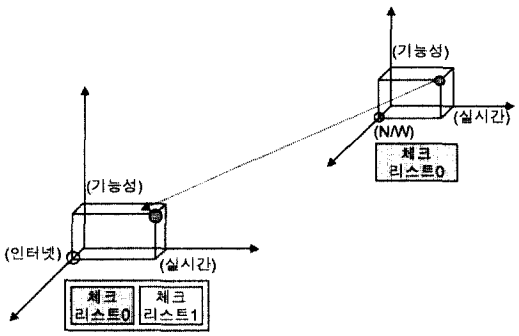


(a) 기존 분류 (b) 새로운 그룹 추가

(그림 7) 소프트웨어 분류 체계의 계층성을 이용하는 방법

앞서, 소프트웨어 도메인은 계층적으로 세분화가 되고 소프트웨어는 품질 요소별로 평가를 한다고 하였다. 이런 가정 하에서 상세화(Specialization)란 (그림 8)에서 보듯 계층적으로 체크리스트를 선출하고 종합하는 것을 말한다. 즉, 체크리스트1은 공통 체크리스트인 상위 그룹의 체크리스트 0에다 자신만의 고유한 체크리스트1'을 추가하여 생성될 수 있다.

이렇듯 체크리스트에서도 상속의 개념이 적용이 되는데 하위 리스트는 공통된 성질을 가진 상위 메트릭뿐만 아니라 자기만의 고유한 메트릭을 가질 수가 있다. 그리고 <표 1>은 메트릭과 하한 값 그리고 속성(property)으로 이루어진 체크리스트의 구조를 보여준다. 체크리스트를 반드시 이러한 세 개의 요소로만 구성을 해야 하는 것은 아니다. 경우에 따라 인증 기관은 필요로 하는 요소를 추가하거나 필요하지 않은 요소를 생략할 수 있을 것이다.



(그림 8) 상세화

〈표 1〉 체크리스트 구조((그림 8)의 체크리스트0)

Metric	Value(하한 값)	Property
M1	A	Necessary
M2	B	Optional
M3	C	Necessary

여기에서 메트릭(metric)은 품질 요소를 측정하기 위한 방법을 말하며 그 하한 값을 통해서 품질 요소의 평가가 이루어진다. 그리고 속성(property)이란 해당 품질 요소 측정을 위해서 그 메트릭이 반드시(necessary) 필요한지 아니면 선택적(optional)인지를 나타내는 요소로서 하한 값과 마찬가지로 상세화 과정에서 값의 변경이 이루어 질 수도 있다. 속성 선택을 위한 이론적 근거는 [29]의 격자 다이어그램(lattice diagram)을 기반 개념으로 차용하였다. 해당 품질 요소를 측정하는 메트릭은 여러 가지가 있으나 리스트에 모두 추가되지는 않으며 측정 불가능하거나 불필요한 메트릭은 리스트에서 제외된다. 체크리스트의 상세화 과정에서 앞서 말했듯이 메트릭의 하한 값과 속성의 값이 변경될 수가 있는데 이는 객체 지향 언어에서 상속이 가지는 특성중의 하나인 오버라이딩(overriding)의 개념과 유사하다.

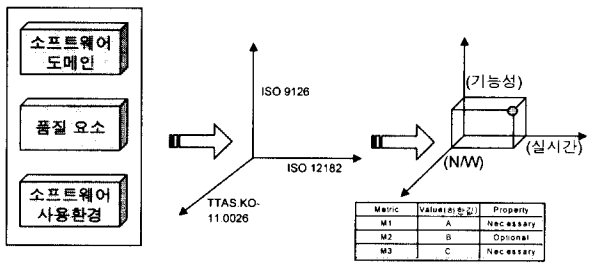
<표 1>과 <표 2>를 살펴보면 하위 체크리스트에 새로

운 M4, M5 두 개의 메트릭이 추가되었으며 상위 체크리스트<표 1>의 M1의 하한 값과 M2의 속성이 각각 하위 체크리스트에서 변경이 되었음을 알 수 있다. 이는 상위 분류에서 하위 분류로 세분화되면서 메트릭의 중요도가 달라질 수 있는 가능성을 반영한 것이다. 이러한 값의 변경은 조건이 강화되는 방향으로 일어날 수 있다. 즉 하한 값의 상승이나 "선택"에서 "필수"로의 변경이 유효하다.

〈표 2〉 체크리스트 구조((그림 8)의 체크리스트 1)

Metric	Value(하한 값)	Property
M1	***D***	***Necessary***
M2	***B***	***Necessary***
M3	***C***	***Necessary***
M4	*E*	*Optional*
M5	*F*	*Optional*

여기서 메트릭의 측정 방식까지는 정하지 않는다. 여러 가지 메트릭들 중에서 인증을 하려는 소프트웨어의 분류에 따라 필요한 메트릭이 무엇인지를 정하는 기준을 마련하는 것이 목적이다. 실제 측정은 [6]의 각 메트릭 별 측정 수식을 참조하면 된다. 예를 들어, ISO 9126의 "Functional adequacy"의 경우, 메트릭 값 산출 수식은 다음과 같다 : 1-(오작동 기능 개수/평가한 기능 개수). 해당 메트릭을 측정하기 위해선 해당 수식이 요구하는 데이터를 먼저 얻어야 한다.



(그림 9) 메타 모델, 인증 모델, 평가 집합과의 관계

(그림 9)는 지금까지 설명했던 메타 모델의 전반적인 관계를 한눈에 보여준다. 이와 같이 메타 모델은 실제 인증에 사용되는 모델이 아닌 인증 모델을 생성하기 위한 하나의 틀이 되며, 그 틀에 실질적인 요소를 적용하여 하나의 인증 모델이 만들어지게 된다. 또한 그 인증 모델을 이용하여 평가 집합이 생성되고 그 평가 집합을 이용하여 실질적으로 소프트웨어 품질 인증을 수행하게 되는 것이다.

4. 실현 가능성 검증

이 장에서는 인증 모델과 프로토타입 수준의 평가 집합 생성을 통해 인증 메타 모델의 실현 가능성에 대해서 논의한다.

4.1 인증 모델 생성

메타 모델은 소프트웨어 도메인, 품질 요소, 소프트웨어 사용 환경이라는 세 가지 요소를 가지고 있는데 이 세 가지 요소에 대해 각각 적절한 기준 혹은 방법을 선택함으로써 인증 모델이 생성된다.

먼저 소프트웨어 도메인은 소프트웨어 분류 체계로 여기에서는 한국정보통신기술협회의 소프트웨어 재사용 분류체계표준[22]을 사용하였다. 소프트웨어 분류 체계에 대한 국제적인 표준이 존재하지 않으며 각국 별로 실정에 맞게 분류 기준을 마련하고 있으므로 여기에서도 소프트웨어 도메인의 성질을 최대한 잘 만족시킬 수 있는 기준을 선택하려고 노력하였다. 앞서 제3장에서 보듯 소프트웨어 도메인은 모든 소프트웨어의 분류가 가능해야 하며, 계층성과 확장성을 가져야 한다. 이러한 특성을 소프트웨어 재사용 분류체계표준이 만족시켜주고 있다.

품질 요소로는 ISO/IEC 9126을 사용하였는데 이는 국제적인 표준으로 소프트웨어의 품질을 기술하기 위한 6가지의 요소로 이루어져 있다. 명확하게 기준이 정해지지 않은 소프트웨어 분류와는 달리 대부분 소프트웨어 품질 측정과 인증 작업에서는 이 표준안을 따르고 있다.

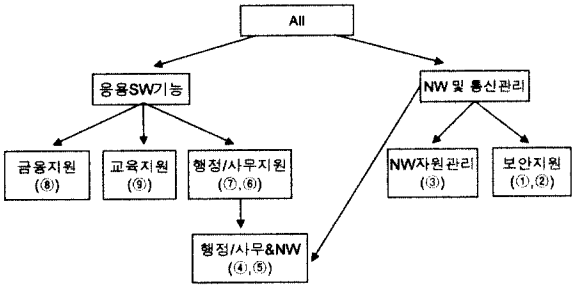
소프트웨어 사용 환경으로 여기에서는 ISO/IEC 12182를 선택하였다. 실질적으로 ISO/IEC 12182는 소프트웨어 분류와 관련된 문서이지만 우리는 ISO/IEC 12182에 나타난 16개의 뷰(View)중에서 소프트웨어 사용 환경에 맞게 선택한 6개의 뷰[5]를 사용하였다. 해당 6개의 뷰는 소프트웨어 도메인과 품질 요소에 중복이 되는 것들을 제외한 나머지 것들이다.

4.2 평가 집합 생성

제3장에서 세 가지 평가 집합 생성 방법을 설명하였으며 그중 소프트웨어 분류 체계의 계층성을 이용하는 방법을 사용하기로 하였다. 그러나 이 생성 방법은 소프트웨어 그룹별로 평가 집합이 미리 구축되어 있다는 전제에서 사용 가능한 것이며 실질적으로 처음부터 사용할 수 있는 방법은 아니다. 따라서 처음에는 우선 소프트웨어 그룹별 평가 집합을 구축하는 작업이 필요하며, 이를 위해서 연역적 방법이나 귀납적 방법을 사용할 수 있을 것이다. 여기서 연역적 방법은 소프트웨어 그룹별로 필요하다고 판단되는 메트릭을 선정하여 평가 집합을 생성하는 방법을 의미한다. 예를 들면, 특정 소프트웨어 그룹에 대해 충분한 지식을 가진 전문가가 이러한 방법을 사용하여 평가 집합을 생성할 수

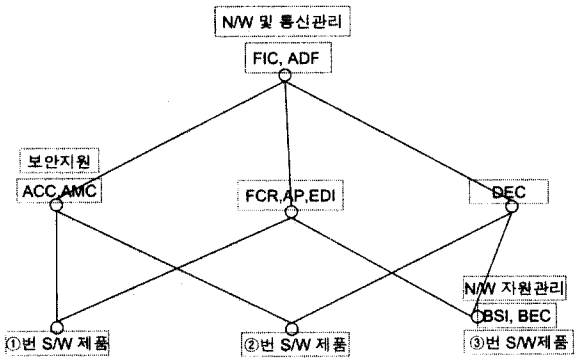
5) Operation mode, Application area, Computer system and environment, User class, Software criticality, Usage of software data

있을 것이다. 반면 귀납적 방법은 기존에 인증된 소프트웨어의 체크리스트들부터 소프트웨어 그룹별 평가 집합을 작성하는 방법을 의미한다. 즉, 특정 그룹에 속하는 소프트웨어들의 구체적인 체크리스트들로부터 그 그룹의 일반적인 체크리스트를 이끌어내는 방법이다. 이렇게 일단 평가 집합이 구축되고 나면 차후에 새로운 소프트웨어 그룹이 추가되거나 인증 환경이 변경되더라도 소프트웨어 분류 체계의 계층성을 이용하는 방법을 이용하여 쉽고 유연하게 새로운 평가 집합을 생성할 수가 있을 것이다.



(그림 10) 선택된 소프트웨어들과 분류 체계도

본 논문에서는 귀납적 방법을 이용하고자 한다. 귀납적 방법을 이용하여 평가 집합을 작성하기 위해서는 먼저 인증 받은 소프트웨어를 수집할 필요가 있다. 본 논문에서는 선택되는 소프트웨어들에게 대외적인 신뢰성을 부여하기 위해서 허가 받은 사람에 한하여 직접 열람만이 가능한 문서인 "인증 심사보고서"와 "시험성적서"를 참고하여, 한국정보통신기술협회에서 인증한 27가지의 소프트웨어들을 평가 집합 작성에 사용하였다. 한국정보통신기술협회는 S/W 제품을 평가하기 위한 기준은 ISO/IEC 9126과 ISO/IEC 12119를 기반으로 만들어진 평가 모듈(Evaluation Module Ver 1.2)을 기준으로 한다고 명시하고 있다. 해당 평가모듈은 ISO 9126에 없는 추가적인 메트릭을 포함하거나 동일한 의미를 가지나 명칭은 다른 메트릭을 갖고 있다. 그러나 근본적인 메트릭 구성은 ISO 9126과 거의 동일하다. 이러한 개별 인증에 사용된 각종 메트릭은 한국정보통신기술협회가 인증 신청업체와의 상담을 통해서 평가범위를 부특성 별로 결정하였다고 명시한 것이다. "인증 심사보고서"는 "시험기관은 인증 신청업체와의 상담을 통해서 평가범위를 부특성 별로 결정"하였다고 기술하고 있다. 또한, "시험성적서"는 "S/W 제품을 평가하기 위한 기준은 ISO/IEC 9126과 ISO-/IEC 12119를 기반으로 만들어진 Evaluation Module(Ver 1.2)를 기준으로 함"이라고 명시한다.



(그림 11) 네트워크 및 통신 관리 그룹의 라인 다이어그램

또한 "필수"와 "선택" 속성 선택 방법의 이론적인 근거를 제공하기 위하여 [29]의 개념 분석 방법을 적용하였다. 개념 분석에서는 격자 모양 라인 다이어그램을 구성하고 개념적 특징들 사이에 가장 공통적이고 연관성이 높은 것을 다이어그램 상의 최상위 부분에 위치시킨다. (그림 11)은

〈표 3〉 9 개 소프트웨어의 체크리스트

품질특성	부 특 성	메트릭 명	네트워크 및 통신 관리			응용 소프트웨어 기능					
			보안지원		NW자원	행정 및 사무지원 기능				금융지원	교육지원
			①	②	③	④	⑤	⑥	⑦	⑧	⑨
기 능 성	적 합 성	기능정보 제공 (FDI)	●		●	●	●		●		●
	적 합 성	기능 구현 완전성 (FIC)	●	●	●	●	●	●	●	●	●
	적 합 성	경계값 정보 제공 (BSI)			●	●	●		●		
	적 합 성	경계값 처리율 (BEC)			●	●	●		●	●	
	정 확 성	기능 구현 정확성 정보 제공 (AIP)	●		●	●	●		●		●
	정 확 성	기능 구현 정확성 (ADF)	●	●	●	●	●	●	●	●	●
	상 호 운 용 성	데이터 교환 정보 제공 (DEI)									
	상 호 운 용 성	데이터 교환성 (DEC)		●	●	●	●	●	●	●	
	보 안 성	접근 통제 정보 제공 (ACI)									
	보 안 성	접근 통제 가능성 (ACC)	●	●			●		●	●	
	보 안 성	접근 감시 정보 제공 (AMI)									
	보 안 성	접근 감시 가능성 (AMC)	●	●					●		
	준 수 성	기능표준 준수 정보 제공 (FCI)									
	준 수 성	기능표준 준수율 (FCR)	●		●				●	●	

네트워크 및 통신 관리 그룹 경우의 라인 다이어그램을 이용한 메트릭 속성 선택의 예를 보여준다.

반면에, 라인 다이어그램은 근본적으로 공통 특성만을 도출하므로 "선택" 메트릭은 간접적으로 유도해야 한다. 본 논문에서는 "선택" 메트릭은 해당 노드가 전체 분류 그룹 위치 노드들 중에서 잎(leaf) 노드가 되는 경우와 그렇지 않은 경우로 나누어서 판별하였다.

해당 노드가 잎 노드일 경우 : "선택" 메트릭들은 하위 분류 노드들로부터 해당 노드와 같은 수준 노드에 도달하는 경로상에 메트릭들이다.

해당 노드가 잎 노드가 아닌 경우 : 하위 노드들로부터 해당 노드에 이르는 경로에 있는 메트릭들 중에서 그 해당 노드와 하위 노드 상에 있는 메트릭들을 제외한 나머지 메트릭들의 집합이 "선택" 메트릭들이다.

평가 집합을 작성하기 위해서 우선 기반 구축을 위해 확보한 27가지 기 인증 소프트웨어들을 메타 모델의 소프트웨어 도메인으로 사용하기로 한 소프트웨어 재사용 분류체계 표준에 맞게 분류하는 작업이 필요하다. 좀 더 명확한 분류를 위해서 소프트웨어 재사용 분류체계표준을 조금 수정하였으며 적절한 분류가 가능하지 않은 소프트웨어를 제외한 나머지 소프트웨어들을 (그림 10)과 같이 분류하였다. (그림 10)은 평가 집합 생성에 사용하기 위해서 선택한 소프트웨어들(①~⑨)과 소프트웨어들의 분류 체계를 이해하기 쉽도록 도식화한 것이다. 여기서 특정 소프트웨어에 대해 적절한 분류가 가능하지 않았던 이유는 분류 체계의 문제가 아니라 그러한 소프트웨어에 대해 분류를 위한 정보가 충분하게 제공되지 않았기 때문이다.

인증 방법을 구축하기 위하여 기존에 나와 있는 소프트웨어별로 인증에 사용되었던 메트릭에 대한 정보 수집이 필요하다. <표 3>에 각 소프트웨어별 측정 메트릭의 일부가 나와있다. 즉, ISO/IEC 9126-2의 외부 메트릭 중에서 각 소프트웨어들이 선택한 메트릭에 대해서 표시(●)를 하였다.

<표 3>을 바탕으로 주요한 소프트웨어 그룹에 대한 체크리스트를 작성하면 다음과 같다.

● 네트워크 및 통신 관리

네트워크 및 통신 관리는 하위 분류로 보안지원, 네트워크 자원 관리를 가진다. 그러므로 보안 지원(①, ②)과 네트워크 자원 관리(③)의 공통 메트릭을 네트워크 및 통신 기능의 메트릭으로 선출할 수 있다. 여기에서 세 가지 소프트웨어가 공통으로 모두 갖고 있는 메트릭을 일차적으로 상위 그룹의 공통 메트릭으로 선출하되 속성을 "필수"로 하고, 네트워크 자원 관리에서 갖고 있는 메트릭을 보안 지원 그룹의 한 소프트웨어가 갖고 있으면 이 메트릭을 상위 그룹의 공통 메트릭으로 선택하되 속성을 "선택"으로 하였다 (<표 4> 참조). 이러한 "필수" 혹은 "선택" 속성 선택 설정은 [29]에서 차용한 라인 다이어그램을 이용한 유도 결과와도 일치한다. 아래의 나머지 분류 그룹들에서도 "필수" 혹은 "선택" 속성 선택 설정은 [29]의 라인 다이어그램을 통하여 근거가 뒷받침되고 있다.

〈표 4〉 네트워크 및 통신 관리 그룹의 체크리스트

(●: necessary, ○: optional)

품질특성	부 특 성	메트릭 명	속성
기 능 성	적 합 성	기능정보 제공 (FDI)	○
기 능 성	적 합 성	기능 구현 완전성 (FIC)	●
기 능 성	정 확 성	기능 구현 정확성 정보 제공 (AIP)	○
기 능 성	정 확 성	기능 구현 정확성 (ADF)	●
기 능 성	상호운용성	데이터 교환성 (DEC)	○
기 능 성	준 수 성	기능표준 준수율 (FCR)	○
신 뢰 성	성 숙 성	문제 해결률 (PRR)	○
신 뢰 성	성 숙 성	결함 회피율 (FOR)	○
신 뢰 성	오류허용성	다운 회피율 (DAR)	○
신 뢰 성	오류허용성	고장 회피율 (FAR)	○
신 뢰 성	오류허용성	오조작 회피율 (IOA)	●
효 율 성	시간반응성	평균 반응시간 (MST)	○
사 용 성	이 해 성	인터페이스 이해도 (IUA)	●
사 용 성	이 해 성	도움말 이해도 (HUA)	○
사 용 성	이 해 성	입/출력 데이터 이해도 (DUA)	●
사 용 성	이 해 성	인터페이스 일관성 (IFC)	●
사 용 성	이 해 성	내용 일관성 (CTC)	●
사 용 성	이 해 성	사용자 안내성 (UGA)	●
사 용 성	습 득 성	기능 학습 용이 정도 (FLA)	●
사 용 성	습 득 성	도움말 접근 용이 정도 (EHA)	○
사 용 성	운 용 성	오류 방지 정도 (EPA)	●
사 용 성	운 용 성	메시지 이해 용이 정도 (MRA)	●
사 용 성	운 용 성	진행상태 파악 가능 정도 (PIA)	●
사 용 성	친 밀 성	인터페이스 선호도 (IAL)	●
유지보수성	해 석 성	문제해결 구현률 (PRR)	●
유지보수성	변 경 성	변경 가능률 (CFR)	●
이 식 성	적 응 성	적용환경 적응률 (AFR)	●
이 식 성	설 치 성	설치 정보 제공 (IIP)	○
이 식 성	설 치 성	설치 가능률 (IAR)	●
이 식 성	설 치 성	제거 정보 제공 (UIP)	○
이 식 성	설 치 성	제거 가능률 (UIR)	●
이 식 성	대 치 성	기능 지속 가능률 (FCR)	○
이 식 성	공 존 성	공존 가능률 (EIR)	●

● 보안지원

상위 그룹인 네트워크 및 통신 관리에는 포함되지 않으면서 보안지원 그룹에만 속하는 메트릭을 선출한다. 여기에서 두 소프트웨어 모두에 포함된 메트릭은 속성을 "필수"로, 하나에만 나타난 메트릭의 경우 "선택"으로 하였다<표

5 참조>.

〈표 5〉 보안지원 그룹의 체크리스트

(● : necessary, ○ : optional)

품질특성	부 특 성	메트릭 명	속성
기 능 성	보 안 성	접근 통제 가능성 (ACC)	●
기 능 성	보 안 성	접근 감시 가능성 (AMC)	●
신 뢰 성	회 복 성	복구 가능율 (RAR)	○
효 율 성	시간반응성	평균 처리시간 (MTT)	○
유지보수성	해 석 성	진단 기능 지원 (DFS)	○
이 식 성	적 응 성	데이터구조 적응률 (DAR)	○

〈표 6〉 응용 소프트웨어 기능의 체크리스트

(● : necessary, ○ : optional)

품질특성	부특성	메트릭 명	속성
기 능 성	적 합 성	기능정보 제공 (FDI)	○
기 능 성	적 합 성	기능 구현 완전성 (FIC)	●
기 능 성	적 합 성	경계값 처리율 (BEC)	○
기 능 성	정 확 성	기능 구현 정확성 정보 제공 (AIP)	○
기 능 성	정 확 성	기능 구현 정확성 (ADF)	●
기 능 성	상호운용성	데이터 교환성 (DEC)	○
기 능 성	보 안 성	접근 통제 가능성 (ACC)	○
기 능 성	준 수 성	기능표준 준수율 (FCR)	○
신 뢰 성	성 숙 성	결함 회피율 (FOR)	●
신 뢰 성	성 숙 성	결함발생 평균시간 (MTBF)	○
신 뢰 성	오류허용성	다운 회피율 (DAR)	●
신 뢰 성	오류허용성	고장 회피율 (FAR)	●
신 뢰 성	오류허용성	오조작 회피율 (IOA)	●
신 뢰 성	회 복 성	데이터 회복률 (DRR)	○
효 율 성	시간반응성	평균 반응시간 (MST)	●
사 용 성	이 해 성	인터페이스 이해도 (IUA)	●
사 용 성	이 해 성	도움말 이해도 (HUA)	○
사 용 성	이 해 성	입/출력 데이터 이해도 (DUA)	○
사 용 성	이 해 성	인터페이스 일관성 (IFC)	●
사 용 성	이 해 성	내용 일관성 (CTC)	●
사 용 성	이 해 성	사용자 안내성 (UGA)	○
사 용 성	습 득 성	기능 학습 용이 정도 (FLA)	●
사 용 성	습 득 성	도움말 접근 용이 정도 (EHA)	○
사 용 성	운 용 성	오류 방지 정도 (EPA)	●
사 용 성	운 용 성	메시지 이해 용이 정도 (MRA)	●
사 용 성	운 용 성	운영절차 조정 정보 제공 (OCI)	○
사 용 성	운 용 성	진행상태 파악 가능 정도 (PIA)	○
사 용 성	친 밀 성	인터페이스 선호도 (IAL)	●
유지보수성	해 석 성	문제해결 구현률 (PRR)	●
유지보수성	변 경 성	변경 가능률 (CFR)	○
이 식 성	적 응 성	적응환경 적응률 (AFR)	●
이 식 성	설 치 성	설치 정보 제공 (IIP)	○
이 식 성	설 치 성	설치 가능률 (IAR)	●
이 식 성	설 치 성	제거 정보 제공 (UIP)	○
이 식 성	설 치 성	제거 가능률 (UIR)	●
이 식 성	공 존 성	공존 가능률 (EIR)	●

● 응용 소프트웨어 기능

응용 소프트웨어 기능은 하위 분류로 행정 및 사무지원 기능(④~⑦), 금융지원(⑧), 교육지원(⑨) 세 가지 그룹으로 나누어져 있다. 일단 6개 소프트웨어가 모두 가지고 있는 공통 메트릭을 "필수" 속성으로 뽑아내었다. 그 다음 6개의 소프트웨어가 모두 가지고 있지는 않지만 세 그룹 중 두 그룹 이상에서 가지고 있는 메트릭을 "선택" 속성으로 선출하였다(<표 6> 참조).

● 행정 및 사무지원 기능

행정 및 사무지원 기능에 속한 소프트웨어에는 4가지(④~⑦)가 있는데 이중 ④와 ⑤는 네트워크 및 통신 관리 그룹의 속성도 함께 상속을 받는다. 일단 이 네 가지 소프트웨어의 공통적인 메트릭 중에서 상위 그룹인 응용 소프트웨어 기능 그룹의 메트릭에는 포함되지 않으면서 행정 및 사무지원 그룹에만 속하는 메트릭을 속성을 부여하여 뽑아내었다. 단, 여기에서는 선택되어진 모든 메트릭들의 속성이 "선택"이었다(<표 7> 참조).

〈표 7〉 행정 및 사무지원 기능의 체크리스트

(● : necessary, ○ : optional)

품질특성	부특성	메트릭 명	속성
기능성	적 합 성	경계값 정보 제공 (BSI)	○
기능성	보 안 성	접근 감시 가능성 (AMC)	○
신뢰성	회 복 성	평균 복구 시간 (MRT)	○
효율성	시간반응성	평균 처리율 (MTM)	○
효율성	자원효율성	I/O 자원 사용률 (IOU)	○
사용성	운 용 성	에러 복구 용이 정도 (ERA)	○
사용성	친 밀 성	인터페이스 조정 가능 정도 (ICA)	○
이식성	적 응 성	데이터구조 적응률 (DAR)	○
이식성	적 응 성	이식 편리성 (PUF)	○
이식성	대 치 성	데이터 지속 정보 제공 (DCP)	○
이식성	대 치 성	데이터 지속 가능률 (DCR)	○
이식성	대 치 성	기능 지속 가능률 (FCR)	○

● 행정 및 사무지원 & 네트워크 및 통신 관리

④와 ⑤ 두 개의 소프트웨어는 행정 및 사무지원 그룹과 네트워크 및 통신 관리 그룹으로부터 다중 상속을 받는다. 따라서 이 그룹의 최종 체크리스트는 응용 소프트웨어 그룹과 그 하위 행정 및 사무지원 그룹의 메트릭을 포함하고 아울러 네트워크 및 통신기능 그룹의 메트릭도 포함하게 된다. 이 경우 이 그룹만의 메트릭을 선출하려면 위 세 그룹에 포함되는 메트릭을 제외한 나머지 메트릭을 뽑아내야 하는데 이번 예제의 경우에는 아무런 메트릭이 나타나지 않았다.

4.3 평가 집합 생성 시의 문제점 및 해결책

선택된 9개의 소프트웨어와 그 인증 자료를 이용하여 귀

납적으로 그룹별 체크리스트를 작성해 보았다. 이번 절에서는 그룹별 공통 메트릭을 선출하면서 나타난 문제점과 그 해결책을 생각해 보도록 하겠다.

첫 번째로 그룹별 체크리스트의 신뢰성 문제를 꼽을 수 있다. 소프트웨어 그룹에 대해 공통 메트릭을 선출할 때 이 그룹에 속하는 소프트웨어의 수가 많아질수록 그만큼 선출된 공통 메트릭의 신뢰도가 높아질 것이다. 이러한 문제는 공인 인증기관이 본 연구의 제안을 활용하여 많은 기 인증 소프트웨어 관련 데이터를 참조할 수 있다면 해결이 가능한 문제라 할 수 있다. 소프트웨어 분류 영역 별 전문가들을 활용하여 그룹별 체크리스트를 보다 전문적인 입장에서 재구성하는 방법도 있다.

두 번째로 공통 메트릭 선별을 위한 명확한 기준을 정립할 필요가 있다. 상위 그룹은 여러 개의 하위 그룹으로 나누어지며 이러한 하위 그룹들의 메트릭으로부터 상위 그룹의 공통 메트릭이 선출될 수 있다. 개념분석[29]을 이용하여 유도할 수 있고, 추가적으로 이 과정에서 하위 그룹들이 모두 가지고 있는 메트릭의 경우는 이 메트릭을 상위 그룹의 공통 메트릭으로 선출하고 속성을 "필수" 혹은 "선택"으로 하면 합리적일 것이다. 그러나 하위 그룹 중 일부만 갖고 있는 메트릭의 경우는 이 메트릭을 공통 메트릭으로 보고 상위 그룹의 "선택"인 메트릭으로 선출할지, 아니면 공통 메트릭이 아닌 것으로 판단하여 각 하위 그룹의 메트릭으로 볼지 결정해야 하는 문제가 생기게 된다. 분류 그룹의 메트릭 자체 선정은 공인 인증기관에서 많은 경험이 있는 도메인 전문가를 활용[26]하거나 많은 기 인증 소프트웨어 관련 데이터를 이용한 통계적인 분석 방법을 고려해 볼 수 있다. 예를 들면, 사용자들의 설문을 통계 처리한 후, 사용자 설문 분석에 의한 메트릭 집합을 도출한 뒤, 실제 메트릭 심사 결과 값을 사용자 만족도 설문 조사 값과 대비하여 상호 연관도가 높을 때 해당 메트릭들을 해당 소프트웨어의 품질을 반영한다고 간주하는 방법 등이 있다[31].

5. 결 론

소프트웨어는 유형 별로 서로 다른 인증 방법을 요구한다. 왜냐하면 소프트웨어를 사용하는 환경 또는 소프트웨어의 특성에 따라 품질을 평가하기 위한 메트릭의 선택이 달라질 수 있기 때문이다. 현재 새로운 유형의 소프트웨어를 포함하여 소프트웨어가 급진적으로 증가하고 있으며 소프트웨어가 사용되는 영역 또한 확장되고 있다. 인증 기관이 이러한 상황에 유연하게 대처하기 위해서는 새로운 인증 방법을 보다 적은 비용으로 신속하게 개발할 수 있어야 할 것이다. 이에 본 논문에서는 품질 인증을 위한 메타 모델을 제안하였다.

메타 모델이란 인증 기관이 인증 모델, 평가 집합을 생성하기 위해서 참조하는 모델로 평가 집합을 체계적으로 생성하기 위한 방법까지 포함하는 개념이다. 이러한 메타 모델의 세 요소에 적절한 기준을 적용함으로써 인증 모델과 평가 집합을 생성할 수가 있다. 이러한 메타 모델의 실현 가능성을 검증하기 위하여 프로토타입 수준의 평가 집합을 생성해 보았다.

프로토타입 수준의 평가 집합 생성 과정에서 살펴보았듯이, 소프트웨어 그룹별로 적합한 메트릭을 선택하거나 메트릭의 하한 값 등의 품질 평가 기준을 결정하는 일은 인증에 직접적인 영향을 주는 민감한 사항인 만큼 결코 간단하지 않다. 또한 귀납적인 방법에 의해 적절한 체크리스트를 작성하기 위해서는 실제 수많은 인증 작업의 결과물이 필요하다.

향후 과제로 공인 인증기관의 많은 경험이 있는 도메인 전문가의 지식을 데이터베이스에 축적하여 그룹별 평가 집합 생성 과정을 자동화하는 방법과 메타 모델을 지원하는 도구의 개발을 고려할 필요가 있다.

참 고 문 헌

[1] Jeffrey Voas, "Certification : Reducing the hidden costs of poor quality," IEEE Software, Vol.16, No.4, pp.22-25, 1999.

[2] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, Feb., 1993.

[3] ISO/IEC, "TR 15504-1 Information technology-Software process assessment-Part 1 : Concepts and introductory guide," ISO/IEC JTC1/SC7, 1998.

[4] ISO, "9001 Quality management systems-Requirements," ISO TC176/SC2, 2000.

[5] ISO/IEC, "FCD 9126-1.2 Information Technology-Software product quality-Part 1 : Quality model," ISO/IEC JTC1/SC7 N1949, 1998.

[6] ISO/IEC, "DTR 9126-2 : Software Engineering-Product Quality Part 2-External Metrics," ISO/IEC JTC1/SC7 N2419, 2001.

[7] ISO/IEC, "DTR 9126-3 : Software Engineering-Product Quality Part 3-Internal Metrics," ISO/IEC JTC1/SC7 N2416, 2001.

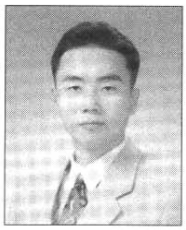
[8] ISO/IEC, "DTR 9126-4 : Software Engineering-Product Quality-Part 4 : Quality In Use Metrics," ISO/IEC JTC1/SC7 N2430, 2001.

[9] William T. Council, "Third-Party Testing and the Quality of Software Components," IEEE Software, Vol.16, No.4, pp.55-57, 1999.

[10] Jeffrey Voas, "Limited software warranties," Proceedings of the 7th IEEE International Conference and Workshop

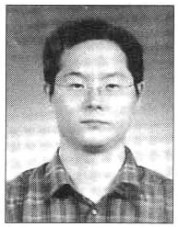
on the Engineering of Computer Based Systems, pp.56-61, 2000.
[11] Nigel Bevan, "Measuring usability as quality of use," Software Quality Journal, Vol.4, pp.115-150, 1995.
[12] Nigel Bevan, "Usability is Quality of Use," Proceedings of the 6th International Conference on Human Computer Interaction, 1995.
[13] 최은만 등, "소프트웨어 제품 메트릭과 데이터분석 기술에 관한 연구", 한국정보과학회 소프트웨어공학회지, 제14권 제4호, pp.62-73, 2001.
[14] ISO/IEC, "Information technology - Software product evaluation - Part 1 : General overview," ISO/IEC 14598 - 1 : 1999(E), 1999.
[15] ISO/IEC, "Software engineering - Product evaluation - Part 2 : Planning and management," ISO/IEC 14598-2 : 2000(E), 2000.
[16] ISO/IEC, "Software engineering - Product evaluation - Part 3 : Process for developers," ISO/IEC 14598 - 3 : 2000 (E), 2000.
[17] ISO/IEC, "Software engineering - Product evaluation - Part 4 : Process for acquirers," ISO/IEC 14598 - 4 : 1999 (E), 1999.
[18] ISO/IEC, "Information technology - Software product evaluation - Part 5 : Process for evaluators," ISO/IEC 14598 - 5 : 1998(E), 1998.
[19] ISO/IEC, "Software engineering - Product evaluation - Part 6 : Documentation of evaluation modules," ISO/IEC 14598 - 6 : 1999(E), 1999.
[20] ISO/IEC, "Information technology-Software packages - Quality requirements and testing," ISO/IEC 12119 : 1994 (E), 1994.
[21] ISO/IEC, "Information technology - Categorization of software," ISO/IEC TR 12182 : 1998(E), 1998.
[22] 한국정보통신기술협회(TTA), "소프트웨어 재사용 분류체계표준", TTAS.KO-11.0026, 2000.
[23] 통계청, "한국표준산업분류," 통계청 고시 2000-1호, 2000.
[24] U.S. Census Bureau, "North American Industry Classification System," 2002.
[25] T. Capers Jones, "Reusability in Programming : A Survey of the State of the Art," IEEE Transactions on Software Engineering, Vol.10, No.5, pp.488-494, 1984.
[26] Osman Balci, "A Methodology for Certification of Modeling and Simulation Applications," ACM Transactions on Modeling and Computer Simulation, Vol.11, No.4, pp.352-377, Oct., 2001.
[27] 김길조 등, "S/W 품질 향상을 위한 표준 연구", 한국전자통신연구원(ETRI), Dec., 2000.
[28] 안유환 등, "소프트웨어 품질 평가 기술 개발", 한국전자통신연구원(ETRI), Dec., 1999.
[29] B. Ganter and R. Wille, "Formal Concept Analysis, Mathematical Foundation," Springer, 1998.
[30] Keith Miller and Jeffrey Voas, "An ethical can of worms for software certifiers," IT Professional, Vol.1, No.5, pp.18-20, 1999.
[31] Hareton K. N. Leung, "Quality Metrics for Intranet Applications," Information and Management, Vol.38, pp.137-152, 2001.

오 재 원



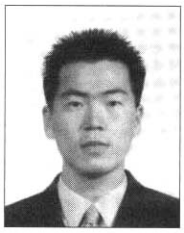
e-mai. : jwoh@selab.snu.ac.kr
1997년 서울대학교 계산통계학과(학사)
1999년 서울대학교 대학원 전산과학과 (이학석사)
1999년~2004년 서울대학교 대학원 컴퓨터 공학부(박사)
2004년~현재 삼성전자 S/W센터 책임연구원
관심분야 : 소프트웨어 품질 시험/평가/인증, 실시간 태스크 스케줄링, 분산객체기술

이 종 원



e-mai. : ljw@selab.snu.ac.kr
2001년 연세대학교 정보산업공학과(학사)
2003년 서울대학교 대학원 컴퓨터 공학부(석사)
2003년~현재 서울대학교 대학원 컴퓨터 공학부 박사과정
관심분야 : 소프트웨어 품질 시험/평가/인증

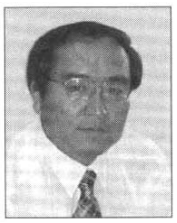
박 동 철



e-mail : tongss@selab.snu.ac.kr
2001년 중앙대학교 컴퓨터공학과(학사)
2003년 서울대학교 대학원 컴퓨터공학부 (석사)
관심분야 : 소프트웨어 품질 시험/평가/인증, 소프트웨어 재공학

이 병 정

e-mail : bjlee@venus.uos.ac.kr
1990년 서울대학교 계산통계학과(학사)
1990년~1998년 현대전자 소프트웨어 연구소 주임연구원
1998년 서울대학교 대학원 전산과학과 (석사)
2002년 서울대학교 대학원 컴퓨터공학부(박사)
2002년~현재 서울시립대학교 컴퓨터과학부 조교수
관심분야 : 소프트웨어 품질 시험/평가/인증, 소프트웨어 재공학, 웹 공학



우 치 수

e-mail : wuchisu@selab.snu.ac.kr

1965년~1972년 서울대학교 응용수학과 (학사)

1972년~1974년 한국과학기술원 연구원

1975년~1977년 서울대학교 대학원 전산과학과(석사)

1977년~1982년 서울대학교 대학원 전산과학과(박사)

1978년 영국 라퍼러 대학 연구원

1975년~1982년 울산대학교 전자계산학과 조교수, 부교수

1985년~1986년 미국 미시간대학교 postdoc

1988년~1990년 한국정보과학회 총무 이사

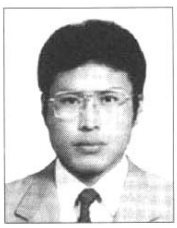
1990년~1992년 한국정보과학회 학회지 편집위원장

1992년~1993년 한국정보과학회 부회장

1996년~1998년 한국정보과학회 소프트웨어공학연구회 운영위원장

1982년~현재 서울대학교 컴퓨터공학부 교수

관심분야 : 소프트웨어 품질 시험/인증



김 순 용

e-mail : syk@tta.or.kr

1981년 시스템공학연구소 선임연구원

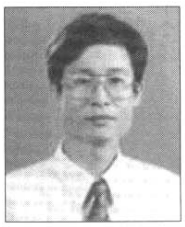
1998년 한국전자통신연구원 선임연구원

2001년 한국전자통신연구원 선임연구원

2001년 한국정보통신기술협회 S/W시험센터 선임연구원

2003년~현재 한국정보통신기술협회 이동통신시험센터 선임연구원

관심분야 : 소프트웨어 품질 시험/평가/인증, 휴대폰 시험/인증



송 기 평

e-mail : gpsong@tta.or.kr

1980년 한국전자통신연구원 선임연구원

2001년~현재 한국정보통신기술협회 S/W시험센터 선임연구원

관심분야 : 소프트웨어 품질 시험/인증