2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

일체형원자로 MMIS 설계에 적용을 위한 소프트웨어 시험 계획

서용석*, 허섭*, 박근옥*, 이종복*, 김동훈*
*한국원자력연구소 일체형원자로MMIS설계기술개발분야
e-mail:yssuh@kaeri.re.kr

A Software Testing Plan for Integral Reactor MMIS Design

Yong-Suk Suh*, Seop Hur*, Geun-Ok Park*, Jong-Bok Lee*, Dong-Hoon Kim*
*Dept. of SMART MMIS Design Technology Development,
Korea Atomic Energy Research Institute

요약

소프트웨어 개발자로부터 독립된 소프트웨어 시험자가 수행하는 소프트웨어 시험은 소프트웨어의 안전성 향상을 위해 필요하다. 컴퓨터기반의 디지틀시스템으로 설계되는 일체형원자로 MMIS에 적용하기 위한 소프트웨어 시험 계획을 개발할 필요가 있다. 본 논문은 소프트웨어 시험 계획을 소프트웨어 시험 조직 구성, 시험 문서, 시험 절차, 시험 방법을 중심으로 제시한다. 소프트웨어 시험 방법은 원시코드 정적분석과 동적시험을 구분하여 기술한다. 본 논문에서 제시된 소프트웨어 시험 계획은 원자력규제기관에서 요구하는 소프트웨어 시험 요구사항을 만족한다. 본 논문을 통해 제시된 소프트웨어 시험 계획을 일체형원자로 MMIS 소프트웨어 개발 시 적용하여 소프트웨어 고장율 데이터를 수집할 예정이다.

1. 서론

한국원자력연구소(KAERI)는 90년대 중반부터 전력생산과 담수화 목적을 위해 330MWt급의 열출력을 생산할 수 있는 일체형원자로인 SMART(System-integrated Modular Advanced Reactor)를 설계해 오고 있다. SMART MMIS(Man-Machine Interface System)는 원자로로부터 일반인에게 방사능이 누출되지 않도록 원자로를 보호, 제어, 감시하는 컴퓨터기반의 제어시스템이다[1]. 우리나라 기존의 원자력발전소 제어시스템은 아나로그기반으로 설계되었으나, 최근에는 디지틀기반으로 설계하려는 추세이다. 그러나 원자력규제기관(KINS)은 디지틀기반의 제어시스템을 원자력발전소에 적용하는 것에 대해 아직까지 완전한 동의를 보이고 있지 않다. 그 이유 중의 하나가 디지틀시스템을 동작시키는 소프트웨어가 잠재적으로 내포하고 있는 불안전성 때문이다. 소프트웨어의 고장 원인과 그에 대한 대처가 불완전하다는 것이 그들의 입장이다.

소프트웨어의 안전성(reliability)을 향상시키는 방법으로써 규범화된 소프트웨어 수명주기 수립과 그에 따른 개발, 독립적인 검증 및 확인 수행, 정형화 기법의 적용, 소프트웨어 안전성 및 신뢰도 분석 수행, 독립적인 소프트웨어 시험 수행 등을 들 수 있다[2]. 이 가운데 본 논문은 원자력규제기관의 소프트웨어 시험에 대한 요구사항을 만족하고, 고품질의 SMART MMIS 소프트웨어를 생산하기 위한 소프트웨어 시험에 대해 관심을 둔다. 소프트웨어 시험은 소프트웨어 검증 및 확인 작업과 밀접한 관계가 있다. IEEE-1012 소프트웨어 표준에 의하면 소프트웨어 검증 및 확인 작업이 효율적으로 수행되기 위해서는 요건, 설계, 코드와 같은 생산물이 추적성(traceability)과 그에 따른 시험성(testability)이 가능한 내용으로 기술되어야 함을 언급하고 있다[3]. 그러므로 소프트웨어 시험의 목표는 추적성과 시험성이 가능한 요건, 설계, 소프트웨어의 내용 가운데 잠재되어 있는 오류를 소프트웨어 시험을 통해 찾아내는 것으로 설정하여야 한다.

본 논문의 목적은 위와 같은 목표를 만족하기 위해 SMART MMIS의 독립적인 소프트웨어 시험 그룹

이 수립하여야 할 소프트웨어 시험 계획을 제시하기 위함이다.

2. 소프트웨어 시험 조직

소프트웨어 시험 조직은 개발 조직으로부터 예산 및 관리측면에서 완전히 독립적으로 구성한다. 이는 시험활동의 자유성을 보장하기 위함이다. 소프트웨어 시험 조직은 최소한 시험관리자, 시험분석가, 시험수행원으로 구성한다. 시험관리자는 시험계획을 설정하며, 소프트웨어 품질보증 그룹, 개발 그룹과 시험 그룹간의 조정역할을 담당한다. 개발 그룹과 시험 그룹간의 상충된 의견이 발생한 경우 시험관리자가 이를 중재하고 일정을 재조정한다. 시험분석가는 테스트 케이스(test case)를 작성한다. 또한, 시험수행원이 수행한 시험결과를 분석 및 평가하고 시험보고서를 작성한다. 시험수행원은 테스트 케이스의 내용에 따라 시험을 수행하며 완료된 테스트 케이스를 시험분석가에게 제출한다.

3. 소프트웨어 시험 문서

주요 소프트웨어 시험 문서로써 소프트웨어 시험 계획서, 테스트 케이스, 시험 보고서를 생산한다. 소프트웨어 시험 계획서는 소프트웨어 시험에 대한 전반적인 계획을 수립하는 문서로써 활용되며, 소프트웨어 시험 전체의 윤곽이 기술된다. 소프트웨어를 서로 독립적으로 분리할 수 있는 경우, 이들에 대한 시험 계획은 각각 작성된다. 예를 들어, 소프트웨어 구성 가운데 데이터 입력부와 MMI는 서로 독립적으로 시험이 수행될 수 있는 부분이 있으므로 이들은 각각 별도의 시험 계획서가 작성될 수 있다. 테스트 케이스는 시험환경과 신호입력방법 및 결과출력방법 그리고 예상 결과가 기술된다. 입력방법으로써 모사 소프트웨어에 의한 모사신호 발생방법 또는 하드웨어장비에 의한 입력신호 생성방법을 택한다. 가능한 현장과 동일한 환경을 만들어 모사신호를 발생하는 방법을 추구한다. 이것은 시스템이 현장에 설치되면 입력신호의 값이 시험 시 예상했던 것 이상으로 변화되기 때문이다. 계산결과를 확인하는 방법으로써 출력값을 프린터로 출력하는 방법, 데이터파일 형태로 작성하는 방법 또는 하드웨어장비를 통해 출력상태를 확인하는 방법을 택한다. 시험 보고서는 소프트웨어 시험의 성공 또는 실패를 명시하며, 그 이유에 대해서도 기술한다.

4. 소프트웨어 시험 절차

시험그룹은 개발그룹으로부터 계통설계요건서, 소프트웨어설계요건서, 소프트웨어설계서, 운전절차서와 소프트웨어 생산물인 원시코드를 반드시 획득하여 시험계획과 테스트 케이스를 작성한다. 원시코드를 획득할 수 없는 상용 소프트웨어인 경우, 시험그룹은 계통설계요건서와 상용 소프트웨어 기능요건서를 획득하여 시험계획과 테스트 케이스를 작성한다. 시험은 테스트 케이스에 따라 수행되며, 모든 테스트 케이스를 성공적으로 완료하였을 경우, 그 소프트웨어를 시험관점에서 인수한다. 시험과정 중에 소프트웨어의 오류가 발견되었을 경우, 시험자는 그 오류가 소프트웨어의 결함 때문에 발생된 것인지 또는 테스트 케이스의 오류에 의해 발생된 것인지를 분석한다. 테스트 케이스가 잘못된 경우, 테스트 케이스를 수정한 후 시험을 다시 수행한다. 소프트웨어의 결함이라고 판단되면 소프트웨어 오류 보고서를 작성하여야 한다. 소프트웨어 오류 보고서는 개발자에게 전달되며 시험그룹은 잔여 시험에 대해 계속적인 수행여부를 결정하여야 한다. 이에 대한 기준은 발생된 오류가 다른 시험을 수행하는데 있어서 얼마만큼의 영향을 줄 것인가에 달려있다. 발생된 오류가 다른 시험에 대해 독립적이라면 다른 시험을 계속할 수 있지만 그렇지 못한 경우에는 시험은 중단되어야 한다. 오류가 발생된 소프트웨어가 개발자에 의해 다시 수정되고 그에 따른 소프트웨어 변경 보고서가 시험자에게 전달될 때까지 시험은 중단된다.

5. 소프트웨어 시험 방법

소프트웨어 시험은 크게 정적분석(static analysis)과 동적시험(dynamic testing)으로 구분하여 수행한다. 소프트웨어를 실행하며 수행되는 동적시험은 크게 white-box시험과 black-box시험으로 구분하여 수행한다.

5.1 소프트웨어 정적분석

소프트웨어 정적분석은 원시코드를 실행시키지 않고 코드 내부를 line-by-line 시각적으로 읽어가면서 검토하는 방식(peer review method)이다. 이는 소프트웨어 개발자가 발견할 수 없는 오류를 독립 검토를 통해 발견하기 위함이다. 정적분석을 수행하기 위해서는 소프트웨어요건서, 소프트웨어설계서, 원시코드가 반드시 제공되어야 한다. 정적분석에서 검토할 내용은 다음과 같다: 소프트웨어 코딩의 일관

성, 소프트웨어 개발규범의 준수여부, 소프트웨어 설계서와 코드의 로직 일치성, 사용되지 않는 변수의 존재여부, 잘못된 계산방식, 잘못된 구문(syntax)의 사용 등이다.

본 논문에서 선정한 정적분석 방법은 Fagan이 제시한 3단계 검토방식을 채택한다[4]. 첫 번째 단계는 소프트웨어요건서를 검토한다. 소프트웨어요건에 상위요건에서 요구하는 기능이 모두 반영되었는가를 검토한다. 두 번째 단계는 소프트웨어설계서를 검토한다. 소프트웨어설계서에는 구조, 로직, 데이터 등이 기술되어 있으므로 이들이 소프트웨어요건을 모두 표현하였는가를 검토한다. 세 번째 단계는 원시코드를 검토한다. 원시코드가 표준화된 방식으로 작성되었는지 그리고 로직이 요건과 일치하는지 등에 대해 검토한다. 검토구성원은 최소한 설계자, 코드개발자, 검토자, 중재자로 구성되어야 한다. 먼저, 설계자는 전체적인 설계개념, 기능, 요건, 인터페이스 등을 설명한다. 코드개발자는 자신이 개발한 소프트웨어를 순차적으로 설명해 나간다. 검토자는 소프트웨어 개발에 입력으로 사용되었던 문서들을 사전에 완전히 숙지한 상태에서 설계자 및 코드개발자가 설명한 내용에 대해 질문을 던진다. 코드개발자와 검토자간에 논쟁이 벌어질 수 있으므로 이를 중재자가 조절한다. 검토가 진행되는 과정에서 중재자의 역할이 가장 중요하며, 검토의 진행여부를 결정할 수 있는 권한을 갖는다. 검토시간은 1회에 2시간을 초과하지 않는다. 검토시간이 2시간을 초과하게 되면 검토자의 집중력 저하로 인해 검토효율이 떨어진다. 검토자는 검토리스트를 작성하며, 검토리스트에 입각하여 요건, 설계, 구현이 되었는가를 확인한다.

소프트웨어 정적분석은 소프트웨어 개발이 완료된 후 수행되는 것이 아니라 소프트웨어 개발과정 중에 소프트웨어 시험그룹과 개발그룹이 함께 수행하는 것이 효과적이며, 검토시기는 소프트웨어 개발일정에서 결정한다.

5.2 소프트웨어 white-box시험

white-box시험은 원시코드를 실행하면서 코드의 내부구조를 시험하는 방법으로써 구조적 시험(structured testing)이라고도 한다. white-box시험을 위해서는 원시코드가 반드시 제공되어야 하며, 원시코드를 구체적으로 설명한 소프트웨어요건서 및 설계서가 제공되어야 한다. white-box시험은 경로시험(path testing)이 핵심이며 경로시험을 수행하면서 코드의 제어흐름(control flow)과 데이터흐름(data flow)의 정확성이 요건과 부합되는지를 확인한다.

경로시험 가운데 유일 경로시험(unique path testing 또는 exhaustive path testing이라고도 함)의 수행에 대한 필요성을 고려하여야 한다. 유일 경로시험의 예로써, 2개의 if문에 의해 4개의 분기가 형성되는 문장이 20번의 loop 순환문 안에 있는 프로그램인 경우, 4^{20}(약 100만개 이상)개의 유일 경로(unique path)가 형성된다. 이와 같은 유일 경로를 시험하기 위해서는 100만개 이상의 테스트 케이스를 만들어야 하는데 이는 현실적으로 많은 시간과 인력이 소모될 것이다. 따라서, 소프트웨어의 중요성에 따라 유일 경로시험 수행여부가 결정되어야 한다. 모든 유일한 경로에 대해서 발생되는 상황을 반드시 확인해야 하는 요건이 주어진 소프트웨어에 대해서는 유일 경로시험이 수행되어야 할 것이다. 유일 경로시험은 많은 인력과 시간을 필요로 하는 문제점과 시험의 효용성에 대한 의문을 갖고 있기 때문에 배제되는 경우가 많다.

일반적으로 white-box시험은 다음과 같은 세 가지 시험방법으로써 구분된다: 제어흐름시험, 순환시험, 데이터흐름시험이다. 제어흐름시험은 프로그램의 실행흐름을 확인하는 시험으로써 시험을 통해 모든 실행경로가 적어도 한 번 이상 실행이 되었다면 제어흐름시험은 완료된 것으로 간주한다. 시험방법은 프로그램의 흐름도를 도식하고 각 흐름을 결정하는 변수가 가질 수 있는 값에 대한 변화를 테스트 케이스에 작성하여 프로그램이 실행되는 동안 입력되도록 한다. 각 흐름이 수행되었음을 알리는 출력값을 결정함으로써 출력된 결과를 분석하여 시험의 완료여부를 판단한다. 순환시험은 순환이 시작되고 종료되는 경계값 부근에서 로직오류가 발생할 가능성이 많다는 경험으로부터 비롯된다. 프로그램 개발자는 순환을 결정하는 변수의 초기값과 종료값이 비정상적으로 결정될 경우 순환문은 어떻게 반응할 것인가, 또는 초기값과 종료값이 정상적이라 하더라도 순환문이 요건에서 제시한 내용대로 수행하는지에 대한 시험을 간과할 우려가 있는데, 이를 순환시험을 통해서 확인한다. 시험방법은 순환을 결정하는 변수의 초기값에 대한 최소값(초기값-1으로 결정)과 최대값(초기값+1으로 결정)을 테스트 케이스에 작성한다. 초기값의 최소값을 시험하고자 하는 순환문의 초기

값으로 하고 최대값을 순환문의 종료값으로 하여 프로그램을 실행한다. 순환문에서 출력된 결과를 분석하여 로직오류의 여부를 판단한다. 종료값에 대해서도 같은 방식으로 시험한다. 데이터흐름시험은 프로그램내의 변수가 그 시점에서 올바로 사용(변수의 정당성이라고 함)되고 있는지를 확인하는 시험이다. 특히, 파일 입출력과 관련된 변수의 사용에서 변수 참조 오류가 발생할 가능성이 많다. 예를 들어, 파일이 열리지(open) 않았거나 또는 닫은 후에 파일을 읽거나 쓰려는 경우이다. 시험방법은 프로그램 흐름마다 변수의 사용에 대한 정당성을 확인한다. 변수의 정당성은 테스트 케이스에 정의되어야 한다.

white-box시험은 소프트웨어 내부에서 사용되는 변수와 제어흐름이 복잡한 경우, 이에 대한 테스트 케이스를 손으로 작성하고 수동으로 시험을 수행한다는 것은 무리이다. 따라서, 소프트웨어의 복잡도에 따라 소프트웨어 역공학도구(reverse engineering tool) 또는 자동 테스트 케이스 생성도구, 시험분석도구 등을 사용하는 것이 효과적이다.

5.3 소프트웨어 black-box시험

black-box시험은 소프트웨어의 내부구조에 대해서는 전혀 관심을 두지 않고, 소프트웨어가 외부적으로 반응하는 결과가 소프트웨어요건에 부합되는가를 확인하는 시험이다. black-box시험의 핵심은 기능적 시험(functional testing)이다. 기능적 시험은 소프트웨어가 동작한 후 요구된 시간에 요구된 결과를 정확히 출력하는지 확인하는 시험이다. 기능적 시험을 수행하기 위해서는 소프트웨어 설계문서 및 운전절차서로부터 외부적 요소(운전원 행위, 센서, 작동기, 디스크, 프린터, 네트웍 등)의 행위를 분석하여 시험하고자 하는 기능을 체계적으로 도출한다. 도출된 기능은 계층적으로 그룹화 되어야 시험이 효과적으로 수행될 수 있다. 일반적으로 프로그램 개발자는 프로그램 개발 시 정상적인 조건과 상황에서 자신의 프로그램이 반응해야할 결과에 중점을 두고 개발하는 경향이 있다. 프로그램 개발자는 비정상 상황 및 상태에서 자신의 프로그램이 반응해야 할 경우를 소홀히 하게되는데, black-box시험은 비정상 조건 및 상황을 유발하는 테스트 케이스를 작성하여 소프트웨어를 시험하여야 한다.

black-box시험은 크게 입출력변수시험과 트랜잭션시험으로 구분할 수 있다. 입출력변수시험은 소프트웨어의 외부입력으로 정의된 모든 변수의 입력값과 입력으로 인해 영향을 받는 외부출력의 결과값을 테스트 케이스로 작성하여 수행하는 시험이다. 하나의 입력값이 가질 수 있는 영역이 구분되고 각 영역별로 출력값이 다를 경우 테스트 케이스에는 각 영역이 분리된 입력값이 기술되어야 한다. 트랜잭션시험은 운전원 행위 관점에서 수행하는 시험이다. 운전원 행위는 소프트웨어의 외부 입력이 된다. 하나의 외부 입력에 대해 다수의 출력이 일어날 수 있다. 이러한 외부 입력과 소프트웨어 출력은 테스트 케이스에 명시되어야 한다.

6. 결론

본 논문은 SMART MMIS 설계에 적용하기 위한 소프트웨어 시험 계획을 제시하였다. 소프트웨어 시험 계획으로써 소프트웨어 시험 조직설정, 소프트웨어 시험과 관련된 문서생산, 소프트웨어 시험 절차수립에 대해 제시하였다. 소프트웨어 시험 방법을 소프트웨어 정적분석과 동적시험으로 분류하였으며, 소프트웨어 동적시험을 white-box 시험과 black-box 시험으로 구분하여 제시하였다.

본 논문에서 설정한 소프트웨어 시험 계획을 SMART MMIS 소프트웨어 개발에 적용하고 이행함으로써 원자력규제기관의 요구사항을 만족하고, 고품질의 SMART MMIS 소프트웨어를 생산할 수 있을 것이다. 또한, 본 논문에서 설정한 개념을 바탕으로 소프트웨어 시험 지침 및 절차를 구체화 하여 SMART MMIS 시험검증설비 개발 시 적용할 것이다.

Acknowledgement

본 연구는 과학기술부의 원자력연구개발사업 일환으로 수행되었음.

참고문헌

[1] KAERI/RR-1901/98, "일체형원자로 MMIS설계기술개발", 한국원자력연구소, 1999. 3.

[2] 서용석 외, "SMART MMIS 설계에 적용을 위한 소프트웨어 개발 개념", '00춘계원자력학회 학술발표회, 2000. 5.

[3] IEEE Std 1012-1998, "IEEE Standard for Verification and Validation", IEEE, 1998.

[4] M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development", IBM System Journal, Vol. 15, No. 3, 1976, pp. 182-211.