Module Testing Techniques for In-Core Protection System Software

Soo-Hyun Jeong [*], Kwon-Ki Moon, Hyeong-Soon Yim, Jae-Hee Yun
KEPCO E&C, Inc., 989-111, Daedeok-dae-ro, Yuseong-gu, Daejeon, 34057, Korea
[*]Corresponding author: hanshin8086@kepco-enc.com

1. Introduction

KEPCO E&C is participating in the project titled "Development of the Reactor Core Protection & Monitoring System for Exportation". This project is trying to apply new algorithms, tools and functions for software development and V&V. The new algorithms contain Artificial Neural Network (ANN) techniques for distribution synthesis of reactor core power. Model-based development environment (SCADE) is used to implement software. And new tools are introduced to carry out module testing. Thus, a new name "In-Core Protection System (ICOPS)" was announced instead of "Reactor Core Protection System (RCOPS)".

ICOPS is a safety I&C system of nuclear power plants. General Design Criterion (GDC) 21 of 10 CFR Part 50 Appendix A requires protection system (or safety system) be designed for high functional reliability commensurate with the safety functions to be performed [1]. Module testing is a basic and important step to ensure the reliability.

2. Methods and Results

In this section the techniques and results used to test modules for ICOPS are described.

2.1 Module testing overview

Module testing is the process of testing program components, such as methods or object classes. Components of the final product are tested independently before being assembled into larger units. Modules are tested through the use of 'test harnesses' which simulate the context into which the module will be integrated. The test harness provides a number of known inputs and measures the outputs of the module under test, which are then compared with expected values to determine if any issues exist [2, 3].

Two kinds of tests are performed for module testing; Code coverage tests on PC and module tests on Programmable Logic Controller (PLC) hardware.

2.2 Module testing tools

Code coverage analysis has been performed using a new testing tool, Code Scroll™ Controller Tester. Code coverage analysis is a kind of white-box test based on the knowledge of a module structure. Controller Tester is a useful analysis tool which generates 'test harnesses' automatically and enables testers to measure coverage metrics. And it also enables hundreds of test cases to be executed in minutes. Fig. 1 shows a workflow of Controller Tester.

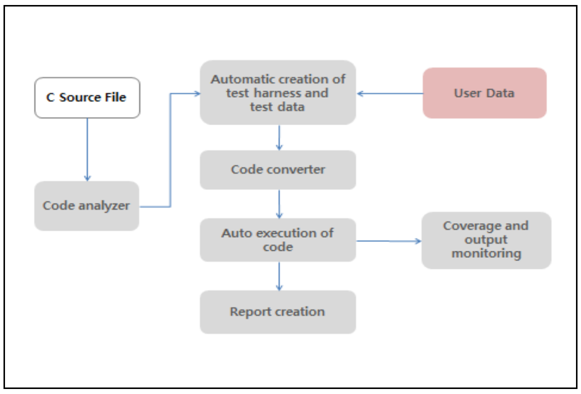


Fig. 1. A workflow of Controller Tester

POSAFE-Q PLC platform and an I/O Simulator are used for black-box module testing. Black-box testing only concerns verifying specified input against expected result and not worrying about the logic inside. Fig. 2 shows the interfaces between a PLC hardware and an I/O simulator. The I/O simulator reads a test case file and provides input data to the PLC hardware via HR-SDL (High Reliability – Safety Data Link). Then it receives output data from PLC via HR-SDL.

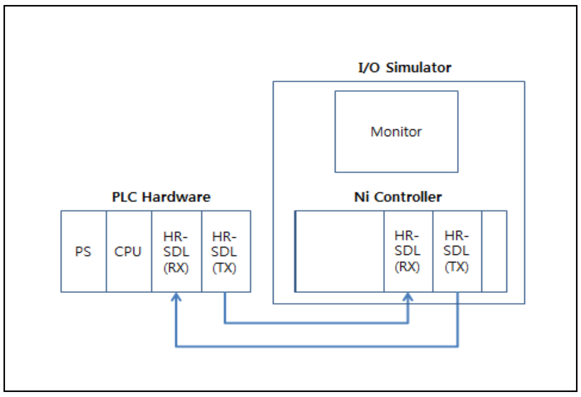


Fig. 2. Interfaces between a PLC hardware and an I/O simulator

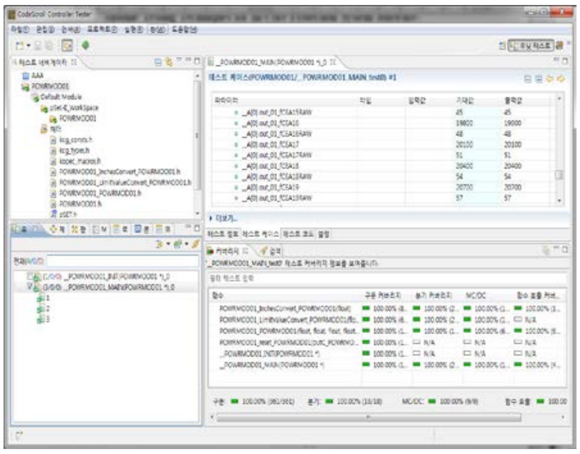
2.3 Module testing procedures

A code coverage test using Controller Tester is performed under Windows 7 environment. The test is performed according to the following procedures: 1) generate C source files using pSet II editor; 2) integrate usercode.c into main.c; 3) convert the original test case file into the Controller Tester test case file with conversion script; 4) launch Controller Tester; 5) create a project and load C source files; 6) load test case files into Controller Tester environment; 7) execute testing; 8) when completed, coverage metrics will appear in the lower right windowpane [4].

A module test on PLC is performed using the I/O simulator. The test is performed as the following procedures: 1) launch pSet II editor; 2) create a project and add a module to be tested; 3) generate input and output modules for communication with the I/O simulator; 4) compile and download to PLC; 5) connect PLC to the I/O simulator with fiber optical cables; 6) launch the Module Test program on the I/O simulator; 7) load test case files and run; 7) when completed, you can save a result file.

2.4 Module testing results

After coverage analysis, Controller Tester shows actual outputs and coverage metrics as in Fig. 3. When an actual output is found in an out-of-tolerance condition, its background turns red. If any coverage metric is less than 100%, source code which is not covered will appear in red. Covered source code appears in green.



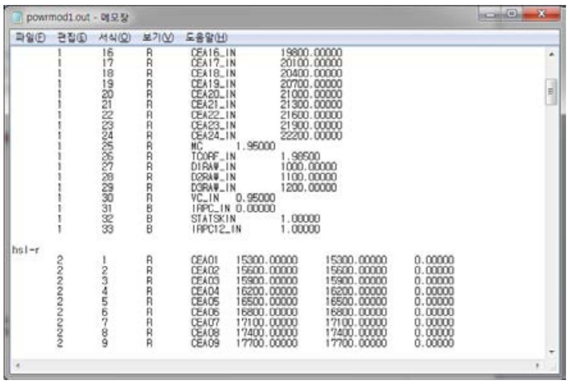
파라미터	타입	입력값	기대값	출력값
_A[0] out_01_fCEA15RAW			45	45
_A[0] out_01_fCEA16			19800	19800
_A[0] out_01_fCEA16RAW			48	48
_A[0] out_01_fCEA17			20100	20100
_A[0] out_01_fCEA17RAW			51	51
_A[0] out_01_fCEA18			20400	20400
_A[0] out_01_fCEA18RAW			54	54
_A[0] out_01_fCEA19			20700	20700
_A[0] out_01_fCEA19RAW			57	57

함수	구문 커버리지	분기 커버리지	MC/DC	함수 호출 커버...
POWRMOD01_InchesConvert_POWRMOD01(float)	🟩 100.00% (...	🟩 100.00% (2...	🟩 100.00% (...	🟩 100.00% (3...
POWRMOD01_LimitValueConvert_POWRMOD01(flo...	🟩 100.00% (6...	🟩 100.00% (2...	🟩 100.00% (...	▭ N/A
POWRMOD01_POWRMOD01(float, float, float, float...	🟩 100.00% (1...	🟩 100.00% (1...	🟩 100.00% (6...	🟩 100.00% (5...
POWRMOD01_reset_POWRMOD01(OUTC_POWRMO...	🟩 100.00% (1...	▭ N/A	▭ N/A	▭ N/A
_POWRMOD01_INIT(POWRMOD01 *)	🟩 100.00% (1...	▭ N/A	▭ N/A	▭ N/A
_POWRMOD01_MAIN(POWRMOD01 *)	🟩 100.00% (1...	🟩 100.00% (2...	🟩 100.00% (...	🟩 100.00% (4...

구문: 🟩 100.00% (361/361) 분기: 🟩 100.00% (18/18) MC/DC: 🟩 100.00% (9/9) 함수 호출: 🟩 100.00

Fig. 3. A screenshot of CodeScroll™ Controller Tester after code coverage analysis

After a module test using the I/O simulator, a result file can be saved as shown in Fig.4. The result file contains inputs, expected outputs and actual outputs from a module for each test case. When an actual output is out-of-tolerance, you can see a “FAIL” on the right of the output.



```
      1       16      R       CEA16_IN        19800.00000
      1       17      R       CEA17_IN        20100.00000
      1       18      R       CEA18_IN        20400.00000
      1       19      R       CEA19_IN        20700.00000
      1       20      R       CEA20_IN        21000.00000
      1       21      R       CEA21_IN        21300.00000
      1       22      R       CEA22_IN        21600.00000
      1       23      R       CEA23_IN        21900.00000
      1       24      R       CEA24_IN        22200.00000
      1       25      R       MC      1.95000
      1       26      R       TCORF_IN        1.98500
      1       27      R       D1RAW_IN        1000.00000
      1       28      R       D2RAW_IN        1100.00000
      1       29      R       D3RAW_IN        1200.00000
      1       30      R       VC_IN   0.95000
      1       31      B       IRPC_IN 0.00000
      1       32      B       STATSKIN        1.00000
      1       33      B       IRPC12_IN       1.00000

hsl-r
      2       1       R       CEA01   15300.00000     15300.00000     0.00000
      2       2       R       CEA02   15600.00000     15600.00000     0.00000
      2       3       R       CEA03   15900.00000     15900.00000     0.00000
      2       4       R       CEA04   16200.00000     16200.00000     0.00000
      2       5       R       CEA05   16500.00000     16500.00000     0.00000
      2       6       R       CEA06   16800.00000     16800.00000     0.00000
      2       7       R       CEA07   17100.00000     17100.00000     0.00000
      2       8       R       CEA08   17400.00000     17400.00000     0.00000
      2       9       R       CEA09   17700.00000     17700.00000     0.00000
```

Fig. 4. A result file of a module test using a PLC hardware and an I/O simulator

3. Conclusions

ICOPS is a safety I&C system. Software modules of a safety system should be tested according to related codes and standards for proving high functional reliability.

Controller Tester is a useful code coverage analysis tool for the V&V of the safety critical software. It enables testers to measure coverage metrics and executes test cases automatically in a short time. And for a black-box module test, we are using a PLC hardware and an I/O simulator with HR-SDL communication. Test results can be saved in a text file. It also shows whether the results are out-of-tolerance or not. By applying both test methods, we figured out these are effective and efficient enough for testing modules of a safety I&C system. We are also applying this design of the I/O simulator for Reactor Core Protection System of Shin-Kori 5&6 nuclear power plants units.

REFERENCES

[1] 10 CFR Part 50, “Domestic Licensing of Production and Utilization Facilities,” U.S. Nuclear Regulatory Commission, Washington, DC.
[2] Ian Sommerville, Software Engineering: 9th Edition, 2011.
[3] Nick Jenkins, “A Software Testing Primer”, 2008.
[4] Suresoft Technologies Inc., CodeScroll™ Controller Tester Manual, 2013.