韓國 컴퓨터情報學會 論文誌
第14卷 第1號, 2009. 1.

2009-14-1-1-3

XML 테스트 스트립트 기반의 내장형 시스템 소프트웨어 단위 테스트 도구

곽 동 규*, 유 재 우*, 조 용 윤*

A Software Unit Testing Tool based on The XML Test Script for Embedded Systems

Donggyu Kwak*, Chae-Woo Yoo*, Yongyun Cho*

요 약

내장형 시스템의 요구사항이 증가함에 따라 내장형 시스템 소프트웨어의 복잡도가 증가하고 있다. 그러므로 내장형 시스템 소프트웨어 테스팅의 필요성이 높아지고 있다. 본 논문은 내장형 시스템 소프트웨어의 테스트를 효과적으로 실행하기 위해 교차 개발 환경에 적합한 테스팅 도구를 제안한다. 본 도구는 호스트/타겟 구조로 이루어져 있으며 호스트에서 작성한 테스트 케이스를 타겟에서 실행할 수 있는 직관적인 환경을 제공한다. 그리고 제안하는 도구의 테스트 케이스는 XML 기반의 테스트 스크립트를 이용한다. 또한, 직관적인 테스트 케이스를 작성하기 위해 트리와 테이블 기반의 테스트 스크립트 편집기를 갖는다. 제안하는 테스트 도구는 테스트 케이스 작성에서부터 결과확인까지 직관적인 GUI를 제공하여 테스트 케이스 작성에 대한 부담을 경감시키는 장점을 가진다.

Abstract

According to increasing requirments in embedded systems, embedded software has been more complicated then before. a software developer is required to test her/his software to make a efficient embedded system software in both time and space. This paper suggests a testing tool with which a software developer can easily test the embedded system software in cross-development environments. The suggested tool is designed based on host/target architecture, to provide an intuitive test environment in which a test case can be executed in a target board. The tool uses an XML-based test script to generate an appropriate test case. It includes a tree-based test script editor with which a developer can easily make a test case. Therefore, with the suggested tool, a develop can put down a burden on an software testing and get more productivity in software development related on embedded system.

▸ Keyword : 소프트웨어 테스트(software testing), 내장형 시스템(embedded system), XML, 단위 테스트(unit testing),

• 제1저자 : 곽동규
• 투고일 : 2008. 11. 9, 심사일 : 2008. 11. 25, 게재확정일 : 2008. 12. 17.
* 숭실대학교 컴퓨터 학과
※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.

www.kci.go.kr

Ⅰ. 서 론

최근 내장형 시스템(Embedded System)의 요구사항이 증가함에 따라 탑재되는 소프트웨어의 복잡도가 증가하고 있다[1]. 이에 따라 신뢰성 높은 소프트웨어의 생산이 어려워지고 있다. 신뢰성이 높은 소프트웨어란 개발자의 의도에 합당하게 동작하는 소프트웨어를 의미한다. 신뢰성이 높은 소프트웨어를 생산하기 위해서는 소프트웨어를 검증할 수 있는 방법이 필요하고 그에 따른 테스트 방법이 연구되어 왔다[2][3][4][5]. 현재 내장형 시스템에서 가장 많이 사용되는 프로그래밍 언어는 함수를 단위(Unit)로 하는 절차적 프로그래밍 언어이다. 본 논문에서는 내장형 시스템 소프트웨어 개발을 위한 단위 테스트 도구에 대해서 연구한다. 단위 테스트 도구에서 단위는 서브프로그램의 집합을 의미한다[7]. 이 레벨은 테스트하기에 적합한 크기를 말하며, 이보다 더 큰 시스템 레벨의 테스트는 의미를 분석하는데 너무 많은 코드가 관여하여 테스트 레벨로서 부적합하다[8]. 일반적으로 많이 사용하는 언어에서는 함수가 하나의 단위로 분류 될 수 있다.

소프트웨어를 테스트하기 위해서는 테스트 케이스(Test Case)를 작성해야 한다[6]. 테스트 케이스는 테스트 데이터(Test Data)와 테스트 경로(Test Path)로 이루어져 있다. 테스트 데이터는 단위 소프트웨어의 일어날 수 있는 입력과 예상되는 출력이고 테스트 경로는 각 단위간의 관계를 의미한다. 일반적인 테스트 툴은 테스트 케이스를 작성하기 위한 테스트 API[5]를 제공하거나 테스트를 위한 독자적인 테스트 스크립트 언어[2]를 제공하여 사용자가 테스트 케이스를 작성하도록 한다. 하지만 기존의 방법들은 테스트 API나 독자적인 테스트 스크립트를 학습해야하는 부담을 가지고 있다. 본 논문이 제안하는 도구는 XML을 기반으로 하는 테스트 스크립트와 GUI 기반의 에디터를 제안한다. XML은 데이터 표현 방식의 표준으로 대부분의 개발자에게 친숙하고 구조가 단순하여 학습하기 쉬운 장점을 가진다. 또한 테스트 스크립트를 작성하기 위한 GUI 기반의 에디터는 직관적인 테스트 스크립트 작성을 위한 환경을 제공한다.

일반적으로 내장형 시스템 소프트웨어 개발은 타겟 시스템의 컴퓨팅 자원이 한정적이기 때문에 교차 개발 환경(Cross-Development Environment)을 이용한다. 교차 개발 환경은 호스트에서 작성한 프로그램을 크로스 컴파일러(Cross Compiler)를 사용하여 타겟 시스템에서 실행 시키는 환경이다. 그러므로 내장형 시스템 소프트웨어를 테스트하기 위한 환경도 호스트/타겟 구조의 교차 개발 환경이어야 한다. 본 논문이 제안하는 도구는 내장형 시스템에서의 소프트웨어 테스트를 위한 호스트/타겟 구조의 테스트 환경을 제안한다. 그림 1은 내장형 시스템에 적합하도록 설계된 호스트/타겟 구조의 시스템 구성도이다.

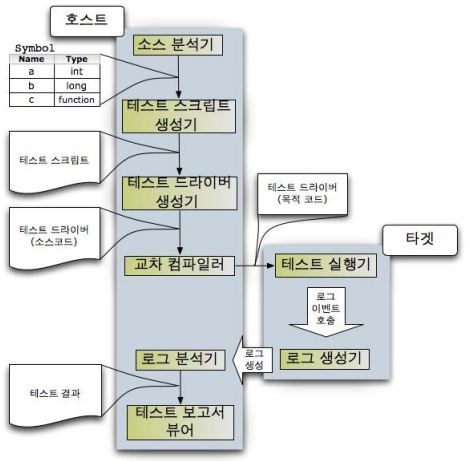

그림 1. 테스트 도구의 시스템 구성도
Fig 1. Testing Tools System Architecture

제안하는 도구는 내장형 시스템 소프트웨어 개발환경에서 사용이 용이하도록 호스트/타겟 구조를 갖는다. 호스트에서 작성된 프로그램 소스와 테스트 케이스는 테스트 작업용구(Test Harness)로 변환하여 테스트 프로그램을 생성한다. 생성된 테스트 실행 프로그램은 통신을 통해 타겟에 전송되고 테스트 프로그램을 실행하여 테스트 결과를 받아 사용자에게 직관적인 표나 그래프로 제공한다.

제안하는 도구는 XML 기반의 테스트 스크립트를 사용하여 테스트 스크립트에 대한 학습 부담을 줄이고 테스트 스크립트를 제공하여 간단한 테스트 케이스는 스크립트의 학습 없이도 작성 가능한 장점을 가진다. 또한 테스트 스크립트에서 테스트 결과에 따른 분기를 제공하고 있어 테스트 결과에 따라 다양한 테스트를 실시할 수 있는 장점을 가지고 있다. 본 논문은 제2장에서 내장형 시스템에 합당한 시스템 구성을 보인 후 3장 테스트 스크립트 언어의 설계에서 제안하는 테스트 스크립트를 소개하고 4장에서 실험을 소개하고 5장에서 결론에 관해 논한다.

Ⅱ. 테스트 스크립트 언어의 설계

www.kci.go.kr

소프트웨어의 단위 테스트를 수행하기 위해서는 테스트 작업용구(Test harness)가 요구된다. 테스트 작업용구는 테스트 대상 소스코드와 스텁, 테스트 드라이버로 구성된다. 이 중 테스트 드라이버는 테스트 대상 소스코드의 프로그래밍 언어와 동일한 언어로 작성되어 테스트 대상을 호출하고 그 수행 결과를 수집한다. 테스트 드라이버는 테스트 수행자가 직접 테스트 대상 소스코드와 동일한 프로그래밍 언어로 작성할 수 있다. 하지만 이 방법은 테스트 수행자가 테스트 대상 소스코드와 대상 프로그래밍 언어에 대한 높은 이해가 필요하다. 그에 따라 테스트 수행에 대한 부담이 증가하고 테스트 수행자는 테스트 내용에 집중하기 어렵다. 테스트 드라이버 작성에 대한 부담을 감소시키는 방법으로는 테스트 스크립트를 사용하는 방법이 있다. 하지만 일반적으로 테스트 스크립트는 독자적인 문법을 가지고 있어 테스트 스크립트에 대한 학습 부담을 가진다.

본 논문은 테스트 수행자의 학습 부담을 줄이기 위해 XML을 기반으로 하는 테스트 스크립트를 제안한다. 제안하는 XML 기반 테스트 스크립트는 개발자에게 친숙한 XML을 기반으로 하고 있어 테스트 스크립트에 대한 학습 부담이 적다. 그림 2는 제안하는 테스트 스크립트의 스키마이다.

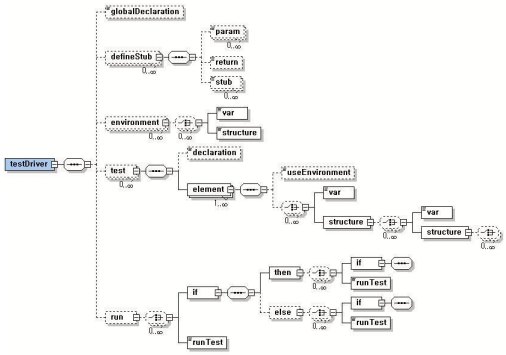

그림 2. 테스트 스크립트 스키마
Fig 2. Test Script Schema

그림 2와 같이 제안하는 테스트 스크립트는 테스트 기술부와 테스트 실행부로 나누어져 있다. 스크립트 문서는 testDriver 태그를 루트로 하고 테스트 기술부는 test 태그로 테스트 실행부는 run 태그에 표현한다. globalDeclaration 태그는 헤더 파일의 포함이나 전역 변수를 선언하는데 사용한다. 그리고 defineStub 태그는 스텁을 정의하기 위한 태그이다. 스텁은 테스트를 실행하는 현재 개발되지 않은 모듈을 위한 임시 모듈이다. environement 태그는 테스트 변수를 설정하는데 사용된다. 그림 3은 테스트 스크립트의 예제이다.

```
<testDriver>                                          테스트
  <test id="add">                                     기술부
    <declaration>
      <![CDATA[
        int x, y, rval_add;
      ]]>
    </declaration>
    <element>
      <var name="x" init="3" />
      <var name="y" init="4" />
      <var name="rval_add" init="7" />
      <execution>
        <![CDATA[
          rval_add = add(x, y);
        ]]>
      </execution>
    </element>
  </test>

  <run>
    <runTest id="add" />                              테스트
  </run>                                              실행부
</testDriver>
```

그림 3. 테스트 스크립트 예제
Fig 3. Test Script Example

예제에서 테스트 대상이 되는 소스는 두 개의 정수를 더하여 반환하는 "int add(int x, int y)" 함수이다. 테스트 스크립트의 테스트 기술부에서는 테스트 단위에 해당하는 테스트 케이스를 기술한다. 테스트 케이스는 입력 데이터와 예상되는 출력 데이터를 포함한다. 테스트 실행부는 테스트 케이스의 연속적인 결합으로 테스트 케이스의 실행을 표현한다. 개발자가 작성한 테스트 스크립트는 시스템을 통해 테스트 대상 언어와 동일한 테스트 드라이버로 변환되어 테스트 결과를 수집한다.

III. 테스트 스크립트와 드라이버 생성기

본 도구는 XML 기반의 테스트 스크립트를 제공하여 테스트 스크립트에 대한 학습 부담을 경감시켰으나 여전히 학습 부담이 존재한다. 그러므로 직관적인 GUI 기반의 테스트 스크립트 생성기를 제안한다. 제안하는 GUI 기반 통합개발환경(IDE : Integrated Development Environment)은 이클립스(Eclipse)[9] 기반의 인터페이스를 가진다. 이클립스는 오픈 프로젝트로 진행 중인 통합개발환경으로써 플러그 인(plug-in) 방식으로 확장이 용이한 장점을 가지고 있다. 또한 CDT(C/C++ Development Tooling)나 CVS(Concurrent Versions System)과 같은 기존의 다양한 개발도구를 지원

www.kci.go.kr

한다. 그림 4는 테스트 대상 예제 프로그램과 GUI 기반의 테스트 스크립트 생성기를 보인다.

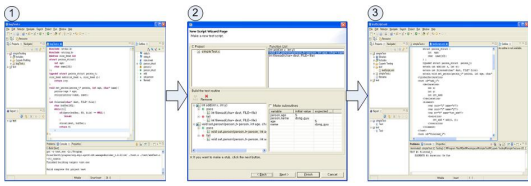

그림 4. 테스트 스크립트 생성기
Fig 4. Test Script Generator

그림 4는 제안하는 도구로서 이클립스의 플러그인으로 제공된다. 테스트를 실행하는 개발자는 이클립스가 제공하는 다양한 개발도구를 이용하면서 테스트에도 집중할 수 있다. 그림 3에서 ①은 이클립스 뷰어를 통해 본 테스트 대상 프로그램이다. ②는 테스트 스크립트를 생성하는 테스트 스크립트 편집기로 테스트 대상 코드를 분석하여 사용자에게 테스트 대상 코드의 함수를 보여준다. 테스트 스크립트 편집기는 사용자가 선택한 프로그램 함수의 입력과 출력의 자료형을 제공하고 테스트 데이터를 입력할 수 있는 방법을 제공한다. 또한, 테스트 경로를 제공하여 다양한 테스트를 작성할 수 있다. ③은 테스트 스크립트 생성기로부터 생성한 테스트 스크립트이다.

개발자는 앞에서 소개한 테스트 스크립트 생성기를 이용하여 직관적인 트리와 테이블로 테스트 케이스를 작성할 수 있다. 직관적인 트리와 테이블로 작성된 테스트 케이스는 테스트 스크립트로 변환한다. 그림은 직관적으로 작성된 테스트 케이스의 자료 구조이다.

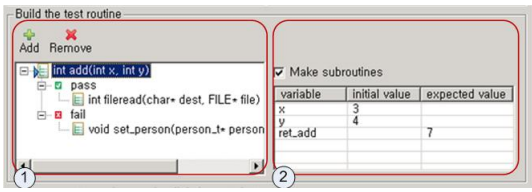

그림 5. 테스트 케이스를 작성하기 위한 GUI
Fig 5. GUI for Test Script Edit

그림 5는 테스트 케이스를 작성하기 위한 사용자 인터페이스이다. 그림에서 ①은 테스트 경로를 작성하기 위한 트리이고 ②는 단일 테스트 케이스를 작성하기 위한 테이블이다. 사용자는 ①의 트리를 통해 테스트 경로를 작성하고 단위 테스트를 선택하면 ②의 테이블을 통해 변수의 초기 값과 예상 값을 작성할 수 있다.

사용자가 입력한 정보를 저장하기 위해서 다음과 같은 자료구조를 제안한다. 표 1은 사용자의 입력을 저장하는 자료구조이다.

표 1. 사용자의 입력을 저장하는 자료구조
Table 1. Data Structure for User Input

```
class TestCaseRootNode {
  String globalDeclaration;
  Vector<TestCase> testCases;
  Vector<TestCaseNode> testCaseNodes;
}
class TestCaseNode {
  String name;
  Vector<TestCaseNode> childNodes;
  String condition;
  TestCaseNode successChildNode;
  TestCaseNode failChildNode;
}
class TestCase {
  String id;
  Vector<TestVar> varVector;
  String execution;
}
class TestVar {
  String name = null;
  String init = null;
  String ev = null;
}
```

표 1은 직관적으로 작성된 테스트 케이스의 자료 구조이다. 자료구조는 단위 테스트와 테스트 경로를 트리의 형태로 저장하고 있으며 TestCaseRootNode 클래스가 루트노드이다. 루트노드는 라이블러리 참조와 함수 선언을 다루는 String 클래스가 있고 단위 테스트를 TestCase로 테스트 경로를 TestCaseNode로 다룬다. 트리를 통해 입력된 데이터는 표 2와 같은 알고리즘으로 테스트 스크립트를 생성한다.

표 2는 사용자가 입력한 데이터를 4장에서 소개한 테스트 스크립트로 변환하는 알고리즘이다. 본 알고리즘은 Gamma[11]가 제안한 디자인 패턴(design pattern)중 비지터 패턴(visitor pattern)을 사용하여 작성한다. 작성된 테스트 스크립트는 더욱 정교한 테스트를 실행하기 위해 개발자가 수정할 수 있다.

표 2. 테스트 스크립트 생성 알고리즘
Table 2. Algorithm for Test Script Generate

```
class GenTestScriptVisitor extends Visitor {
  public void visit(TestCaseRootNode n) {
    makeHead();
    //  xml version과 DOCTYPE을 정의
```

www.kci.go.kr

```
    startTestDriver();  //〈testDriver〉 태그 시작
    if(n.globalDeclaration != null){
      startGlobalDeclaration();
      // 〈globalDeclaration〉 태그 시작
      makePCDATA(n.globalDeclaration);
      // lib 참조와 함수 선언
      endGlobalDeclaration();}
    for(int i = 0; i 〈 n.testCases.size(); i++){
      n.testCases.elementAt(i).accept(this);
      // test case 작성}
    for(int i=0; i 〈 n.testCaseNodes.size(); i++){
      startRun();  //〈run〉 태그 시작
      n.testCaseNodes.elementAt(i).accept(this);
      // test path 작성
      endRun();}
    endTestDriver();}
  protected void visit(TestCaseNode n) {
    if(n.condition == null){
      startTest(n.name);// 〈test〉 태그 시작과 끝
      for(int i=0; i 〈 n.childNodes.size(); i ++){
        n.childNodes.elementAt(i).accept(this);}
    }else{
      startIf(n.name);// 〈if〉 태그 시작
      startThen();// 〈then〉 태그 시작
      n.successChildNode.accept(this);
      endThen();
      startElse();// 〈else〉 태그 시작
      n.failChildNode.accept(this);
      endElse();
      endIf();}}
  protected void visit(TestCase n) {
    startTest(n.id);// 〈test〉 태그 시작
    startElement();// 〈element〉 태그 시작
    for(int i = 0; i 〈 n.varVector.size(); i++){
      n.varVector.elementAt(i).accept(this);}
    startExecution();// 〈execution〉 태그 시작
    makePCDATA(n.execution);// 실행 소스 생성
    endExecution();
    endElement();
    endTest();}
  protected void visit(TestVar n) {
    startVar();// 〈var〉 태그 시작
    if(n.init != null){
      makeAttribute("init", n.init);// init 속성 생성}
    if(node.ev != null){
      makeAttribute("ev", n.ev);// ev 속성 생성}
    makeAttribute("name", n.name);}}
```

IV. 제안하는 도구를 이용한 테스트 및 실험

본 논문은 제안하는 도구를 실험하기 위해 간단한 테스트 대상 소스와 실험 결과를 보인다. 표 3은 실험 환경을 보인다.

표 3. 테스트 환경
Table 3. Testing Environment

항목		사용 장치 및 소프트웨어
호스트 (Host)	프로세서	Intel Pentium 4
	운영체제	Microsoft Windows XP
	개발언어	J2SDK 5.0
	개발환경	Eclipse 3.1 Platform
타겟 (Target)	프로세서	Samsung S3C2440A (ARM9 core)
	시스템	MDS(社) HRP-SC2440 (REBIS)
	운영체제	Linux (kernel version 2.4.18)
교차 개발 도구 (Cross Tool Chain)		GNU Cross Tool Chain for Cygwin - GNU C Compiler (gcc) v2.95 for ARM - ld, as, ar, objdump, objcopy - C Library (libc)

표 3과 같은 개발 환경은 내장형 시스템의 대표적인 ARM 시스템과 일반적인 호스트 환경인 윈도우 환경이다. 또한 프로그램 언어로 자바를 사용하여 높은 이식성을 갖는다. 표 4는 예제로 보이는 테스트 대상 코드이다.

표 4. 테스트 대상 코드
Table 4. Testing Example

```
1  /* file name : main.c */
2
3  #include 〈stdio.h〉
4  #include 〈string.h〉
5  struct person_struct{
6          int age;
7          char name[10];
8  };
9  typedef struct person_struct person_t;
10 int add(int x, int y){
11         return x + y;
12 }
13 void set_person(person_t* person, int age, char* name){
14         person->age = age;
15         strcopy(person->name, name);
16 }
17 int fileread(char* name, FILE* file){
18         char buffer[80];
19         while(1){
20             if(fgets(buffer,80,file)==null)
21               break;
22             strcat(dest, buffer);
23             return 0;
24         }
25 }
```

www.kci.go.kr

표 4의 테스트 대상 코드는 세 함수로 구성되어 있다. 함수 “add”는 두 인수(parameter)를 입력으로 하고 리턴을 갖는다. 함수 “set_person”는 하나의 구조체에 두 값을 셋팅하는 함수이다. 또한, 함수 “fileread”는 파일을 읽어 인수에 저장한다. 본 장에서는 이 세 함수를 통해 실험을 보인다. 그림 6은 테스트 대상 코드를 분석하여 사용자에게 정보를 제공해 주고 테스트를 작성할 수 있는 테스트 스크립트 편집기이다.

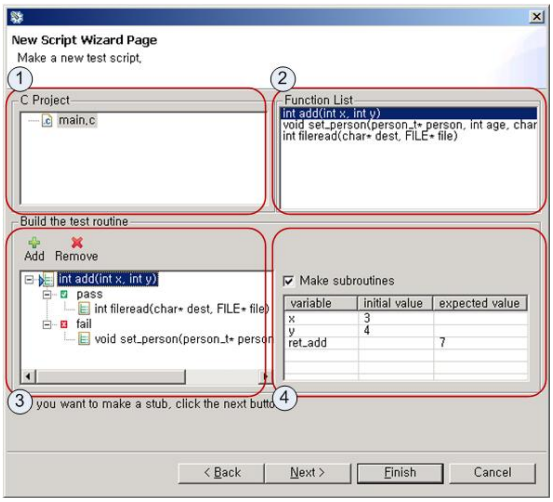

그림 6. 테스트 스크립트 편집기
Fig 6. Test Script Editor

테스트 스크립트 편집기는 네 부분으로 이루어져 있다. 그림 6에서 ①은 프로젝트에 존재하는 소스 프로그램 파일을 나타낸다. 테스트 실행자는 테스트 대상이 작성되어 있는 프로그램 소스 파일을 선택한다. ②는 선택한 프로그램 소스 파일에 존재하는 함수의 리스트를 보여준다. ③은 테스트 경로를 작성하는 창으로 사용자는 ②에서 함수를 선택하여 테스트 경로를 설정할 수 있다. ④는 ③에서 작성한 테스트 케이스에 입력과 예상 출력으로 사용될 테스트 데이터를 작성할 수 있는 테이블이다. 그림 6의 테스트 스크립트 편집기로 작성한 테스트 스크립트는 제 3장에서 보인 테스트 스크립트 XML 문서로 작성된다. 작성된 테스트 스크립트는 테스트 타겟 언어로 변환하여 타겟 시스템으로 전송되고 내장형 시스템인 타겟 시스템에서 실행된다. 실행 시 로그 생성기를 통해 XML 로그가 생성되고 생성된 XML 로그는 테스트 보고서 뷰어로 분석된다. 그림 7과 그림 8은 테스트 결과를 생성하는 XML 문서의 스키마와 테스트 경로에 따른 결과를 보여준다.

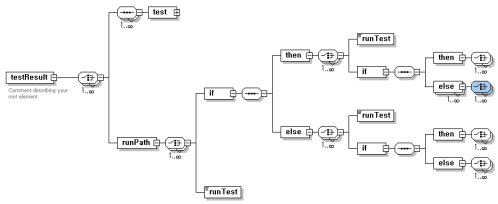

그림 7. 테스트 결과 XML 스키마
Fig 7. Testing Result Script Schema

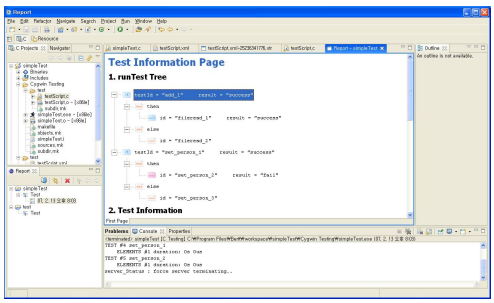

그림 8. 테스트 경로에 따른 결과 화면
Fig 8. Result Screen on Testing Path

그림 7과 그림 8의 결과는 테스트 경로에 따른 테스트 결과를 트리의 형태로 보여준다. 하지만 이 결과 트리는 상세한 결과를 확인할 수 없다. 테스트를 실행하는 개발자는 스크립트에서 작성한 예상 결과와 실행 결과를 비교 분석하기를 요구한다. 그림 9는 테스트에 대한 상세 결과를 테이블 형태로 제공하는 테스트 결과 화면이다.

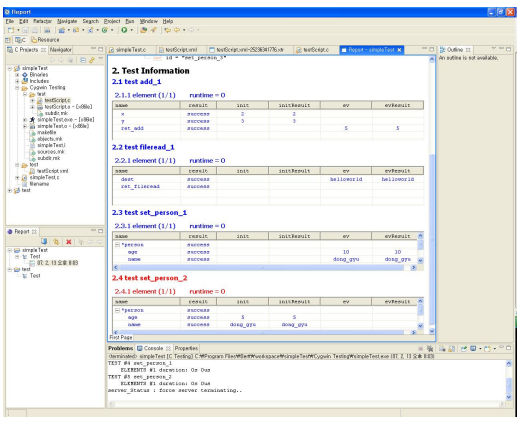

그림 9. 테스트 결과 화면(테이블)
Fig 9. Testing Result Screen(Table)

www.kci.go.kr

제안하는 도구는 직관적인 테스트 방법을 제공한다. 사용자는 직관적인 편집기를 이용하여 테스트 대상 소스를 선택하고 테스트 케이스를 작성할 수 있다. 또한, 본 도구는 자동으로 테스트 실행 프로그램을 작성하고 타겟 시스템에 전송 실행한다. 테스트 결과는 직관적인 트리와 테이블로 제공하여 사용자가 이해하기 용이하다. 제안하는 도구를 실험하기 위해 본 논문은 대표적인 세 가지 케이스의 C언어 프로그램을 이용하여 테스트 한다. 세 가지 테스트 케이스는 표 5와 같다.

표 5. 실험 케이스
Table 5. Testing Case

테스트 프로그램 타입	테스트 결과 예상	프로그램 갯수	예상 결과에 적중한 프로그램
반환 값을 가지는 함수	테스트 결과 성공	52	52
	테스트 결과 실패	48	48
매개변수를 통한 값의 할당 함수	테스트 결과 성공	51	51
	테스트 결과 실패	50	50
파일을 통한 입출력 함수	테스트 결과 성공	49	49
	테스트 결과 실패	50	50

세 가지 케이스의 총 300개의 함수를 통해 테스트를 실행한 결과 테스트 결과는 예상에 따라 실행된다.

V. 결론

내장형 시스템의 요구사항이 증가함에 따라 신뢰성 높은 소프트웨어 생산이 어려워지고 있다. 그러므로 신뢰성 높은 소프트웨어 생산을 위한 내장형 시스템에 합당한 테스트 도구가 필요하다. 본 논문은 내장형 시스템 개발 환경과 동일한 호스트/타겟 구조의 테스트 도구를 제안한다. 호스트에서 작성된 테스트 케이스는 타겟에서 실행 가능한 소스로 변환하여 타겟에서 실행된다. 테스트 실행 결과는 수집하여 호스트에서 결과 보고서를 제공한다. 제안하는 도구는 테스트 케이스 작성과 테스트 드라이버 생성/컴파일, 테스트 프로그램 로딩, 테스트 프로그램 실행, 테스트 결과 수집 및 분석을 모두 제공하여 단일 도구로 테스팅을 위한 모든 작업이 가능하다. 또한, 일련의 작업을 직관적인 GUI 기반의 통합개발환경으로 제공한다. 그러므로 제안하는 도구는 테스트 비용을 절감시키고 나아가 신뢰성 높은 내장형 시스템 소프트웨어를 생산하는데 기여할 수 있다.

참고문헌

[1] B. Beizer. Software Testing Techniques. Van Nostrand Reinhold 2nd edition, 1990.

[2] http://www-306.ibm.com/software/awdtools/test/realtime.

[3] HcMillan, G. J., Design and Validation of Computer Protocols, Prentice Hall, 1991.

[4] Larsen, K, Pettersson, P., and Yi W., "UPPAAL in a Nutshell", Springer International Journal of Software Tools for Technology Transfer, 1(1+2), 1997.

[5] JUnit, http://www.junit.org.

[6] 정인상, "소프트웨어 테스팅을 위한 테스트 데이터의 자동 생성" 정보과학회지, 제19권, 제11호, 10~18쪽, 2001년 11월.

[7] K.S., Pesaran, M.H. and Shin, Y., "Testing for unit roots in heterogeneous panels", Journal of Econometrics 115, pp.53~74, 2003.

[8] Richard Hamlet, R., "Unit testing for software assurance", Computer Assureance COMPASS '89, June 1989.

[9] http://www.eclipse.org.

[10] Steve McConnell, CODE COMPLETE, 1993.

[11] E. Gamma, R. Helm, R. Johnson, J. Vlissides Design patterns: elements of reusable object-oriented software, 1995.

[12] Dongkyu Kwak, Yongyun Cho, Jaeyoung Choi, Chae-Woo Yoo, "A XML-based Testing Tool for Embedded Softwares", 2007 International Conference on Multimedia and Ubiquitous Engineering, 2007.

[13] 정창신, "XML 기반 테스트 정보를 공유하는 소프트웨어 테스팅 자동화 프레임워크의 설계" 한국컴퓨터정보학회 논문지, 제10권, 제3호, 89~99쪽, 2005년 7월

[14] 장영현, "WCBT를 이용한 대규모 자격관리 성능개선 시스템의 설계 및 구현", 한국컴퓨터정보학회논문지 제13권, 제2호, 67~78쪽, 2008년 3월

저 자 소 개

www.kci.go.kr

곽 동 규

2002년 : 서경대학교 응용수학과 학사
2004년 : 숭실대학교 대학원 컴퓨터학과 석사
2004년 ~ 현재 : 숭실대학교 대학원 컴퓨터학과 박사과정
관심분야 : 프로그래밍 언어, 컴파일러, XML, 임베디드 시스템, 유비쿼터스

유 재 우

1976년 : 숭실대학교 전자계산학과 학사
1985년 : 한국과학기술원 전산학과 박사
1983년 ~ 현재 : 숭실대학교 컴퓨터학과 교수
1986 ~ 87년, 1996 ~ 97년 : 코넬대학교 객원교수
1999 ~ 97년 : 한국정보과학회 프로그래밍언어 연구회 위원장
관심분야 : 프로그래밍 언어, 컴파일러, 인간과 컴퓨터 상호작용

조 용 윤

1995년 : 시립 인천대학교 전자계산과 학사
1998년 : 숭실대학교 대학원 전자계산학과 석사
2006년 : 숭실대학교 대학원 컴퓨터학과 박사
관심분야 : 컴파일러, 프로그래밍 언어, 프로그래밍 환경, XML, 임베디드 시스템

www.kci.go.kr