Journal of the Korean Nuclear Society
Volume 27, Number 3, June 1995

《Technical Report》

Validation Testing of Safety-critical Software

Hang Bae Kim and Jai Bok Han
Korea Atomic Energy Research Institute
(Received December 21, 1994)

Safety-critical 소프트웨어의 검증시험

김항배 · 한재복
한국원자력연구소
(1994. 12. 21 접수)

Abstract

A software engineering process has been developed for the design of safety critical software for Wolsong 2/3/4 project to satisfy the requirements of the regulatory body. Among the process, this paper described the detail process of validation testing performed to ensure that the software with its hardware, developed by the design group, satisfies the requirements of the functional specification prepared by the independent functional group. To perform the test, test facility and test software were developed and actual safety system computer was connected. Three kinds of test cases, i.e., functional test, performance test and self-check test, were programmed and run to verify each functional specifications. Test failures were fedback to the design group to revise the software and test results were analyzed and documented in the report to submit to the regulatory body. The test methodology and procedure were very efficient and satisfactory to perform the systematic and automatic test. The test results were also acceptable and successful to verify the software acts as specified in the program functional specification. This methodology can be applied to the validation of other safety-critical software.

요 약

월성원자력 2, 3, 4호기 safety-critical 소프트웨어에 대한 규제기관의 요구사항을 만족시키기 위하여 소프트웨어 엔지니어링 절차가 개발되었다. 본 논문에서는 그중에서 검증시험절차에 대하여 기술하였는데, 검증시험이란 설계그룹에서 개발된 소프트웨어가 독립된 기능그룹에서 부여한 요구사항을 모두 만족하는지를 확인하는 것이다. 이 검증시험을 수행하기 위하여 시험설비와 시험용 소프트웨어가 개발되었으며, 검증시험은 기능시험, 성능시험 및 자기점검시험등으로 구성되었다. 시험결과를 분석하여, 불만족한 경우는 설계그룹에 통보되어 소프트웨어가 수정되었고, 최종결과는 보고서로 작성되어 규제기관에 제출될 것이다. 개발된 검증시험 방법과 절차는 효율적이고 성공적이었으며, 시험결과는 소프트웨어

가 기능사양서를 충분히 만족시킨다는 것을 성공적으로 검증함을 보여주었다. 본 시험방법은 다른 safety-critical 소프트웨어 검증에도 적용될 수 있을 것이다.

1. Introduction

Recently, software has been increasingly adopted to the safety critical systems in the nuclear power plant to take the advantages, such as flexible logic and easy change, over the conventional hardware and to improve the overall reliability of the plant by reducing spurious trips. But the potential concerns, such as error sensitivity, inherent to the software are always the issue for the reliability and trustworthiness of the safety critical software. [1] To overcome the concerns, various kinds of efforts have been performed to develop standards, design process and verification methodology relating to the safety critical software. A software engineering process has been successfully applied to the development of the software for the Safety Shutdown Systems for Wolsong 2/3/4 Nuclear Power Plant project.

Among the serial steps of engineering process, this paper describes the details of the validation testing performed for the Shutdown System (SDS) Programmable Digital Comparator(PDC). For the validation testing, test facility has been setup and the test software has also been developed for systematic and automatic testing. In order to perform the test, the test facility was connected to the actual PDC and the test cases were programmed and run on the test facility. After the test, test failures had been fedback to the design process to revise the software until no failure found and the test results were documented to submit to the regulatory body.

2. Requirements

2.1. Standard

There are many standards, draft standards and guidelines that provide the requirements for the assurance of safety critical software. But no single standard or guideline fully satisfies the requirement that the software is assured to be of high integrity. High integrity means software can be trusted to work dependably in critical functions, and if fails to do so, catastrophic results may occur. [2]

In 1990, Ontario Hydro and AECL CANDU jointly prepared the Standard for Software Engineering of Safety Critical Software to specify the requirements for the engineering of safety critical software used in real-time control and monitoring systems in nuclear generating station. [3] This standard was developed to incorporate the basic concept of IEC 880[4] and the practical experience gained in licensing the safety critical software for Darlington Nuclear Generating Station. [5] Though intended not to be restrictive in terms of methodologies, it establishes the requirements for certain practices such as mathematical specifications, formal verification, reliability testing, and hazard analysis. The engineering process for the development of the PDC has proceeded in accordance with this standard. In this standard, the objective of validation testing is described as to test that the executable code integrated with the target hardware and any pre-developed software meets the requirements specified in the Design Input Document (DID). The inputs to the test are the DID, code, subsystem hardware and pre-developed software. It also describes the detail requirements of validation test procedure required to cover.

2.2. Quality Project Plan [6]

The purpose of this plan is to specify the systematic approach to be used for the whole software engineering process which provides confidence that the software system conforms to the established technical and functional requirements. It applies to software

development, verification and support process, and describes the organizational criteria, standards to be followed, documentation plan, facilities and tools to be used, and informal technical reviews. It meets the requirements of CAN/CSA Q396.1.1[7] and the Standard for Software Engineering of Safety Critical Software.

2.3. Software Functional Specification

The functional requirements for the safety critical software are developed from the reactor safety analysis of the expected station response to postulated initiating events. The functional design specification resulting from the analysis is described in the Program Functional Specification(PFS). [8] It is prepared by the functional group who has a black box knowledge of the implementation of the software design and is independent from the design group. This document provides the detailed information about which trips are required, what process signals to measure, what the trip setpoints and the hysteresis should be, and how these values should change as a function of the operating state of the plant.

For simple example, the trip requirement for the Steam Generator Feedline(SGFL) Low Pressure is specified as "If any SGFL pressure signal is below the trip setpoint(3.9 MPa), open the parameter trip D/O." This specific requirement is designed into safety critical software and undergone validation test by test case described in section 5.3.1.

3. Software Design and Verification

In accordance with the PFS, software design group performs the software development and design verification. Software development process consists of software requirement definition, software design and code implementation. After finishing the design, design verification process is performed, and unit and subsystem testings are performed by the design group.

Unit testing begins at the module level with thorough debugging and testing by the software designer to detect logic errors and to test functional completeness.

Physical inputs, real time interaction with other modules, and other external stimuli are intentionally simulated rather than real.

Subsystem testing is then performed by the independent verifier to ensure that the group of programs or modules operate correctly as the software is integrated into the computer system as a whole. It also verifies the data transmission between modules within a computer.

Once the software has been frozen, it is handed over to the functional group for validation and reliability tests.

Software design and verification process are summarized as follows. (See Fig. 1)

Functional requirement	· Program functional specification (functional group)
Software design	· Software requirement definition · Software design · Code implementation
Requirement verification	· Software requirement review
Design verification	· Software design review · Systematic design verification
Code verification	· Code review · Systematic code verification · Code hazard analysis
Testing	· Unit testing · Subsystem testing · Validation testing (functional group) · Reliability testing (functional group)

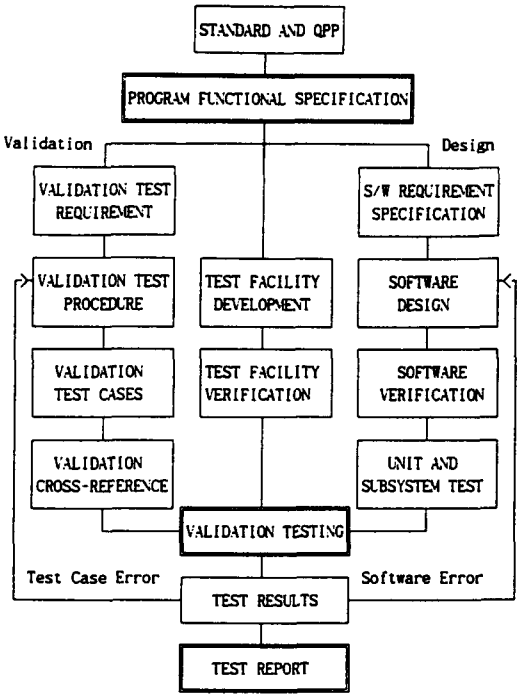

Fig. 1. Software Design and Validation Process

4. Validation Test Facility

Computer based test facility is developed to improve the speed to allow the scope of the test to be widened, reviewability by logging all test actions, repeatability and maintainability of the validation test.

4.1. Test Hardware [9]

Test computer is configured for functional testing of the PDC. It is capable of generating and reading analog voltages and digital signals and performing timing measurements on the responses of the PDC. The hardware consists of test computer, data acquisition hardware and 48V power supply. The PDC inputs are connected to the test computer outputs in order to simulate the process signals and its outputs to the test computer inputs in order to evaluate the PDC responses. (See Fig. 2)

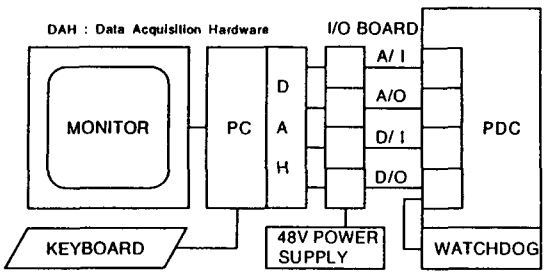

Fig. 2. Test Facility Configuration

The test computer consists of IBM-PC AT-486 compatible with built-in floating point co-processor, clock speed 33MHz, 16 MB RAM, hard-drive 200M , SVGA color monitor, 2 serial and 1 parallel port, two floppy drives and standard mouse and keyboard. The data acquisition system consists of I/O boards, buffers and mounting chassis.

4.2. Test Software [10]

A software tool has been developed using the LabVIEW application software supplied by National Instrument Corporation. It transforms the test cases into specific testing actions on the test computer as follows.

- Generate test signals via process I/O and/or data link to the PDC
- Receive the corresponding response signals from the PDC
- Compare the response signals from the PDC with the expected results set by test case
- Report pass or failure of each test

4.2.1. Instruction Set

Various kinds of instruction sets are supported by the test software and the key instructions are as follows.

CALL : Run a subroutine
CHECK : Check if a variable is in the expected state

COMPARE : Compare the observed values with the expected value
DEFINE : Assign a name to an I/O point
LET : Assign a value to a variable
RAMP : Increase or decrease test signal linearly with time
REPORT : Write a test result to the log file
SET : Set the value of an I/O point of the PDC
TITLE : Identify the test case to be run

4.2.2. Special Names and Variables

Special names and variables are defined internally by the test software and followings are the key names and variables.

TIMING_DO : assigned to the D/O whose response time to be measured.
COSNn : the time of change of state #n
TOLERANCE : the amount by which an analog signal may differ from its expected value while still being considered equal.
X : a special variable used for COMPARE instruction not to flag specified variable as failure in comparison.

5. Validation Test

The unit and subsystem tests are white box tests to detect errors in software coding and execution of the design. On the other hand, validation test is a black box test that is designed to ensure that the PDC meets the functional requirements for the design. Validation test is performed by the functional group purposely without a detailed knowledge of the design of PDC software to keep independence with the design group.

The majority of the validation test are automated and test program, called test cases, are written using a test software that is run in the test computer which simulates the environment of the plant external to the PDC. The process I/Os of the test computer are connected to the process I/Os of the actual PDC.

For Validation testing, following documents are required to prepare.

- Validation test requirement document
- Validation test procedure
- Validation cross-reference document
- Validation test case
- Validation test report

5.1. Requirements for Validation Test [11]

This document describes the functional requirements for validation test itself. It includes general requirements such as objective and responsibility, test facility hardware and software requirement, documentation requirement, test case requirement and the requirement for extension of credit to other PROMs.

5.2. Validation Test Procedure [12]

This procedure explains the test process, analysis and storage of test results, disposition of failed tests and design principles of test cases. It also describes the detail steps of each test cases and procedure for extension of credit.

Validation test is performed as following steps.

- Ensure that the desired PROMs to be tested are in the memory
- Connect the test computer to the PDC
- Start the PDC and test computer
- Follow the test instructions
- Run the test cases
- Analyze the test results

The results of automated tests are recorded in the electronic log files and the results of the manual tests are recorded in the checklist.

By analyzing the test results, if the PDC responses are not consistent with the functional requirements then the PDC software fault must be addressed to the design group through the Software Change Record (SCR). In this case, the validation process shoul d be repeated from the beginning until no failure occurs.

It is possible to extend the credit from a fully tested and credited set of PROMs to other PDC PROMs by binary comparisons of software core image.

5.3. Validation Test Case

Validation test cases are programmed using test software and run in the test facility. Validation test cases are divided into three categories : functional test, performance test and self-check test. The test cases can be chained to automatically start the next test case when a test is finished.

Test cases are prepared in accordance with the following principles.

- Completeness to test as many inputs as practical
- Clarity to make the test simple
- Modularity to use subroutine whenever possible
- Robustness to allow suitable margins for inputs and outputs not to cause spurious test failures.

5.3.1. Functional Test

Functional test is to verify specific PDC functions specified in the software functional specification and consists of following tests.

Single trip test is to test immediate normal and irrational trips on every instrument loop.

Delayed trip test is to test individual delayed trips on the parameters such as Primary Heat Transport (PHT) high pressure and PHT low core differential pressure.

Dual trip test is to test the independence of trip parameters and of loops within a parameter.

Power calculation test is to test the calculation of the weighted average power and compensated ion chamber linear power.

Parameter trip conditioning test is to test that the immediate parameter trips are conditioned out at certain low power.

Setpoint modification test is to ensure that trip setpoints dependent on power or pump mode are set to the specified values.

For example, single trip test is programmed in Table 1 and 2.

5.3.2. Performance Test

Performance test is to ensure that timing requirements are met as specified in the functional specification and consists of following tests.

Single trip timing test is to ensure that an analog signal past the immediate trip setpoint on any given instrument loop will result in a parameter trip D/O being opened within the required time.

Table 1.

```
TITLE SINGLE -Test of single immediate trip
;
CALL DEFINE                                  ; Define I/O names
CALL INITIAL                                 ; Initialize system
;
; Steam Generator Feedline Low Pressure Parameter (SGFLP)
;
DEFINE AI = AI_SGFLP                         ; Define test AI as SGFLP loop
DEFINE DO_PARAMETER = DO_SGFLP               ; Define relay logic D/O
LET LOW_SP = 3.900                           ; Specify low setpoint
LET NOMINAL = 4.700                          ; Specify nominal value
LET AI_TOL = 0.026                           ; Set AI tolerance
CALL SINGLE.SUB                              ; Call subroutine
TESTEND
```

Table 2.

```
TITLE SINGLE.SUB-Trip test subroutine for specified loop
;
SET AI=NOMINAL                          ; Set AI to nominal value and
SET DO_PARAM=CLOSED                     ; ensure that everything is normal
COMPARE
;
SET AI=LOW_SP+AI_TOL                    ; Set AI just above low setpoint
SET DO_PARAM=CLOSED                     ; Expect no parameter trip
COMPARE                                 ; Compare PDC output with expected value
;
SET AI=LOW_SP-AI_TOL                    ; Set AI just below low setpoint
SET DO_PARAM=OPEN                       ; Expect parameter trip
COMPARE
;
SET AI=NOMINAL                          ; Set AI back to nominal value
SET DO_PARAM=CLOSED                     ; Expect no parameter trip
COMPARE
;
EXIT
```

Multiple trip timing test is to ensure that the parameter trip D/O opening times are within specified limits when simultaneous trips occur on parameters.

Delayed trip timing test is to ensure that the specified time requirements for the trip delay and minimum trip duration are satisfied.

Typical functional test is explained as follows using single trip timing test.

- Call DEFINE and INITIAL subroutine to define I/O names and initialize the system
- Identify the A/I and D/O parameter for the instrument loop
- Set setpoint and maximum specified trip time
- Set A/I close but not exceeding setpoint and ensure no trip occurs. Set A/I exceeding setpoint and wait until trip occurs, then log the trip time (continue 100 times)
- Set A/I back to nominal value and calculate the average trip time and log maximum trip time. Check if the maximum value exceeds the specified maximum value.
- Continue other parameters.

5.3.3. Self-check and Error Handling Test

These tests simulate hardware failures so that the PDC error detection and handling capabilities have been verified during unit and subsystem testing.

Incorrect PROM insertion test is to manually check the safeguards built into the PDC PROMs to ensure that PROMs intended for a particular PDC cannot run undetected to a different PDC.

Watchdog trip test is to simulate the watchdog test pushbutton being closed which instructs the PDC to stop toggling the watchdog update D/O and allow the watchdog to time out.

Analog and digital I/O wraparound test is to verify that analog and digital I/O faults are detected by the wraparound self-checks.

Invalid or irrational input test is to ensure that the PDC responds as specified to signals beyond irrational alarm limits and to high signal spreads.

Power failure and restart test is to ensure that the

safest possible initial values specified in the functional specification are selected for PDC outputs if no signals can be read at start-up.

5.4. Validation Cross Reference [13]

The validation cross reference document is intended to provide a cross reference between PDC functional requirements and the test cases programmed to verify those requirements. Each functional requirements specified in the PFS are confirmed by matching the test cases one by one to allow a reviewer to confirm easily that every requirements have been thoroughly tested.

5.5. Validation Test Report [14]

After the validation testing, test report is prepared to describe the test results including the status of test software and hardware, test methodology, log files produced automatically by test cases during testing and summary results. If there were any failures and/or problems during testing, Software Change Request (SCR) was prepared and sent to the responsible group for correction.

6. Conclusion

This paper described the detail process of validation testing performed for the actual project. Though the software testing could not be perfect inherently, the tests appeared to be acceptable and successful to test that the safety critical software, designed by the design group, satisfies all the functional requirements specified by the functional group. The test facility and software developed for the project was very efficient to test the software systematically and automatically.

It is expected that this well structured validation testing process can be applied for the testing of other safety critical softwares. And also it is planned to introduce our experience of reliability testing, which is the next process to the validation testing.

References

1. D.L. Parnas, A.J. van Schouwen, S.P. Kwan, "Evaluation of Safety Critical Software", *Communication of the ACM*, Vol. **33**, No. 6, June 1990, pp. 636−648.
2. D.R. Wallace, D. R. Kuhn, L. Beltracchi, "Standard for High-Integrity Software", Nuclear Safety, Vol. **35**, No. 1, January-June 1994, pp. 86−97.
3. Standard for Software Engineering of Safety Critical Software, rev.0, December 1990.
4. IEC 880 "Software for Computers in the Safety Systems of Nuclear Power Stations."
5. J.R. Popovic, G.J. Hinton, "CANDU Computerized Safety System", EPRI Conference, Advanced Computer Technology for the Power Industry, Scottsdale, Arizona, USA, December 1989.
6. 86-68350-QPP-001, "Quality Project Plan, Programmable Digital Comparator Shutdown System Number 2", rev.1, August 1994.
7. CAN/CSA-Q396. 1.1-89 "Quality Assurance Program for the Development of Software used in Critical Application"
8. 86-68300-PFS-000, "Program Functional Specification, Shutdown System Number 2 Programmable Digital Comparators"
9. 86-68000-220-101, "Hardware Manual, Validation and Reliability Test Manual", rev.0, August 1994.
10. 86-68000-220-102, "User's Manual, Test Language Interpreter", rev.1, August 1994.
11. 86-68000-DR-001, "Design Requirements, Shutdown System Numbers One and Two Validation Test Requirements", rev.0, August 1993.
12. 86-68300-TP-001, "Test Procedure, Shutdown System Number Two Validation Test Procedure", rev.1, August 1994.
13. 86-68300-220-011, "Wolsong 2, 3 & 4 SDS2 Validation Cross-Reference", rev.0, August 1994.
14. 86-68300-TR-000, "Test Report, SDS2 Preliminary Validation Test Report", rev.0, September 1994.