소프트웨어 시험평가 모델 설계에 관한 연구 (공공분야 CALS/EC 중심으로)

임만택, 이상호, 김성용, 변재정
국방과학연구소

Abstract

Nowadays, public domain software CALS/EC projects give their attention to the importance of test and evaluation (T&E). This article proposed a survey on T&E procedures of the international standards ISO 9000-3, ISO/IEC 9126 and ISO 14598, and of the U.S. Army pamphlet as well as that of MIL-STD-498. And features of testing tools such as their characteristics were reviewed for a reference in selecting appropriate one.

Large-scale software projects often encounter with problems during T&E, when they become close to the end of development cycle. The problems generally act as a bottleneck of a project, leading to slippage of the entire schedule. This article tried a close look at the problems and solution alternatives of the T&E policy, management, and technical issues, and proposed a T&E model to support a variety of software development environment.

Ⅰ. 들어가는 말

최근 공공분야의 많은 소프트웨어 개발사업이 정부관리 업체주도 개발형태로 추진되고 있으며, 제한된 자원을 효율적으로 활용하여 경제적으로 소프트웨어를 개발 및 운영하기 위해서는 소프트웨어를 개발하는 수명주기과정에서 시험평가가 중요하다는 사실이 인식되고 있다. 날로 발전하는 정보기술과 다양하고 복잡한 사용자의 요구사항을 고려한 소위 소프트웨어의 위기를 맞아, 개발자는 적기에 고품질의 소프트웨어를 개발하여 공급하여야 하고 사업관리자 및 사용자는 소프트웨어의 인수 전에 다양한 시험을 통하여 신뢰를 가져야 하는 어려움에 처해있다. 특히, 개발사업의 일정상 소프트웨어 시험계획이 계획대로 추진되지 않아 사업의 종료시점에 가까워질수록 병목현상이 발생되어 각종 시험을 시행하는 과정에서 많은 문제들을 내포하고 있다.

이 글에서는 우리 나라의 공공분야에서 추진하고 있는 CALS 사업을 살펴보고, 시험평가의 국제표준인 ISO 9000-3, ISO/IEC 9126, 그리고 ISO 14598 등을 고찰하였다. 또한, 미 육군의 소프

트웨어 시험평가 편람을 통하여 시험평가 절차를 살펴 보고, 국내 대형 소프트웨어 프로젝트에 적용하고 있는 시험평가절차 특히 MIL-STD-498을 근간으로 제정한 국방훈령을 검토하였다. 그리고 이러한 시험평가를 실행하는데 발굴된 문제점과 대책을 살펴보고, 시험평가 자동화 도구에 대하여 유형별로 제품의 특성을 조사하여 도구 선정시에 기능별 성능을 일목요연하게 볼 수 있도록 정리하였다.

마지막으로 우리의 실정에 맞는 소프트웨어 시험평가를 정착시키기 위하여 다양한 환경을 지원할 수 있는 시험평가모델을 설계하였다. 그러나 여기에 필수적으로 수반되어야 하는 시험평가에 대한 자동화 도구가 미비되어, 일부기능만 지원되는 McCabe Tool을 활용하여 공공분야 프로젝트에 적용실험을 실시한 결과사례를 제시하였다.

Ⅱ. 공공분야 CALS/EC 시험평가

2.1 공공분야 CALS/EC 추진 실태

최근에 국내의 공공기관이나 민간부문에서도 경쟁력 향상을 위해 CALS/EC 전략을 도입하기 위한 연구 활동과 시범 사업들이 활발히 진행되고 있다. 정보통신부, 산업자원부를 포함한 정부 각 부처에서는 대형 소프트웨어 개발사업을 추진 중에 있고, CYBER KOREA 21, 전자정부 구현, CIO제도, 전자거래 기본법, 전자 서명법 등 제도적 장치도 마련하여, 새로운 천년을 맞이하는 창조적인 지식기반의 국가건설을 위한 정보 인프라 구축과 지식정보기반을 활용한 국가전반의 생산성 향상을 위해서 매진하고 있다. 공공분야 CALS/EC중에서 EC는 전자상거래중심으로 쇼핑몰과 기업간 또는 기업과 소비자간에 활발히 추진되고 있으나, 본고에서는 이를 제외하고 CALS를 중심으로 한 공공분야 추진 실태를 논하고자 한다.

<표 1> 공공분야 CALS 추진 실태

번호	사업명	주관기관	번호	사업명	주관기관
1	대법원 부동산 등기	대법원	10	119종합방재시스템	서울시
2	출입국관리	법무부 출입국	11	BPR 및 통합정보시스템	한국도로공사
3	감사종합정보시스템	감사원	12	세입민원행정종합시스템	부산특별시
4	국민의료보험정보시스템	의료보험공단	13	국방시설정보시스템	국방부
5	의료보험종합전산망	의료보험연합회	14	국방지휘소자동화사업	국방부
6	교육원 학사행정관리	정보통신부	15	국방보급정보시스템	국방부
7	세무종합전산화	서울특별시	16	조달 EDI	조달청
8	시군구종합행정	행정자치부	17	국가안전관리시스템	행정자치부
9	한방분야 의료보험구축	보건복지부	18	체신금융분산시스템	정보통신부

<표-1>은 공공기관에서 추진하고 있는 100억원 이상의 대형사업들로서, 대법원 부동산 등기, 법무부 출입국관리업무, 감사원 감사종합정보 시스템을 비롯한 18개 프로젝트들이 정부관리 업체주도 개발형태의 외주 용역사업으로 3-5년 기간에 걸쳐 추진되고 있다.

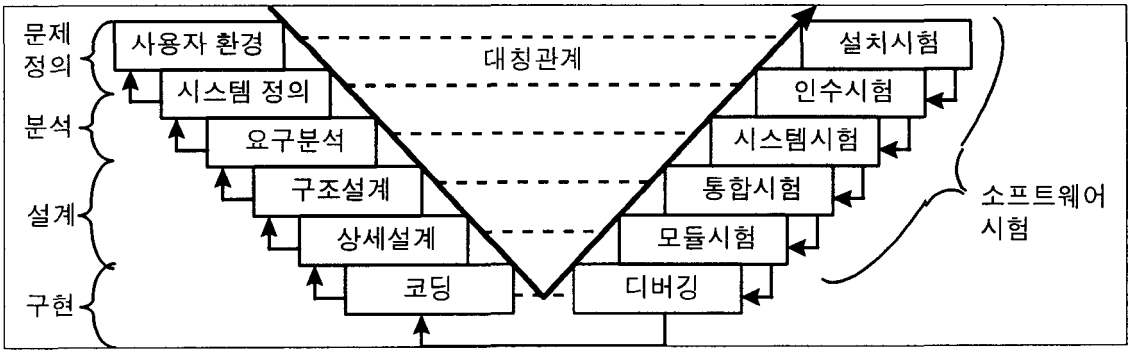
또한 국방 분야에서도 과거 몇 년 전부터 군 연구기관을 중심으로 국방CALS 구현을 위한 기초 연구 및 정책 방향에 관한 연구결과, 국방 CALS 정책 및 기본 방향을 이끌어 갈 국방CALS 사업단을 중심으로 무기체계 획득 및 군수지원 업무에 대한 기관별 혹은 단위 기능별 정보체계 구축사업이 활발히 진행 중에 있다. 예를 들어 국방군수정보체계(탄약, 보급, 장비정비), 국방시설통합정보체계, 국방조달관리 정보체계, 국방형상관리정보체계, 국방고속정보통신망 사업, 잠수함/항공기의 기술교범(IETM), 항공기 정비정보체계 등이 추진되고 있다.

2.2 소프트웨어 시험평가

2.2.1 시험평가 개요

소프트웨어 시험이란 숨어 있는 결함을 찾기 위하여 소프트웨어를 작동시키는 일련의 행위와 절차로서 소프트웨어에 결함이 있음을 보여주는 작업이라고 정의할 수 있다. 본 장에서는 국내 공공분야 소프트웨어 개발사업에 적용되고 있는 시험평가의 일반적인 방법론을 기술하고자 한다.

소프트웨어 시험에 임하는 일반적인 원칙은 시험의 독립성으로서, 개발자는 자신이 작성한 프로그램을 시험하지 않으며, 개발 조직은 그 조직에서 제작된 소프트웨어를 직접 시험하지 않도록 한다. 각 시험 사례(test case)는 기대되는 출력 내용을 포함하고 있어야 하고, 사용상 오류나 기대되지 않은 입력 조건에 관해서도 시험사례가 준비되어야 한다. 또한, 결함이 발견되지 않을 것이란 가정으로 시험 계획을 수립해서는 안되며, 각 시험의 결과는 세밀히 검토되어야 한다. 소프트웨어 개발 노력의 일반적인 분포는 분석 및 설계에 40%, 구현에 20%, 그리고 시험에는 40%를 차지하고 있어, 사용자의 만족도는 마지막의 시험을 얼마나 철저하게 했느냐에 따라 좌우되고 있다.



<그림 1> 소프트웨어 개발 및 시험 수명주기(V-Diagram)

2.2.2 시험의 유형

단계별 소프트웨어 시험의 유형은 개발시험과 운용시험으로 구분하며 개발시험은 다시 단위시험, 단위통합시험, 소프트웨어 품질시험, 소프트웨어 및 하드웨어 통합시험, 시스템품질시험으로 나눈다. 시험 목적에 의한 분류로 기능시험, 성능시험, 스트레스시험, 구조시험이 있고, 시험 방법에 의한 분류는 소프트웨어 외부에서 기능과 성능 등을 시험하는 블랙박스시험과 소프트웨어 내면의 세계를 분석하는 화이트박스시험이 있다. <그림 1>은 소프트웨어 개발 및 시험을 표현한 것이다.

<표 2>는 소프트웨어 시험 단계와 각 단계별로 수행되는 시험 내용을 요약 기술하고 있다. 단

위시험은 예를 들면 화이트박스 시험의 일종으로 개발업체에서 시행되는 초기의 시험이다. 다른 모듈과의 데이터 인터페이스를 시험하는 인터페이스 시험, 모듈 내의 자료구조상에 오류가 있는지 시험하는 자료구조 시험, 구조시험이나 루프시험을 통해 논리경로를 시험하는 수행경로 시험, 각종 오류들이 모듈에 의해 적절하게 처리되는지 시험하는 오류처리 시험, 오류가 발생하기 쉬운 경계 값들을 시험 사례로 만들어 시험하는 경계 시험 등이 여기에 속한다.

본 시험단계별 분류와 국방정보체계관리규정(훈령 561호)과는 약간 차이가 있다. 통합시험에는 단위 통합시험, 소프트웨어 품질시험, 소프트웨어/하드웨어 통합시험이 포함되며, 시스템 시험은 시스템 품질시험으로 표현되어 있고, 인수시험과 설치시험은 운용시험으로 각각 분류하고 있다.

<표 2> 시험단계와 단계별 시험내용 ( ): 국방훈령 분류

시험 단계	단위시험	통합시험 (단위통합,CSCI, CSCI/HWCI시험)	시스템시험(시스템 품질시험)		인수시험 (운용시험)	설치시험 (운용시험)
시험 내용	인터페이스시험 자료구조시험 수행경로시험 오류처리시험 경계시험	하향식통합 상향식통합 샌드위치형 통합	외부기능시험 내부기능시험 부피시험 스트레스시험 성능시험 보안시험 사용용이성시험	기억장치시험 호환성시험 설치용이성시험 신뢰성시험 복구시험 구성시험 유지보수용이성	확인시험 알파시험 베타시험	소프트웨어선택 사양 하드웨어 구성 파일분배, 적재 타체계 연결

2.2.3 시험 기법

소프트웨어 시험 기법은 시험 사례 최적화 방안으로 모든 가능한 사례 중에서 결함 발견 확률을 가장 높일 수 있는 시험 사례들을 찾는 기법으로 시간, 비용과 컴퓨터 시간 등이 소요된다.

블랙박스 시험 기법은 데이터 위주 혹은 입출력 위주 시험으로 원시코드는 보지 않은 채 목적코드를 수행시켜 가면서 결함을 발견할 수 있는 시험 사례들을 준비하며 시험에 임하는 방식으로, 부정확하거나 누락된 결함, 인터페이스 결함, 자료구조상의 결함, 성능 결함, 시작 및 종료상의 결함 등을 발견하는 기법이다.

동등분할(Equivalence partitioning) 기법은 입력 자료 값의 범위가 0 ~ 100사이인 경우, [x < 0]과 [x > 100]은 오류 유형 사례로 시험 범위(대상)를 다양한 입력 조건들을 갖춘 시험 사례의 유형(equivalence class)들로 분할하여 최소의 사례를 경험적(heuristic) 방법으로 작성하여 입력 조건에 치중하는 특징이 있다.

경계값 분석(Boundary-value Analysis)은 x 값의 범위가 0 ~ 100사이인 경우, 시험 사례 유형을 [x=0], [x=100], 및 [x=-0.01]으로 작성하여 임의의 값에 대한 경계치 전후를 시험 목적으로 작성하는 기법이다.

원인-결과 그래프(Cause-Effect Graph)는 소프트웨어의 기능·성능 요구를 다룰 수 있을 만큼의 영역으로 분할하는 기법으로 각 원인은 하나의 입력 조건이나 입력 조건의 시험 사례 유형이며, 결과는 출력 조건 혹은 소프트웨어 형상에 의하여 변환된 결과 값을 말한다. 그 다음 의사결정표(decision table)를 만들어 각 열은 하나의 시험 사례가 된다.

오류예측(Error Guessing) 기법은 블랙박스 시험 기법들이 놓칠 수 있을 만한 오류들을 감각과 경험으로 찾아보는 기법이다. 즉, 입력 값의 범위를 넘는 값(예: 나이에 음수 입력)을 입력한다던가, 문법에 어긋난 입력을 시험한다던가, 입력 값 없이 입력시켜 오류를 예측해 보는 기법이다.

화이트박스 시험 기법은 프로그램 상에 허용되는 모든 논리적 경로(logical path)를 파악하거나 경로들의 복잡도(complexity)를 계산하여 시험 사례를 작성하는 기법이다. 대표적인 방법으로 소스코드의 문장마다 통과하는 커버리지(Statement Coverage)방법, 기준선 조건 커버리지(Baseline Condition Coverage), 결정 커버리지(Decision Coverage), 조건 커버리지(Condition Coverage), 결심/조건 커버리지(Decision/Condition Coverage)마다 통과하여 시험하는 방법 등이 있다.

2.2.4 시험평가도구

사용자 요구사항의 다양화와 복잡화, 그리고 시간의 제약으로 다양한 플랫폼의 지원으로 시험하기가 더욱 복잡해지고 있다. 따라서 시험 수행 시간의 단축과 효율적인 시험을 위하여 단순한 디버거에서 벗어난, 다양한 시험도구의 적용 필요성이 대두되고 있다. 이때 시험 수행에 필요한 인력과 장비, 그리고 시간의 제약성을 고려하여 대상 산출물의 환경에 적합한 시험 기법과 도구의 선택이 중요하다.

<표 3> 시험도구 유형과 시험도구명

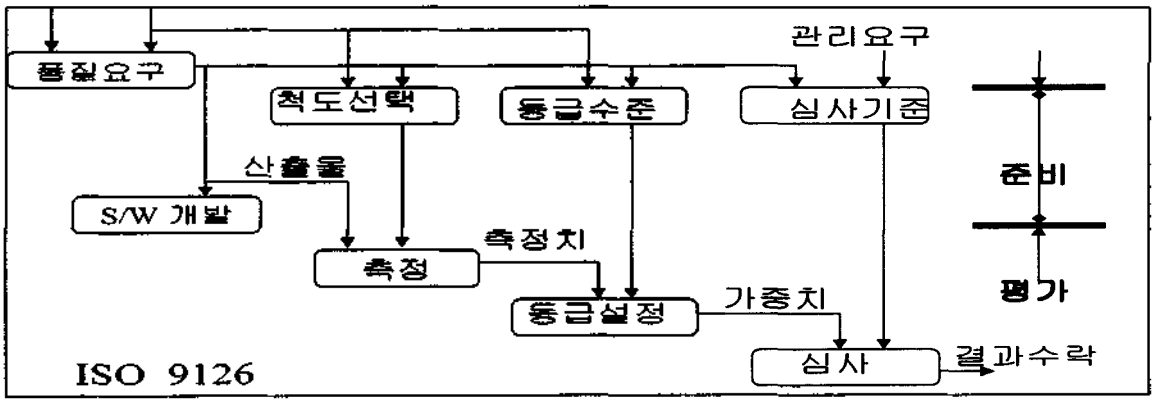
유형	도구명	비고
Capture/ Replay tools	ATF, ATS/X-Tester, FERRET, CAPBAK/X, CAPBAK/MSW, avaStar, QA Partner, QA Planner, QARun, Smalltalk test Mentor, SQA Suite, TALC 2000, TestRunner, VALIDOR, Visual Test, WinRunne, rXRunner	클라이언트/서버, Tcl, X와 윈도우에 대한 캡쳐/재생 시스템, 자바 도구, GUI 객체지향,C/S, Smalltalk와 VisualAge Generator, Windows client/server, POS, Validor Pro와 S, X 응용의 GUI 캡쳐/재생
Communications Testing Tools	Chariot, Chisel, FastBench Agent Tester, ITF, SDTF, SNMP Test Suite	네트워크 성능, 응용과 서비스, TMN, TCP/IP, SNA, 프로토콜 일치성, 에러/예외 조치,성능 검증
Source Code Testing Tools	AdaTEST,ATTOL Coverage, Bounds Checker, C-Cover, Cantata, Code Check, CodeIntegirty, CodeWizard, DeepCover, Hindsight, Insure++, Java Scope, LDRA Testbed, Logiscope, OSPC, Panorama, PureCoverage, Purify, TCAT	Ada 정적/동적 테스트, 코드, 메모리리크(누출) 감지, C/C++ 코드 ,C 소스코드, C++ 분석, C/C++, 자바, 정적/동적 코드, 런타임 에러와 메모리리스크(누출) 감지, C, C++, 자바용 코드 커버리지 분석
Test Management Tools	ATTOL System Test, PRS, QADirector. QAPlan.QATrack, QES/Architect, TCS, Qualit Defect manager,SMARTS, Test Director, Test Expert, TestMaster, Test Mate for Ada, TestTrak, TestWorks, VisionSoft/PROJECT	분산 시스템, 문제식별보고, 테스트 프로세스, 품질 프로세스, 결함 추적, 소프트웨어 관리, 테스트 관리, Mac을 위한 결함추적, UNIX, Windows, NT용 소프트웨어 통합 도구
Web Testing Tools	HTML validation tools, Web site tools and site management tools,Test browser , Auto Tester Web, Bobby, CGI Tester, CYRANO Web Tester, HTML Validator, InfoLink Link Checker, NetMechanic, preVue-Web, e-TESTER, Silk ,SQA SiteCheck, STW/Web,	대규모 인터넷, 인트라넷 애플리케이션용, 웹페이지 접속, 웹 기반 애플리케이션에 대한 리그레션, HTML DTD, 링크가 끊어진 곳을 식별, 서버의 응답시간 측정, 로드테스팅, 성능테스팅, 기록/재생 도구, 전사적 수준의 웹, 에러 감지 및 수리
Performance Testing Tools	Astra SiteTest, Bang, Benchmark Factory, CYRANO, e-LOAD, Final Exam Web Load, WebLoad, JavaLoad, QALoad, LoadRunner, LoadTester, PreVue-C/S,, Silk Performer,QAStress, Web Raider	전사적 규모의 웹 사이트와 웹 기반, 복수의 벤치마크 개발, 윈도우즈 애플리케이션, 인터넷, 인트라넷 응용체계, 윈도우즈 NT/95용 웹사이트, HTTP 애플리케이션
Other Tools	ATTOL UniTest, CodeTEST, DBCL, Dr. DeeBee, Error Projection Tool, TestBase, VisionSoft, McCabe & Associates,	C, C++, Ada, CSQL로 된 내장시스템 또는 mission-critical 시스템,ODBC, 내장 시스템, 런타임 에러, 리버스 엔지니어링 도구,

<표 3>은 시험도구들을 분석한 내용이다. 소프트웨어 시험평가 도구는 해당 소프트웨어의 기능에 따라서 평가 방법이 다양하다. 국내에는 아직 시험평가 도구가 개발되어 운영중인 것이 거의 없으며 외국에서는 다양한 도구가 개발되어 운영되고 있다. 그러나, 각각의 기능별로 성능이 다양하여 주로 소프트웨어 개발 프로젝트의 기획과 소요 분석보다는 코드의 분석, 시험 수준, 모의 시험, 부하 시험 등 개발 수명주기의 후반부에 치중되어 있는 실정이다. 일반적인 시험 도구로는 Static Analyzer, Code Editors, Assertion Processor, Test File Generation, Test Data Generation, Test Verifier Generation 등이 있으나, 최근에는 인터넷과 관련된 하이퍼미디어 자료를 시험 평가할 수 있는 다양한 도구들이 출시되고 있다. 즉, 캡쳐 앤 리플레이(Capture/Replay) 도구, 통신 관련 도구, 성능 관련 도구, 소스코드 관련 도구, 시험관리 도구 그리고 웹 시험 관련 도구들이 30여 개가 출시되고 있다.

Ⅲ. 소프트웨어 시험 평가 표준과 문제점

3.1 시험 관련 표준

소프트웨어 시험 평가에 대한 중요성을 인식한 미 국방부, NATO, NASA 등에서는 시험 평가를 자체적으로 개발하여 활용하여 오던 중에 1991년 ISO 9000-3 지침이 공포되어 22개의 항목을 제시하여 인증심사를 하고 있다. 그러나, 품 질향상을 위한 품질시스템의 인증 이후에 품질 개선을 위한 계량적인 측정이 필요하게 되어 ISO에서는 1991년 ISO/IEC 9126 표준(<그림 2>)을 발표하였으나, 사용의 어려움으로 인하여 JTC1/SC7/WG6에서 이를 개정 검토 중에 있다.



<그림 2> ISO 9126

현재 ISO 9126 시리즈로 3개의 표준과 ISO 14598 시리즈로 6개의 표준이 개발되고 있다.

-ISO 9126-1: 소프트웨어 품질 특성 및 척도- 주/부 품질 특성

-ISO 9126-2: 소프트웨어 품질 특성 및 척도- 개발자중심의 외부척도

-ISO 9126-3: 소프트웨어 품질 특성 및 척도- 구매자중심의 내부 척도

-ISO 14598-1: 소프트웨어 제품의 평가- 일반사항
-ISO 14598-2: 소프트웨어 제품의 평가- 기획 및 관리
-ISO 14598-3: 소프트웨어 제품의 평가- 개발자를 위한 프로세스
-ISO 14598-4: 소프트웨어 제품의 평가- 구매자를 위한 프로세스
-ISO 14598-5: 소프트웨어 제품의 평가- 평가자를 위한 프로세스
-ISO 14598-6: 소프트웨어 제품의 평가- 평가모듈

3.2 미 육군의 시험평가 개요

미 육군은 미국 국방부 소프트웨어 개발 및 문서화 표준(MIL-STD-498)에 의해 시험 평가 지침을 작성하여 활용하고 있다. 총 10장으로 구성되어 있으며, 시험 평가 개요, 획득과 개발 분야에서의 시험 평가, 시험평가팀의 편성, 예비 시험 평가 활동, 시험 활동, 소프트웨어 계측 등으로 구성되어 있다. 시험 평가의 특성은 지속적인 평가(CE-Continuos Evaluation)로 소프트웨어 개발 수명주기 동안에 걸쳐서 수행된다. 이러한 지속적인 평가는 소프트웨어 품질보증, 소프트웨어 구성관리, 독립적인 확인과 검증(IV&V-Independent Verification & Validation), 정부 주도 개발과 운용 평가, 그리고 소프트웨어 개발자를 포함한 인력에 의해서 수행된다.

소프트웨어 시험 수준은 개발자에 의해서 사용자의 요구사항을 수행되는 소프트웨어 시험, 시스템 수준에서 소프트웨어 단위 시스템을 통합하여 시험하는 소프트웨어/시스템 시험, 권위 있는 자에 의해서 정부 주도 시험대에서 수행되는 시스템 기술시험, 사용자의 효과성과 적합성을 고려한 시스템 수준의 시스템 운용시험으로 구분한다.

시스템의 시험은 개발자에 의해서 초기에 수행되는 것으로 사용자의 요구를 만족하는 설계 및 구현되었는지 확인하는 소프트웨어 시험, 소프트웨어 요구사항을 개발자의 환경에서 시험하는 소프트웨어/시스템 시험, 정부 주도 시설에서 기술적인 시스템 수준의 요구사항의 만족도를 시험하는 시스템 기술시험, 사용자의 효과성과 적합성에 초점을 맞춘 시스템 수준의 운용시험이 수행된다.

정적인 분석 기술은 에러를 찾아내기 위하여 시험으로 코드를 실행하기보다는 소프트웨어의 제품의 분석과 실험을 포함하여 코드 편집 상태, 연동 검사, 검토회, 검사회의 자료 검토, 단위 시험, 자료 흐름 분석, 구조 분석, 상호참조 분석, 입력 여백 상태 기술, 복잡성 분석 등이 이에 속한다. 동적인 시험은 확인 시험, 결과-효과 그래프, 성능 측정 기술, 경로 구조 분석, 내부 디버깅, 기능 시험, 분산 시험, 실시간 시험, 에러 찾기 시험, 변화 시험 등이 이에 속한다.

시험 평가 준비 및 요원 구성은 소프트웨어 시험 계획이 작성되는 시점에서 독자적인 평가 요원이 구성되어 다음과 같은 시험을 추진한다. 평가 자료의 상호 공유, 조기에 결함을 식별하고, 시험 분석 전략에 의해서 대안을 제시하며, 개발 책임자와 의사결정자에게 통보 가능한 위치에서 임무를 수행한다. 시험평가 요원의 구성은 개발자, 사용자 대표, 프로젝트 관리자, 개발 시험자, 평가자, 운용시험자, 운용평가자 등으로 구성한다. 시험 계획은 시험평가 종합계획(최소 수용 가능한 성능 요구, 치명적인 기술 요소, 치명적인 운용 요인), 소프트웨어 시험 계획(환경, 일정, 수행절차), 시험 설계 계획, 운용시험 평가 계획 등을 사전에 준비한다.

시험 활동은 단위 시험, 단위통합 시험, CSCI 품질시험, CSCI/HWCI 통합시험, 시스템 품질시험, 시스템 개발시험, 시스템 운용시험이 실시된다. 개발 평가 점검 요소는 성능(시스템 반응 시간, 정확도, 회복/재작동 절차, 전환 프로세스, 반복성, 격심한 운용 절차), 연동성(송신 확인, 송신,

우선순위, 스트레스, 이해성, 연동성 고려), 사용성(사용자 상호작용 시스템 반응, 출력 품질, 훈련), 정비성(문서화 품질, 코드의 품질, 컴퓨터 자원), 안전성(입력 절차, 자발성), 그리고 보안성(접근 방지, 부인 방지, 계정 관리)으로 구성된다.

3.3 시험평가 문제점 및 대책

3.3.1 문제점

소프트웨어 시험평가상의 문제점은 제도적인 측면, 관리적인 측면, 그리고 기술적인 측면에서 도출할 수 있다. 제도적인 측면에서 보면 첫째, 시험평가에 대한 국제 표준의 준수가 안되고 있다. 둘째, 소프트웨어 시험평가에 대한 국내 개발업체의 제도 및 규정이 미흡하다.

관리적인 측면에서 첫째, 시험 기간의 장기 소요 및 불안정한 시험 수행으로 인한 납기 지연 초래이다. 시험 수행 계획에 의한 일정보다 항상 시험 마감 기간에 임박하여 병목현상이 발생되어 충분한 시험을 수행하지 못하여 주어진 일정에 차질을 초래하는 사례가 빈번하고, 무상 유지보수기간(통상 납기 후 1년)까지 연장하여 수행하는 사례가 많다. 둘째, 고품질의 소프트웨어 확보에 차질을 초래한다. 셋째, 사용자의 다양한 요구를 시험 사례별로 작성하여 동시 시험이 불가능하여 비효율적이다. 넷째, 재래식 기법으로 소프트웨어 개발 생산성 향상에 기여하지 못한다. 다섯째, 시험 기준에 만족한 충분한 시험과 효율성 및 신뢰성 보장이 곤란하다. 여섯째, 하나의 시험 항목에 대하여 다수의 결함 정보를 결함 순기에 따라 체계적으로 관리하지 못하고 있다. 또한 발견된 결함에 대한 다양한 통계 처리가 안되고 있다. 일곱째, 시험 수행 계획과 실제 수행 집행의 차이가 발생(인력, 조직, 장비, 기법, 시험방법론, 범위, 일정 등)한다. 여덟째, 대규모의 프로젝트에 개발자와 시험자가 동시에 시험 사례, 스크립트를 포함한 시험 정보를 공유하여 시험 결과를 자동 저장할 수 있는 저장소가 없다.

기술적인 측면에서 첫째, 육안 식별의 한계로 인하여 대용량의 이미지 변환, 전송된 파일간의 동일시에 대한 자동 검증이 불가하다. 둘째, 작성된 스크립트를 자동으로 재수행(Replay)하여 실제 수행 결과와 예상치를 비교할 수가 없다. 셋째, 시험된 경로와 시험되지 않는 경로의 미표시 및 프로그램간의 차이를 자동으로 비교하지 못한다. 넷째, 오류 수정에 따른 버전 변경이 있는 경우 이전의 스크립트를 재수행하므로서 새로운 시스템의 영향 범위 및 정확성 확인이 곤란하다. 다섯째, 시험 항목과 절차를 포함한 시험 계획에서부터 시험 결과의 분석에 이르기까지 전 과정의 저장 관리가 되지 못한다. 여섯째, 인터넷 환경의 웹 기반의 시험 도구 및 객체 지향 등의 다양한 환경을 지원할 수 있는 시험 도구가 없다.

3.3.2 대책

위와 같은 시험평가에 대한 문제점을 해결하기 위한 여러 가지 대책을 요약하면 다음과 같다.

1) 개발자, 사용자 환경에 적합한 시험평가 모델 개발
2) 다양한 환경에 적합한 자동화 시험 도구의 도입 및 시험방법론의 채택
3) 개발 조직과 독립된 별도의, 소프트웨어 품질과 시험활동을 위한 전문 조직의 구성 및 활용
4) 시험 목표, 시험 기법, 시험기준 설정, 시험 사례와 시나리오 등 시험 계획의 사전 작성 준비
5) 시험평가에 대한 국제 표준의 준수와 제반 제도 및 규정의 보완
6) 시험 기법, 오류 정보, 시험 스크립트 등의 정보를 저장할 정보저장소 유지
7) 개발 초기 단계에서부터 계획하여 반복적으로 자주 수행하여 효과적인 시험 수행

8) 구체적인 시험데이터를 입력하고 그 결과를 분석하여 체계적으로 관리 조치

9) 자동화 도구의 사용으로 시험 설계 단축, 사용의 편리성, 만들어진 정보의 재사용성

10) 여러 사람이 사용하여도 그 결과는 동일하여 신뢰성이 보장되도록 시험

11) 최소의 노력으로 최대의 효과를 거두도록 효율성 제고

12) 소스코드의 경로, 문장, 조건, 의사결정 등의 폭 넓은 시험 커버리지(Coverage)

13) 다양한 결과 분석을 통하여 각종 도표 또는 그래프로 가시적인 시험 결과 파악

이러한 대책들을 고려하여 다음 장에서 다양한 환경에서 적용할 수 있는 시험평가모델을 제시한다.

Ⅳ. 소프트웨어 시험평가 모델 설계

4.1 소프트웨어 시험평가 모델 요구사항

공공분야의 CALS환경에서 소프트웨어 개발시에 활용되는 시험평가의 중요성은 아무리 강조하여도 지나치지 않다. 일부 공공분야의 소프트웨어 개발 사업에 간단한 디버거와 같은 시험 도구를 활용하고 있거나, 아니면 일부 기능별 점검표에 의하여 육안 검사로 시험평가를 추진하고 있는 실정이다. 따라서 다양한 환경에서 적용 가능한 자동화 시험 도구를 활용한 새로운 시험평가 모델이 요구되고 있다. 이에 대한 요구사항을 종합하여 보면 다음과 같다.

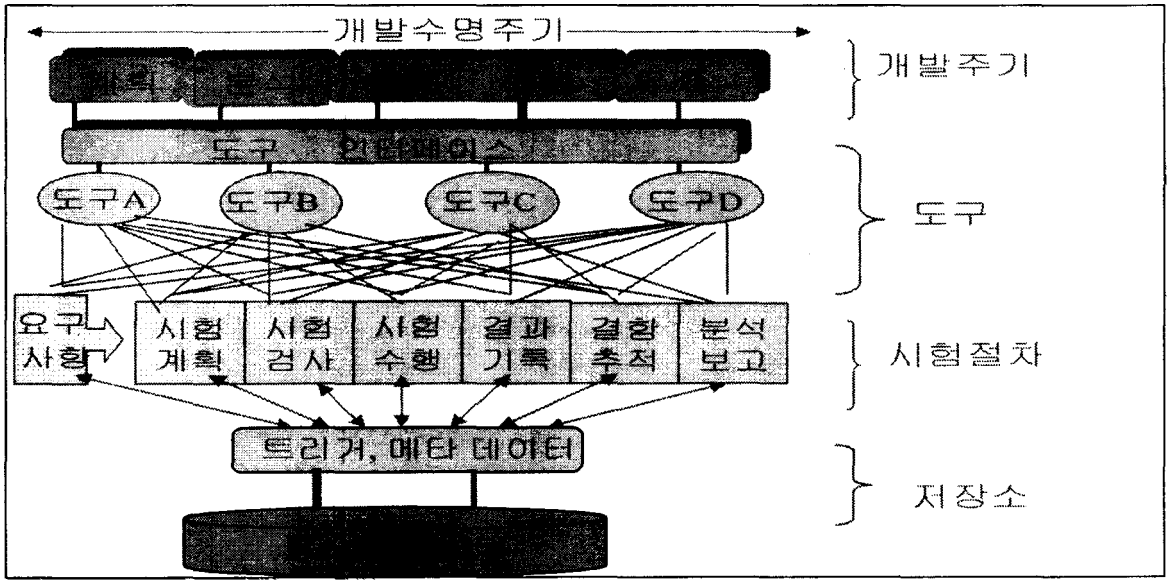
첫째, 다양한 소스코드의 분석, 시험 관리 도구, 캡쳐 앤 리플레이(Capture and Replay), 웹 관리 기능, 통신 기능 등이 통합적으로 관리되어 개발자나 시험평가자에게 필요한 정보를 제공할 수 있도록 구성한다. 둘째, 윈도우 환경에서 웹 브라우저를 통하여 자바를 비롯한 객체지향 차세대 언어로 작성된 소스코드에 쉽게 접근이 가능하도록 구성한다. 셋째, 소프트웨어 개발 전체의 수명주기를 지원하는 대형의 정보 저장소(Repository)를 공유하여 시험평가 계획, 절차, 일정, 그리고 결과 자료를 통합 운영한다. 넷째, 시험경로의 선정, 사용자의 도움 없이 입력과 수행 내역을 캡쳐하고 재생, 시스템의 부하 태스크의 수행 등 시험의 완전성을 보장하는 커버리지 측정 기능을 제공해야 한다. 다섯째, 모듈의 복잡도와 코딩의 준수 지침 등을 포함한 원시 프로그램의 제어 흐름과 데이터의 흐름을 분석하여 시험 용이성과 유지보수성, 그리고 통신망 성능의 시험을 평가 제공해야 한다. 여섯째, 개방형 구조를 중심으로 한 네트워크 플랫폼에 의존하지 않는 시스템, 데이터베이스 플랫폼에 의존하지 않는 시스템, 어떠한 시스템과도 연동이 자유로워 확장성과 유연성이 보장되는 인터페이스 구조로 된 시스템이어야 한다.

4.2 소프트웨어 시험평가 모델 설계

새로운 소프트웨어 시험평가 모델 설계는 <그림 3>에서 보는 바와 같이 위의 모델 요구사항을 고려하여 시험평가의 수명주기동안에 발생되는 모든 일을 자동화 도구를 사용하여 통합적으로 관리하는 방향으로 설계하였다. 즉, 사용자의 요구사항을 반영한 시험 계획을 작성하고 시나리오에 의해서 검사하고 각종 기법을 도입하여 시험을 수행하고 그 결과를 기록하고 결함 사항을 추적하여 결과 보고 하는 일련의 절차를 정보저장소에 관리 유지한다. 또한 수작업으로 점검표에 의하여 시나리오 작성을 하던 재래식 절차를 자동화 도구를 활용하여 일관성 있게 추적한다. 여기에

서 자동화 도구는 캡쳐 앤 리플레이(Capture/Replay) 도구, 통신 관련 도구, 성능 관련 도구, 소스 코드 관련 도구, 시험 관리 도구, 그리고 웹 시험 관련 도구 등을 포함하여 각각의 도구가 상호 연동성을 가지도록 인터페이스시켜야 한다.

시험 계획 단계에서는 사용자의 요구사항을 반영하여 시험의 목표, 범위, 시험 환경, 시험 식별 및 수준, 일정과 시험 조직 및 편성, 그리고 요구사항의 추적성이 포함되어 있으며, 시험 검사의 단계에서는 시험사례와 시나리오, 시험 수준이 자동화 도구에 의해서 작성된다. 시험 수행 단계에서는 적용 시험 기법에 의하여 각 모듈별로 순환복잡도(Cyclomatic Complexity)에 의하여 분기의 다수 여부에 따라서 결함 발생 확률을 예측하고, 핵심복잡도(Essential Complexity)에 의하여 정형화 척도를 측정하여, 통합순환복잡도(Integration Cyclomatic Complexity)에 의하여 모듈간의 인터페이스를 측정하여 복잡도와의 상관 관계로 시험 대상을 선정한다.



<그림 3> 시험평가 모델 설계

또한 시험 결과의 기록과 결함 추적 단계에서는 원시 프로그램의 복잡도, 제어 흐름, 데이터의 흐름, 경로 커버리지의 측정, 결함의 발견과 추적으로 시험의 용이성과 변경의 용이성으로 향후 유지보수를 지원한다. 여기에서 대형 정보저장소(Repository)에는 상기의 시험 절차, 각종 도구간의 관계 자료를 상호간에 공유할 수 있도록 보관 관리하고, 문서의 색인기능, 검색기능, 통신기능, 웹 기능, 저장관리자기능을 지원한다. 물론, 정보저장소와 이러한 시험 절차 사이에는 각종 절차를 조정하는 트리거 기능과 메타 데이터 기능을 부여하여 원활한 정보의 교환이 되도록 한다.

이러한 일련의 시험 절차는 자동화 도구에 의하여 수행된다. 소프트웨어의 구조와 제어 흐름 생성을 통해 복잡도와 시험 경로를 생성하는 정적인 시험 기능, 각종 프로그램에 대한 분석 기능, 블랙박스 시험에 의하여 시험되지 않은 경로를 제시하고 프로그램간의 비교 분석을 해주는 동적인 시험 기능 등을 포함하고 있으며, ISO 9126의 척도 선택에 의한 측정과 평가 등급에 의한 평가로 인수 여부를 결정짓는 절차를 따른다.

소프트웨어 시험평가 모델은 다양한 시험 도구, 시험 도구 인터페이스, 개발 지원 서버, 시험 지원 서버, 보안 통제 서버, 대형 정보저장소, 저장 관리자, 검색 관리자, 네트워크 관리자, 웹과

브라우저 등으로 이루어져 있으며, 다양한 시험의 지원을 위해서 개방성과 확장성을 고려한 윈도우 환경의 3계층(3-Tiered)과 웹 기반의 지식 정보 시스템으로 구성되어 있다.

4.3 시험평가 모델 적용 사례

설계된 시험평가 모델을 적용하기 위하여 대상 업무를 조사한 결과, 국방보급정보시스템이 현재 대규모의 사업으로 객체지향 방법론을 적용하여 정부관리 업체주도 개발방식으로 개발되고 있으며 MIL-STD-498을 기반으로 한 국방정보체계 관리규정에 따라 사업관리가 수행되고 있다.

<표 4> 보급정보체계 구축사업의 시험단계

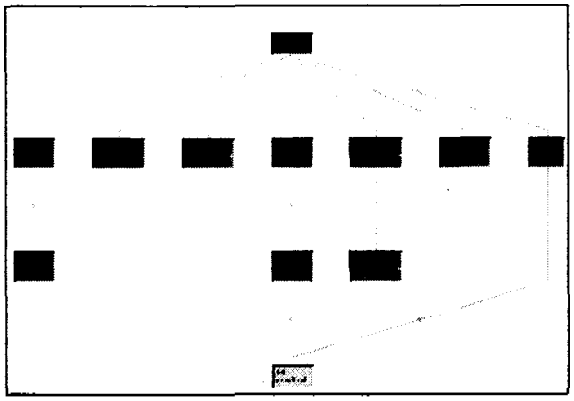
시험단계명		활동내용	주관	지원
개발시험	단위시험	단위수준의 개발자 자체시험이며, 획득자는 S/W 개발과일(SDF)을 검토.	개발자	획득자
	단위통합시험	단위들을 통합시키고, 수행하는 개발자 자체시험이며, 획득자는 S/W개발파일(SDF)을 검토.	개발자	획득자
	응용S/W구성요소 품질시험	개발된 응용S/W의 품질을 측정하는 시험으로, 획득자가 개발자의 개발환경에서 수행.	획득자	사용자 개발자
	응용S/W구성요소/H/W구성요소 통합시험	응용S/W구성요소 품질시험에서 품질이 만족스러운 응용S/W가 개발되었다고 판단되면, 개발자는 정부 및 군에서 제공한 하드웨어와 상용소프트웨어에 개발된 응용소프트웨어를 통합시키고 시험을 수행.	개발자	획득자
	시스템 품질시험	획득자 및 사용자의 운용환경에서 개발된 응용소프트웨어가 이상 없이 동작되는 지와 제시한 요구사항이 만족되는지를 시험.	획득자	사용자 개발자
운용시험		사용자의 운용환경에서 사용자의 수락여부와 사용가능여부를 판단하는 시험. 사용자가 제시한 요구사항의 만족도를 측정하여 사용여부를 결정.	사용자	획득자 개발자

본 사업의 시험은 5단계의 개발시험과 사용자 운용시험을 포함하여 총 6단계로 추진 중에 있으며, 이를 요약하면 <표 4>와 같다. 1차 단위시험은 재래식 방법으로 시나리오를 사전에 작성하여 1,184개의 모듈에 대하여 시행한 결과, 정상이 765개(64%), 기능불량이 176개(15%), 화면의 수정필요성이 141개(2%), 그리고 시스템의 수정요구가 102개(9%)로 <표 5>와 같이 측정되었다.

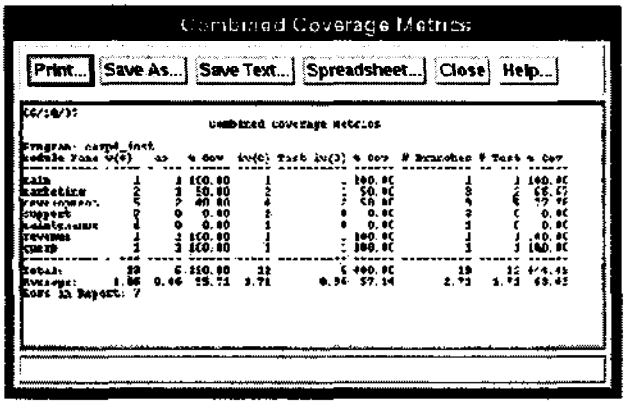
<표 5> 국방보급정보시스템 1차 단위모듈 시험 결과

구분	전 체 모듈수	검 토 모듈수	검토결과			
			정상	기능불량	화면수정	시스템수정
육군	582	372	248	57	42	25
해군	476	440	279	62	57	42
공군	477	372	238	57	42	35
계	1535	1184(100%)	765(64%)	176(15%)	141(12%)	102(9%)

본 사업에 이번에 설계된 시험평가 모델을 적용하려고 시도하였으나, 시험 환경이 미비되어 부분적으로 적용하였다. 즉, 다양한 도구가 아닌 단일 도구를 사용하였기 때문에 전반적인 인터페이스를 관리하지 못했다. McCabe Visual Toolset를 사용한 결과 소스코드에 대한 메트릭스 산출 및 품질을 도표로 가시화하였고, 수만 라인의 소스코드를 역이용(Reverse)하여 구조도(Battlemap)를 보여 줄 수 있었다. <그림 4>는 시험의 수행 여부에 따라서 다양한 표시와 색상으로 시험 진도를 표시하고 있다.

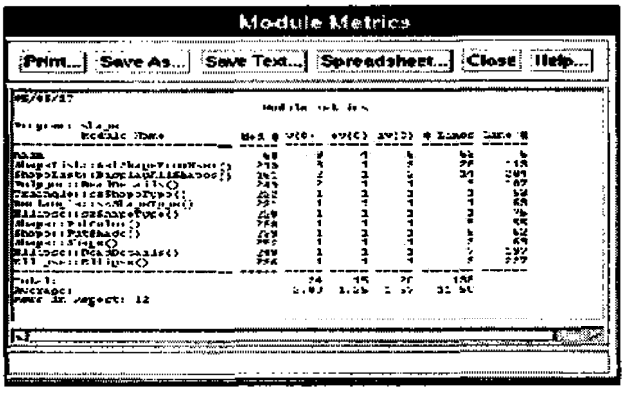


<그림 4> McCabe 도구의 구조도

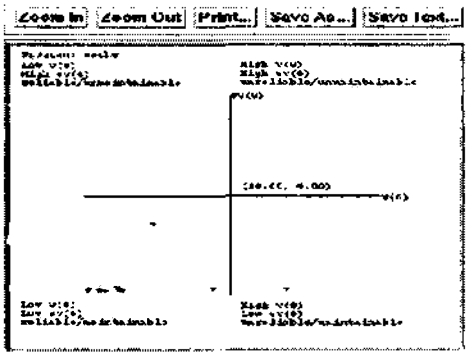


<그림 5> 결합 모듈 매트릭스

<그림 5>와 <그림 6>은 모듈별로 경로의 수량과 복잡도(V(G), E(G), I(G) -- Cyclomatic / Essential Complexity)를 나타내어 결함 발생 확률, 구조화 정도, 모듈간의 인터페이스 정도를 파악하여 중복성을 식별하고 변경에 따른 영향도를 예측할 수 있다. 또한 <그림 7>은 분포(Scatter) 도표로서 복잡도, 구조화, 모듈의 호출 복잡도가 어느 상한에 위치하는가에 따라 시험을 철저하게 할수있다. 기타 기비아트(Kiviat) 그라프는 시험성, 규모, 정비성, 라인당 평균 복잡도 등을 방사형으로 표시하여 문제가 예상되는 부분을 예측할 수 있다.



<그림 6> 모듈 매트릭스



<그림 7> 분포 도표

향후 다양한 자동화 도구를 구매 또는 개발하여 본 시험평가 모델에 연동시켜 시험 기간의 단축, 시험 비용의 절감, 그리고 고품질의 생산성 향상으로 성공적인 소프트웨어 개발 사업에 크게 이바지 할 것이다.

V. 맺는 말

최근 CALS/EC가 공공 분야에서도 활성화되어, 업체 개발 형태인 외주 용역으로 추진되고 있다. 이러한 대형 프로젝트에 시험평가의 중요성을 감안하여 국내외적으로 수행되고 있는 시험평가의 절차를 살펴보고, 국내 소프트웨어 개발시 시험평가에 내재되어 있는 문제점과 대책을 검토하였다. 또한 국외에서 개발되어 시판되고 있는 시험 도구들과 기능별로 비교 평가한 자료를 조사하여 시험 도구 선정시에 참고할 수 있도록 하였다.

소프트웨어 시험평가 모델은 다양한 시험 도구들을 수용할 수 있는 인터페이스, 사용자 요구사항을 고려한 시험 계획에서 결과 보고에 이르기까지 시험 절차들을 저장 및 관리하는 대형 정보 저장소, 검색 관리자, 네트워크 관리자, 웹 브라우저로 구성되도록 설계하였다. 이러한 절차에 의하여 설계한 시험평가 모델을 검증하고자 공공 프로젝트에 적용하여 보았다. 그러나 시험 도구가 구비되지 않아 다소 미흡한 시험이었지만, 오류를 식별하는 데 중복성을 배제하고 체계적인 시험의 효과를 거두었다. 따라서 본 시험평가 모델은 향후 시험평가에 필요한 다양한 자동화 도구를 획득하여 도구간을 인터페이스시키고, 시험평가 절차를 대형 정보 저장소와 통합적으로 관리함으로써 시험 기간의 단축과 생산성의 향상, 그리고 고품질의 소프트웨어를 획득하는데 크게 이바지할 것이다.

참 고 문 헌

[1] 임만택,"CALS/EDI 개념과 발전방향," 한국정보통신진흥협회, 1994.

[2] 임만택,"CALS환경에서의 소프트웨어 개발," 한국 CALS/EC학회, 1997.

[3] 이주헌, 실용 소프트웨어 공학, 법영사, 1996.

[4] 전호원, 김종윤, "소프트웨어 계량적 품질평가," 소프웨어 공학연구회, 1997.

[5] 손세창외 1명, "ISO/IEC 9126 품질모델 분석," 소프트웨어 공학연구회, 1997.6.

[6] 정원철, "소프트웨어 품질향상을 위한 자동화 시험도구 적용효과분석," 소프트웨어, 1997.

[7] 국방부, "국방정보체계관리규정", 국방부훈령 제561호, 1997.

[8] 국방부, "국방고위정보화 책임관(CIO) 운영규정", 국방부훈령 제620호., 1999.

[9] 국방정보체계연구소, "국방통합정보체계구축사업 및 연구", 1997.

[10] DOD, MIL-STD-498: "Software Development and Documentation", 1996.

[11] U.S. Army, Army Pamphlet 73-7: "Software Test and Evaluation Guide", 1996.

[12] Rechard A DeMillo, W.Michael McCracken, R.J.Martin, John F. Passafiume, "Software Test and Evaluation, Georgia Institute of Technology", The Benjamin Cumming Publishing Company, 1997.

[13] OVUM, "OVUM Evaluation: Software Testing Tools," http://www.info-edge.com/

[14] McCabe Associates, "McCabe Visual Toolset," http://www.genesis.co.kr/

[15] ISO/IEC 9126, "Information technology- Software product evaluation", 1991.

[16] Ap Test, "Software Testing Resources," http://www.aptest.com/