論文 J. of The Korean Society for Aeronautical and Space Sciences 45(10), 892-897(2017)
DOI:https://doi.org/10.5139/JKSAS.2017.45.10.892
ISSN 1225-1348(print), 2287-6871(online)

무기체계 SW 동적시험 회귀시험 자동화 프로그램 개발

차상철*, 김정열**

Development of the program automating regression test of dynamic test of weapon system software

Sang-Cheol Cha* and Jeong-Yeol Kim**
LIG NEX1*,**

ABSTRACT

As the weapon system SW development and management manual of the DAPA, which is the regulation for the overall weapon system SW development, is revised, the level and scope of SW reliability test are upgraded to improve the reliability and quality of SW. It is a big burden for SW developers. In particular, the dynamic test requires a schedule and manpower required to implement the weapon system SW, and should be performed every time the source code changes, not just one time. In this paper, we propose a regression test automation program(VectorCast Environment Manager) that performs a dynamic test using VectorCast, a dynamic test tool, and then performs a regression test automatically by minimizing human intervention in the regression test of dynamic test due to the change of the source code.

초 록

무기체계 SW 개발 전반에 대한 규정인 방위사업청의 무기체계 SW 개발 및 관리 매뉴얼이 개정될 때마다 SW 신뢰성 및 품질향상을 위해 SW 신뢰성 시험의 수준과 범위를 상향하는 방향으로 변경됨에 따라 SW 신뢰성 시험은 SW 개발자에게 큰 부담이 되고 있다. 특히 동적시험은 무기체계 SW를 구현하는데 필요한 일정과 인력에 육박할 정도의 비용이 소요되고 있으며, 1회성으로 그치지 않고 소스코드 형상이 변경될 때마다 수행해야 한다. 본 논문에서는 동적시험 도구인 VectorCast를 이용하여 동적시험을 최초 1회 수행한 후 소스코드 형상변경으로 인한 동적시험 회귀시험 시 사람의 개입을 최소화하여 자동으로 회귀시험을 수행하고 결과 보고서를 생성해주는 회귀시험 자동화 프로그램인 VectorCast Environment Manager 개발에 대해서 기술한다.

Key Words : SW Reliability Test(SW 신뢰성 시험), Dynamic Test(동적시험), Regression Test(회귀시험)

Ⅰ. 서 론

과거에 비해 무기체계에서 SW가 차지하는 비중이 비약적으로 증가하면서 무기체계 SW의 품질과 신뢰성에 대한 관심과 중요성이 부각되고 있다. 무기체계 SW 결함은 직간접적으로 인명

† Received : August 19, 2017 Revised : September 19, 2017 Accepted : September 21, 2017
* Corresponding author, E-mail : sangcheol.cha@lignex1.com

피해를 유발할 수 있기 때문에 무기체계 SW의 신뢰성 확보를 위한 논의가 계속되었고, 그 결과 무기체계 SW 개발 전반에 대한 규정을 기술하는 문서에서 방위사업청과 유관기관의 협의를 통해 수립된 SW 신뢰성 시험에 대한 범위와 수준에 대해 규정하고 있다.

SW 신뢰성 시험은 정적시험과 동적시험으로 나눠지며 개발 및 시험기간 중 많은 인력과 일정이 투입되는 시험은 동적시험이다. 동적시험은 시험대상 SW의 모든 함수에 대한 코드 실행률 충족 여부를 검사하기 위해 동적시험 자동화 도구를 이용하여 함수의 제어 흐름에 따른 모든 테스트 케이스를 작성하고 코드 실행률을 검사한다. 작성된 테스트 케이스가 코드 실행률 검사 조건을 충족시키지 못하거나 함수에 Dead Code가 포함된 경우 테스트 케이스를 수정하거나 소스코드를 수정한 후 다시 코드 실행률을 검사해야 한다. 만약 동적시험이 완료된 후 또는 양산 단계에서 소스코드 형상이 변경될 경우 변경된 소스코드에 대해서도 동일한 절차를 거쳐 코드 실행률 검사를 하는 회귀시험이 요구된다. 이 과정에서 동적시험 자동화 도구의 기본적인 테스트 케이스 작성이나 기본 제어 흐름 검색 기능 등으로 도움을 받을 수 있지만 개발자가 많은 시간을 예외적인 테스트 케이스를 작성하고 Dead Code를 없애기 위해 소스코드를 수정하는 등의 작업을 해야 한다. 본 논문에서는 앞서 기술한 동적시험 절차 중 개발자의 직접적인 개입이 필요한 테스트 케이스 작성 및 수정을 제외한 전 과정을 자동화해서 동적시험을 위한 인력 투입을 최소화할 수 있는 동적시험 회귀시험 자동화 프로그램에 대해서 제안한다.

II. 본 론

2.1 무기체계 SW 신뢰성 시험

2011년에 제정된 무기체계 SW 개발 및 관리 지침에서 처음으로 무기체계 SW 개발 시 SW 신뢰성 시험을 적용할 것을 명기하였으며, 2014년 2월에 제정된 후 2016년 7월에 개정된 무기체계 SW 개발 및 관리 매뉴얼에서는 SW 신뢰성 시험 기준 및 대상이 대폭 상향되었다. 무기체계 연구개발 사업에만 신뢰성 시험을 적용하였으나 핵심기술(시험개발), 핵심 SW(시험개발), 신개념 기술시범사업(ACTD)도 시험 대상으로 포함하는 등 신뢰성 시험 대상이 대폭 확대되었다. 더 세부적으로 개정내용을 살펴보면 C/C++에 국한됐던 시험 대상 소스코드에 C#과 Java가 포함되었으며, 오프소스 이용 시 사용된 오픈소스 코드도 신뢰성 시험을 수행할 것을 요구한다. 정적시험인 코딩규칙 검사의 경우 방위사업청 코딩규칙 65개(공통 45개, C 전용 5개, C++ 전용 15개)를 만족시킬 것을 요구하던 수준에서 MISRA-C(143개)[1], MISRA-C++(228개)[2], JSF++(229개)[3], Java Coding Convention(28개)[4], C# Coding Convention(37개, 2015년 기준)[5] 등의 코딩규칙을 사업 특성에 맞게 테일러링하여 적용하도록 개정되었다. 또한 기존에는 없었던 소스코드 메트릭 점검 항목을 추가하여 Cyclomatic Complexity, Number of Call Levels, Number of Function Parameters 등에 대한 제한값을 설정하고 따르도록 규정하고 있다. 동적시험의 경우 함수단위의 구조기반 화이트박스 테스트 방식에서 소프트웨어 시험절차서에 기술된 시험절차를 이용하여 요구도 기반으로 소프트웨어 코드 실행률을 점검하여 목표값을 달성하도록 규정하면서 동적시험에 더 많은 공수가 필요할 것으로 예상된다.

2.1.1 동적시험 개요

동적시험은 소프트웨어가 실제 하드웨어에 탑재한 상태에서 소프트웨어 시험절차서에 기술된 시험절차를 이용하여 요구사항 기반으로 소프트웨어 코드 실행률을 점검하는 일련의 과정이라고 방위사업청의 무기체계 SW 개발 및 관리 매뉴얼에서 정의하고 있다[6]. 코드 실행률은 문장(Statement) 실행률, 분기(Branch) 실행률, MC/DC(Modified Condition/Decision Coverage)로 구성되며 동적시험 대상 SW의 CSCI, CSC, CSU 별 결함 발생빈도, 영향성을 평가하여 시험수준을 설정한다. 동적시험 수행 시기는 통상적으로 소프트웨어 통합시험 전 또는 체계 통합시험 전 1회, 개발시험 평가 전 1회, 규격화 전 1회 수행해야 하며, 양산 후 소스코드 형상이 변경될 때마다 재수행해야 한다.

동적시험 방법은 크게 2가지 유형으로 나눌 수 있다. 첫 번째는 LDRA, VectorCast 등의 도구를 이용하여 개발자가 테스트 케이스를 작성하고 테스트를 수행하는 방식이며, 두번째는 DT10 같은 도구를 이용하여 소스코드에 코드 실행률 검사를 위한 탐침코드를 자동으로 삽입하고 실제로 시험대상 소스코드를 빌드하여 생성된 실행파일을 소프트웨어 시험절차서에 따라 운용하면서 탐침코드를 통해 코드 실행률을 검사하는 방식이다. 본 논문에서는 VectorCast를 이용하여 C++로 작성된 소스코드에 대한 테스트 케이스를 작성한

후 테스트를 수행하여 코드 실행률을 검사하는 방법을 이용한다.

2.1.2 동적 시험 수행 절차

VectorCast를 이용하여 최초 동적시험을 수행하는 절차는 다음과 같다. VectorCast를 실행하고 시험 대상 소스코드 파일 1개마다 VectorCast 프로젝트 단위인 Environment를 생성한 후 각 Environment를 빌드한다. Environment 빌드가 완료된 후 시험 대상 함수에 대한 테스트 케이스를 작성한 후 VectorCast의 Test Execute 기능을 이용하여 코드 실행률 검사를 수행하고 보고서를 생성한다.

모든 Environment에 대한 코드 실행률 검사가 완료된 이후 소스코드가 변경되어 동적시험을 재수행해야 할 경우 최초 동적시험 절차와 동일하게 변경된 소스코드의 Environment를 빌드한 후 Test Execute를 수행하여 코드 실행률 검사를 수행한다. 이 때 만약 소스코드 변경으로 인해 코드 실행률이 충족되지 않을 경우 기존에 작성된 테스트 케이스를 수정하여 다시 Test Execute를 수행하고 코드 수행률 충족 여부를 검사한다. 최초 동적시험의 경우와 소스코드 변경 후 회귀 시험의 절차상 차이점은 테스트 케이스를 새로 작성하는가 아니면 기존에 작성된 테스트 케이스를 활용하여 수정하는가이다.

2.1.3 동적시험 수동 회귀시험 시 문제점

동적시험 회귀시험은 변경된 소스코드에 대한 테스트 케이스가 포함된 모든 Environment에 대해 Test Execute를 수행해서 변경된 소스코드로 인해 시험에 통과하지 못한 테스트 케이스가 있는지를 확인하고, 해당 테스트 케이스를 수정해서 최종적으로 목표 코드 실행률을 달성하는 과정이라고 할 수 있다. 이와 같이 회귀시험은 두 단계로 나눌 수 있으며 각 단계를 구체적으로 기술하면 다음과 같다. 첫 번째 단계는 변경된 소스코드를 이용하여 Environment를 빌드하고 Test Execute를 수행하여 해당 Environment에 포함된 테스트 케이스 중 시험에 통과하지 못한 테스트 케이스를 찾아내는 단계이고, 두 번째 단계는 첫 번째 단계에서 식별된 시험에 통과하지 못한 테스트 케이스만을 수정하고 Test Execute를 재수행해서 수정된 테스트 케이스가 시험에 통과할 수 있도록 하는 단계이다.

이 때 Environment 빌드의 경우 최소 수십 초에서 최대 수십 분이 소요되며, Test Execute 역시 코드 복잡도에 따라 수십 분이 소요되기도 한다. 따라서 최악의 경우 소스코드 파일 1개에 대한 테스트 케이스 시험 통과 여부 확인 위해 최대 수십여 분이 소요되고, VectorCast GUI를 이용하여 Environment 빌드 및 Test Execute만을 수행하기 위해 수십 초 또는 수십 분 간격으로 해당 기능의 툴바버튼이나 메뉴를 클릭해야 하며 이 과정동안 개발자가 상당시간 단순 대기해야 하는 문제가 발생한다.

또 다른 문제점은 VectorCast가 고가의 도구이기 때문에 규모가 큰 기업의 경우에도 개발자가 필요로 하는 만큼의 라이센스를 충분히 구입하기 어렵기 때문에 대부분의 기업은 VectorCast 구입 비용을 절감하기 위해 Floating 라이센스를 구입하여 다수의 개발자가 함께 사용하는 방법을 이용한다. VectorCast의 라이센스 정책은 GUI 실행 권한에 대한 라이센스와 Environment 빌드 또는 Test Execute에 대한 라이센스를 분리하고 있기 때문에 GUI 라이센스가 가용하여 VectorCast GUI를 실행하더라도 Environment 빌드 또는 Test Execute 라이센스가 가용하지 않을 경우 코드 실행률 검사를 할 수 없다. 만약 VectorCast Floating 라이센스가 모두 사용 중일 경우 개발자는 동적시험을 시작조차 할 수 없으며 라이센스 가용 여부를 수시로 확인한 후 동적시험을 수행해야 하는 불편함이 있다.

2.2 동적시험 회귀시험 자동화 프로그램 설계

2.2.1 코드 실행률 검사 자동화

VectorCast는 GUI를 이용한 코드 실행률 검사 외에도 명령행을 이용하여 코드 실행률 검사를 할 수 있는 clicast(Command LIne vectorCAST) 라는 인터페이스를 제공한다. VectorCast Environment Manager는 clicast에 대한 프론트엔드 역할을 하여 Environment 빌드, Test Execute, 보고서 생성을 배치(Batch) 처리 방식으로 자동화하도록 설계하였다.

본 논문에서는 VectorCast를 대상으로 하는 동적시험 회귀시험 자동화 프로그램에 대해 기술하였으나 VectorCast처럼 명령행 인터페이스 제어용 실행파일을 따로 제공하거나 소켓통신 등 GUI 이외의 동적시험 제어 인터페이스를 제공하는 어떠한 동적시험 툴도 인터페이스를 플러그인 형태로 모듈화하여 회귀시험을 자동화 할 수 있을 것으로 예상된다.

동적시험 회귀시험 대상 소스코드를 이용하여 생성된 Environment가 저장된 폴더를 지정하면 Fig. 1처럼 해당 폴더에 저장된 모든 Environment 목록을 보여주며 이전에 수행한 코

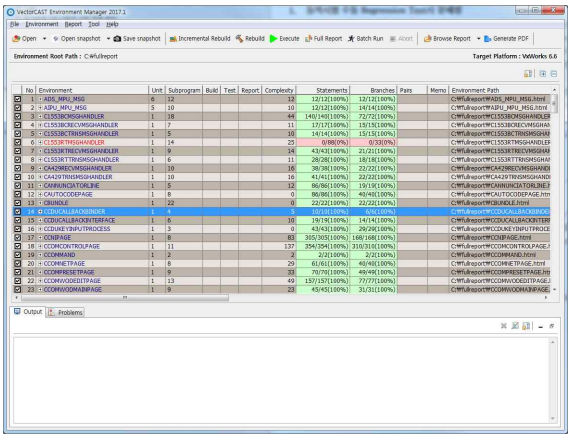

Fig. 1. Main window of VectorCast Environment Manager

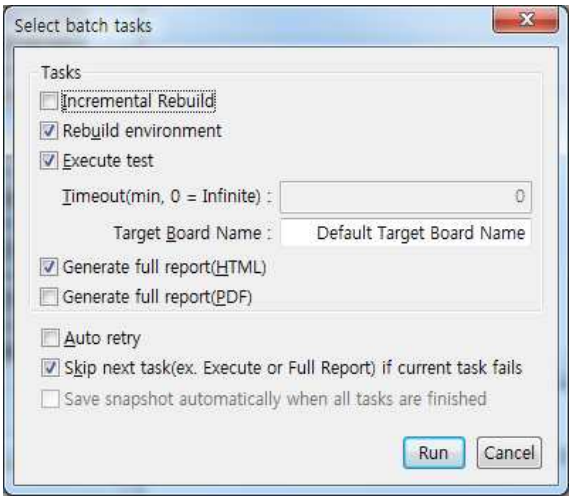

Fig. 2. Selection of batch tasks

드 실행률 검사 결과를 확인할 수 있다.

앞서 기술한 바와 같이 소스코드 변경 후 회귀시험을 수행할 경우 코드 실행률 충족 여부 판단을 위해 기존의 Environment를 빌드하고 Test Execute를 수행해야 한다. VectorCast Environment Manager는 이 과정을 자동화하기 위해 Fig. 1의 Environment 목록에서 선택된 Environment를 대상으로 배치 작업을 수행할 수 있도록 Fig. 2와 같이 배치 작업 시 수행할 하위 작업을 선별적으로 선택할 수 있는 기능을 구현하였다.

선택 가능한 하위 작업에는 Incremental Rebuild(변경된 소스코드에 대한 테스트 케이스만 빌드), Rebuild environment(Environment에 포함된 전체 테스트 케이스 리빌드), Execute test(코드 실행률 검사), Generate full report(코드 실행률 검사 결과 보고서 생성)가 있으며 VectorCast Floating 라이센스가 가용하지 않을 경우 자동으로 다시 시도할 수 있는 옵션을 추가

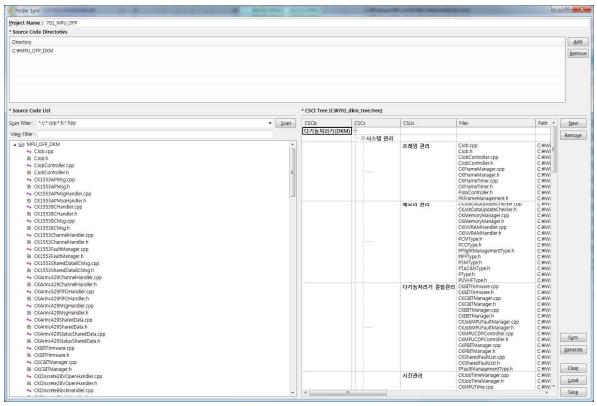

Fig. 3. GUI of source code information generator

하였다. 앞서 기술한 바와 같이 Environment 하나에 대한 코드 실행률 검사를 위해서는 Environment 빌드, 테스트 케이스 실행(Execute Test), 보고서 생성의 절차를 거치며 Environment 빌드 시 빌드 에러가 발생한 경우 Execute Test나 보고서 생성을 할 필요가 없기 때문에 Environment 빌드 실패 시 Execute Test, 리포트 생성 절차를 건너뛰고 다음 Environment에 대해 빌드, Execute Test, 리포트 생성 절차를 수행함으로써 전체 Environment 코드 실행률 검사 시간을 단축할 수 있도록 하였다.

또한 VectorCast를 이용한 동적시험은 상당한 시간이 소요되기 때문에 전체 Environment에 대한 코드 실행률 검사를 연속적으로 할 수 없으며 동적시험을 중단했다가 이어서 할 수 있도록 수행중이던 동적시험 상태를 저장하거나 저장된 마지막 상태를 불러와서 나머지 동적시험을 수행할 수 있는 스냅샷 기능을 구현하였다.

2.2.2 동적시험 수행결과 보고서 기초자료 자동 생성

무기체계 SW 개발시험평가 시 개발시험평가용 소스코드 형상을 대상으로 수행된 신뢰성 시험 결과 보고서를 SW 신뢰성 시험 평가 기관에 제출해야 한다. 보고서에 포함되는 주요 내용에는 소스코드 파일 목록, 각 소스코드 파일의 수정날짜, CRC, 크기, LOC 등이 있으며 소스코드 형상 변경 시 Fig. 3과 같이 보고서에 포함될 내용을 자동으로 일괄 추출하여 결과 보고서에 바로 붙여 넣을 수 있는 CSV 파일로 Export 하는 기능을 구현하였다.

2.3 VectorCast Environment Manager 구현

VectorCast Environment Manager는 윈도우 환경에서 실행되도록 구현되었으며, 개발 환경은

다음과 같다.

- 개발언어 : Java
- IDE : Eclipse
- GUI 툴킷 : SWT(Standard Widget Toolkit), WindowBuilder Pro

앞서 기술한 바와 같이 VectorCast Environment Manager는 VectorCast의 명령행 인터페이스 제어를 위한 프론트 엔드 역할을 수행하며 설계의 핵심은 Command Line 인터페이스 기능을 제공하는 VectorCast의 clicast.exe 실행파일을 외부 프로세스로 실행하고 결과를 받아와서 GUI에 시현하는 것이다.

2.3.1 clicast 인터페이스 제어

VectorCast GUI를 직접 조작하지 않고 VectorCast가 제공하는 기능을 이용하기 위해서는 clicast 인터페이스를 제어해야 한다. VectorCast Environment Manager가 clicast 인터페이스를 통해 제공하는 주요 기능은 Environment Incremental Rebuild, Environment Rebuild, Test Execute, Full Report 생성이다.

clicast 인터페이스는 clicast.exe라는 명령행 처리를 위한 전용 실행파일을 통해 제어할 수 있으며 각 기능별 인터페이스 제어명령은 Table 1과 같다.

clicast.exe를 통한 명령행 인터페이스 제어는 VectorCast Environment Manager 프로세스가 clicast.exe라는 별도의 프로세스를 실행하고 clicast.exe의 표준 출력 스트림을 리다이렉션해서 VectorCast Environment Manager의 Output창에 시현하는 방식이다. Java 프로세스에서 외부 프로세스를 실행하는 방법은 여러가지가 있지만 가장 많이 사용되는 방식은 Java의 Process 클래스를 사용하거나 Apache 소프트웨어 재단에서 관리하는 오픈소스 프로젝트 중 Java 기반의 재사용 가능한 컴포넌트를 모아놓은Apache Commons 라이브러리를 이용하는 것이다. Java의 Process 클래스는 Java의 표준 라이브러리에 기본적으로 포함되어 프로젝트 구성이나 사용이 용이하지만 외부 프로세스의 표준 출력 스트림을 받아올 때 외부 프로세스로부터 받아온 표준 출력 메시지를 즉시 처리하지 않으면 외부 프로세스가 멈추거나 교착상태에 빠질 수 있는 단점이 있으며 표준 출력 스트림과 에러 출력 스트림을 동시에 처리하기 위해서는 별도의 스레드를 직접 구현해야 하는 등 필요 이상의 작업이 필요하다. 따라서 VectorCast Environment Manager는 Apache Commons 라이브러리 중 Java 외부 프로세스 실행 및 환경관리를 위한 Exec 컴포넌트를 사용하였다.

Table 1. Commands for controlling command line interface

Functions	Commands
Incremental rebuild	clicast.exe -lc -e [Environment Name] EN Incremental Rebuild
Environment rebuild	clicast.exe -lc -e [Environment Name] EN RE_B
Test Execute	clicast.exe -lc -e [Environment Name] EN Batch
Full report generation	clicast.exe -lc -e [Environment Name] Reports Custom Full [html Path]

2.4. VectorCast Environment Manager 적용 실험결과

VectorCast Environment Manager는 개발시험 평가 시 SW 신뢰성 시험 평가 준비를 위한 내부용 툴로 개발되었으며, 실제 항공기 임무컴퓨터 OFP의 신뢰성 시험 개발시험 평가에 이용하였다. 개발시험 평가 준비를 위해 총 2차례에 걸쳐 회귀시험을 수행하였다. 한 번은 VectorCast Environment Manager를 이용하지 않고 개발자가 직접 VectorCast GUI를 이용하여 일일이 코드 실행률을 검사하였고, 나머지 한 번은 VectorCast Environment Manager를 이용하여 자동으로 코드 실행률을 검사하여 소요시간을 측정하였다. 2회 모두 소스코드가 변경된 상태에서 코드 실행률을 검사하였으며, 코드실행률을 충족시키지 못하는 테스트 케이스를 식별하기 위한 목적으로 테스트 케이스 수정은 하지 않고 변경된 소스코드의 코드 실행률만을 검사하였다. 그 결과 VectorCast Environment Manager를 이용한 경우 비교적 라이센스 여유가 있는 야간에도 사람의 GUI 조작없이 자동으로 코드 실행률 검사를 할 수 있었으나 주간에만 개발자가 직접 GUI를 조작하면서 코드 실행률 검사를 한 경우 VectorCast 라이센스가 가용하지 않은 시간이 많았기 때문에 전체적으로 소요시간이 증가하였다. 그 결과 VectorCast Environment Manager를 이용한 경우 전체적인 소요시간이 약 70% 정도로

Table 2. Comparision of total elapsed time for regression test

Regresstion Test Method	Test Time	Test manpower
VEM Unused	104hrs (=13 days X 8 hrs)	13 M/D
VEM Used	72 hrs (=3days X 24 hrs)	0 M/D

줄어 들었음을 알 수 있다. 측정 결과 소요시간이 30% 정도 차이가 나지만 다른 측면에서 실험결과를 분석해보면 VectorCast Environment Manager 미 이용 시 소요시간 104시간 중 상당 시간을 개발자가 코드 실행률 검사를 위해 GUI를 조작해야 했던 반면, VectorCast Environment Manager 이용 시 전 과정이 자동화되었기 때문에 개발자는 그 시간동안 다른 업무를 할 수 있었으므로 더 효율적이라고 할 수 있다. 실험환경은 다음과 같다.

- VectorCast 버전 : 6.3f
- 시험 대상 SW 개발 언어 : C++
- 시험 대상 SW 실행환경 : VxWorks 6.6
- 컴파일러 : WindRiver Diab 5.6.0.0
- 시험 대상 소스코드 파일 개수 : 402개
- Environment 개수 : 231개

III. 결 론

본 논문에서는 VectorCast를 이용한 동적시험 회귀시험 시 기존에는 개발자가 직접 GUI를 통해 코드 실행률 검사를 하던 방식에서 코드 실행률 검사를 배치 처리 방식으로 자동화하고 결과보고서를 자동으로 생성해주는 VectorCast Environment Manager 프로그램에 대해서 기술하였다. VectorCast Environment Manager를 이용한 코드 실행률 검사 시 테스트 케이스 수정을 제외한 코드 실행률 검사 전과정이 자동화되고 VectorCast 라이센스가 가용하지 않을 경우 라이센스가 가용할 때까지 자동으로 재시도함으로써 개발자가 코드 실행률 확인에 투입하는 시간을 상당히 줄일 수 있어서 효율적임을 확인할 수 있었다. 본 논문에서 제안한 VectorCast Environment Manager는 윈도우에서 동작하며 WindRiver VxWorks 6.6/6.9와 Visual Studio 2010/2013을 지원한다. 추가 연구에서는 리눅스 버전을 개발하고 더 많은 컴파일러를 지원할 계획이다.

References

1) MISRA-C, “https://www.misra.org.uk/misra-c“

2) MISRA-C++, “https://www.misra.org.uk/misra-cpp”

3) Lockheed Martin Corp., “JOINT STRIKE FIGHTER C++ CODING STANDARDS FOR THE SYSTEM DEVELOPMENT AND DEMONSTRATION PROGRAM”, 2005

4) Java Code Conventions, “http://www.oracle.com/technetwork/java/codeconvtoc-136057.html”

5) C# Coding Convections, “https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions”

6) The weapon system SW development and management manual, “www.dapa.go.kr”