원자력 내장형 시스템의 테스팅 방안

성아영* 최병주* 이나영** 황일순**
*이화여자대학교 컴퓨터학과, **서울대학교 원자력공학과
{wishmoon, bjchoi}@ewha.ac.kr, {grasia2, hisline}@snu.ac.kr

Testing Methodology of Embedded System in Nuclear Power

Ahyoung Sung* Byoungju Choi* Nayoung Lee** Ilsoon Hwang**
*Dept. of Computer Science & Engineering, Ewha Womans University
**Dept. of Nuclear Engineering, Seoul National University

요 약

원전보호계통(RPS; Reactor Protection System)은 사고 시 치명적 피해를 입을 수 있다는 점에서 안전에 대한 중요도가 가장 높은 Safety 1E class로 분류되며, 이러한 보호계통을 디지털라이즈 하는데 있어서 높은 신뢰도에 대한 보장이 필요하다. 따라서 본 논문에서는 DPPS(Digital Plant Protection System) 내에서 작동하는 내장형 소프트웨어에 높은 신뢰성을 보장하기 위한 테스팅 방법론을 제시하고자 한다. DPPS에서 작동하는 내장형 소프트웨어를 테스트 하기 위한 방법은 크게 두 가지로 나누어진다. 첫번째 단계는 절차중심의 프로그램에서 객체를 추출하고 이를 이용하여 클래스를 추출하는 재공학의 단계이다. 두 번째 단계는 이러한 클래스들을 이용하여 레벨별 테스팅을 수행하기 위한 테스트 아이템을 추출하고, 추출된 테스트 아이템을 이용하여 테스트 케이스를 선정하는 단계이다. 이렇게 각 레벨별로 선정된 테스트 케이스를 이용하여 단위 테스팅, 통합 테스팅, 시스템 테스팅 이렇게 3단계의 레벨별 테스팅을 수행한다.

1. 서론

원자력 발전소, 항공제어 시스템, 국방관련 시스템처럼 사고의 피해가 치명적인 시스템을 Safety-Critical System[1]이라고 하며, 이러한 시스템에서 사용되는 소프트웨어는 안전성 검증을 통해 높은 신뢰도를 보장 받아야 한다. Safety-Critical System 중 하나로 원자력 발전소의 RPS (Reactor Protection System)의 경우, 원전 안전계통 중에서도 Safety 1E class로 안전에 대한 치명도가 가장 높다[2]. 따라서 이러한 시스템에서 사용되는 소프트웨어의 신뢰도 문제가 중요하게 부각된다. 이러한 신뢰도를 정량화하기 위해서 소프트웨어에 대한 테스트가 필요하며 고신뢰도의 보장에 필요한 많은 수의 테스트를 수행하기 위해서는 많은 시간과 노력이 요구된다.

특히 이러한 소프트웨어는 특정 기능을 수행하기 위해 하드웨어와 조합되어 하나의 시스템으로 나타내지며, 이런 시스템에서 사용되는 소프트웨어를 내장형 소프트웨어라고 한다. [3] 내장형 소프트웨어의 경우 다른 소프트웨어에 비해 수정하는 것이 용이하지 않으며, Real-time-ness적인 특성과 Robust-ness적인 특성이 있는데, 특히 안전도가 높은 원전 보호계통 시스템 내에 들어갈 내장형 소프트웨어이기 때문에, Robust-ness가 높게 요구된다.[4,5] Robust-ness가 높게 나타나기 위해서는 높은 Fault-tolerance가 요구되며 이를 위해서는 철저한 테스팅을 통해 안전성과 신뢰성을 보장 받아야 한다.

본 논문에서는 원자력 발전소에서 사용한 기존의 아날로그 방식의 RPS를 기기 낙후에 따른 부품 조달의 어려움과 한번 구현된 시스템에 대한 수정의 어려움 때문에, 이런 안전에 대한 중요도가 가장 높은 원전보호계통 시스템을 디지털화 하려는데 있어, 이 시스템에서 사용되는 소프트웨어의 안전성 검증을 위해 체계적이고 효율적인 테스트 방법론을 제안 하고자 한다

본 논문의 구성은 다음과 같다. 2장에서는 원전 보호계통 시스템인 DPPS(Digital Plant Protection System)[2]의 등장 배경과 설명을, 3장에서는 그러한 시스템을 테스트하기 위한 테스트 방법론에 대해서 기술하며, 4장에서는 결론 및 향후 연구 과제를 제시한다.

2. DPPS(Digital Plant Protection System)

원자력 발전소의 RPS는 원전에 문제가 발생했을 때, 원전을 안전하게 정지시키기 위한 장치이다. 기존의 원전에는 릴레이 로직에 의해 구현된 아날로그 방식을 사용하였으나, 시스템의 낙후에 따른 부품의 조달이 어려움과 한 번 구현된 시스템은 수정이 어렵기 때문에 신호의 drift 등의 문제에 대해 디지털 방식으로의 개발이 대두되었다. 그렇지만, 원자력 분야에 신기술을 적용하기 위해서는 안전에 관한 까다로운 규제를 만족시켜야 한다. 특히 RPS은 원전 안전 계통 중에서도 safety 1E class로 안전에 대한 치명도가 가장 높다. 따라서 소프트웨어의 신뢰도 문제가 중요하게 부각된다.

현재, 하드웨어 부분에 대한 신뢰도 분석 방법론은 수립되어 있지만, 소프트웨어에 대한 방법론은 아직 확립되지 못하였다. 규제에서도 소프트웨어의 개발 방법론으로 정형기법을 이용한 V&V(Verification and Validation)의 수행이 권고되고 있으며, 소프트웨어의 공통모드 고장, 종속적인 고장, 검출되지 않은 고장 등에 대해 신뢰도 분석에 포함시킬 것을 요구하고 있지만, 디지털 안전계통의 적용 경험이 거의 없어 방

법론적인 면에 대한 언급은 없다. 따라서 입증된 기술 중 원전 DPPS에 적합한 방법론을 개발하여 테스트를 수행하고 이 결과물 신뢰도 분석에 활용할 수 있도록 하고자 본 연구를 수행한다.

DPPS는 아래의 그림 1처럼 4개의 분리된 채널 장치[2]로 구성되어져 있다. 각 채널 장치는 CPC(Core Protection Calculation) cabinet , PPC(Plant Protection Calculator) cabinet 과 인터페이스로 구성되어져 있다. PPC는 다시 Bistable processor , Coincidence processor, Interface and Test processor로 구성되어져 있다.

보호 계통 케비넷의 Bistable processor는 10개의 아날로그 신호와 2개의 디지털 신호를 입력으로 받아 기존에 설정되어 있는 설정치와 비교하여, 입력 받은 값들이 설정치보다 높아지거나 낮아지게 되면 정지 신호를 발생시킨다. 이러한 정지신호는 2/4 Coincidence processor로 입력으로 들어가게 되며, 이때 Coincidence processor는 독립적인 4개 채널의 Bistable processor로부터 나오는 출력을 모두 비교하여, 두 개 이상의 채널에서 정지신호가 발생되면, Coincidence 프로세스는 원자로 정지와 공학적 안전설비계열을 동작하기 위한 트립 개시신호를 트립 개시 프로세서에 제공한다. 원자로 정지는 제어봉 구동장치의 코일로 가는 전원을 차단함으로써 제어봉을 노심내로 자중 낙하 시켜 달성되고, 공학적 안전설비는 신호 및 계열별로 할당된 펌프나 밸브의 작동을 위한 신호를 발생시킴으로써 동작된다[2].

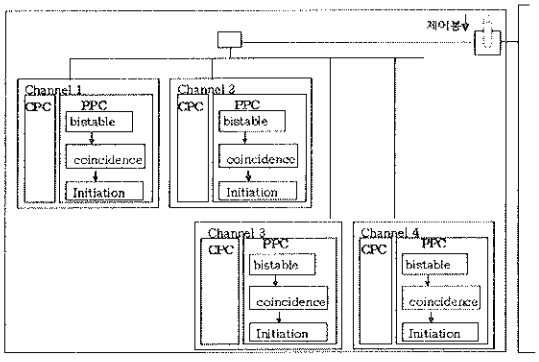

그림 1. DPPS의 구조

3. 테스팅 방안

DPPS에서 사용되는 소프트웨어는 절차중심의 언어인 C로 구성되어져 있으며, 이러한 소프트웨어는 해당 시스템 안에 내장되어 있는 내장형 소프트웨어이다. 내장형 소프트웨어는 시스템에 내장되는 컴퓨터 상에서 동작하는 소프트웨어로, 시스템의 하드웨어로부터 입력을 받아 적절한 제어 신호 발생 기능을 수행하는 소프트웨어이다[3,5].

본 연구에서 제안하는 테스팅 방법은 오른쪽의 그림 2와 같이 크게 2가지의 단계로 나눌 수 있다. 먼저 재공학 단계로, 원래의 절차 지향적인 프로그램에서 객체를 추출하고 추출된 객체를 이용하여 클래스를 추출한다[6,7]. 두 번째 단계는 추출된 클래스를 토대로 각 레벨별로 테스팅을 수행하는 것이다.

3.1 재공학 단계

이 단계에서는 절차중심으로 짜여진 프로그램으로부터 클래스를 추출하기 위한 단계이다.

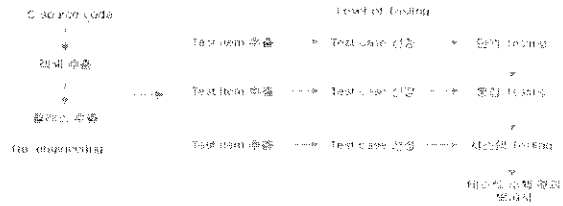

그림2. 테스팅 방안

먼저 소스코드로부터 객체를 추출하기 위한 알고리즘은 아래와 같다.

1. H/W diagram으로부터 H/W 관련 객체 H 추출
2. S/W diagram으로부터 S/W 관련 객체 S, S_i, S_d추출
   2-1. 객체 H와 직접적으로 관계 있는 객체 S_d 추출
   2-2. 객체 H와 간접적으로 관계 있는 객체 S_i 추출
   2-3. S_i, S_d 제외한 순수 S/W 객체 S 추출
3. 객체 H와 객체 S_d, S_i 각각을 합성하여 객체 HS_d , HS_i 추출
4. 객체 HS_d, HS_i를 단계1의 객체 H와의 relation으로 구축
5. 단계4와 객체 S를 단계2의 객체 S_d,S_i,S들과의 relation으로 구축

아래의 그림 3은 DPPS 안에 있는 내장형 소프트웨어의 흐름(S/W diagram)과 하드웨어의 흐름을(H/W diagram)을 나타낸 것이다.

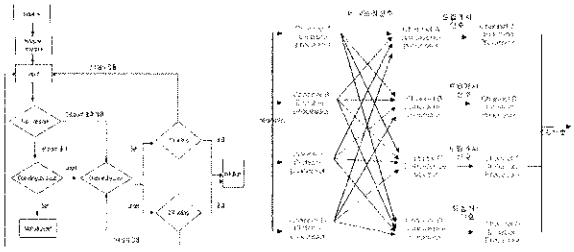

그림 3. S/W diagram 과 H/W diagram

위의 알고리즘을 적용하여 단계1과 단계2에서 객체를 추출하는 과정은 아래의 그림4와 같다.

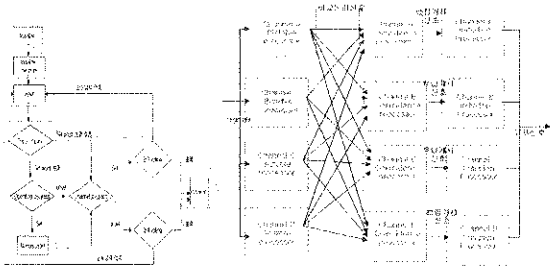

그림 4. S/W다이어그램으로부터 S_i, S_d 추출과 H/W다이어그램으로부터 H 추출

단계 1에서는 실제 하드웨어 별로 존재하는 각 모듈이 하나의 객체가 되며, 단계 2에서는 소프트웨어 다이어그램으로부터 실제 하드웨어의 모듈의 기능을 고려하여 관련 있는 함수들끼리 묶음으로서 S_i, S_d 를 추출한다. 이렇게 하여 얻은 각 객체들은 단계 3에 의해 객체 HS_d , HS_i 를 추출하며, 추출된 객체들을 정리하면 아래의 표1과 같다.

H (H/W 관련 객체)	S_d (H/W와 직접적인 기능이 있는 S/W 부분)	HS_d	chA_bis_obj chB_bis_obj chC_bis_obj chD_bis_obj chA_coin_obj chB_coin_obj chC_coin_obj chD_coin_obj chA_Ini_obj chB_Ini_obj chC_Ini_obj chD_Ini_obj
	S_i (H/W와 간접적인 기능이 있는 S/W 부분)	HS_i	Initialize Initialize_measure Input
S (S/W 관련 객체)	이 경우는 해당 사항이 없음		

표 1. DPPS로부터 추출된 객체

위의 알고리즘을 적용하여 얻은 각 S/W 객체와 H/W객체를 함께 표현한 다이어그램은 아래의 그림 5와 같다. DPPS에서는 S에 관한 객체가 존재하지 않기 때문에 위의 알고리즘에서 4단계까지만 적용해서 나타난 다이어그램이다. 이때 다이어그램으로 표시하는 규칙은 다음과 같다.

- 객체 표현 규칙
  - ⬚ : HS_d 를 표현
  - ⬚ : HS_i 를 표현
  - ⬚ : S를 표현
- 관계 표현 규칙
  - 실선 : 객체 내부의 논리적인 관계 표현
  - 점선 : 각 객체들간의 관계 표현
  - ◎ : 객체 HS_d 간의 입력이나 출력 값

그림 5에서는 실제 존재하는 4개의 채널별로 논리적으로 흐르는 S/W를 객체로 표현한 그림이다. 즉 실제 존재하는 각 하드웨어와 그것의 각 기능을 함께 표현한 객체들간의 관계를 나타낸 그림이다.

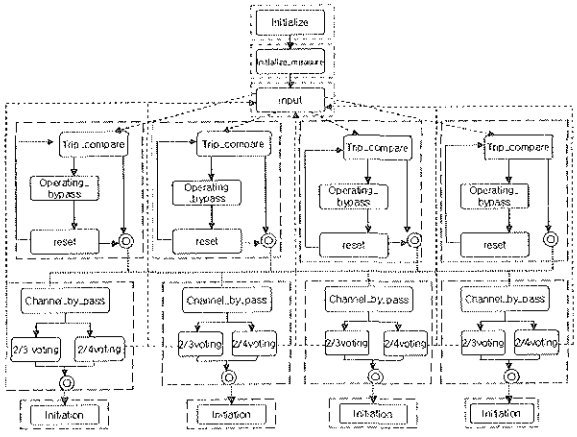

그림 5. 각 객체간의 관계를 표현한 다이어그램

3.2 레벨별 테스팅 단계

레벨별 테스팅 단계에서는 단위 테스팅, 통합 테스팅, 시스템 테스팅을 수행한다.

단위테스팅 단계에서는 각각의 모듈에 대한 테스팅을 수행하는 단계이다. 제공학 단계에서 추출한 각 클래스들에 대해 테스팅을 수행한다.

통합테스팅 단계에서는 단위 테스팅을 거친 각 클래스들의 결합한 조합에 대해서 테스팅을 수행한다. 이때 모든 가능한 모듈의 조합에 대해 다 테스팅을 수행할 필요는 없다. 그림 5에 나타난 각 객체들간의 관계를 토대로 클래스들을 테스팅 한다. 특히 각 객체를 수행한 후 얻어지는 그림 5의 토큰 값과 객체간의 점선을 중심으로 테스트를 수행한다.

시스템 테스팅 단계에서는 통합 테스팅 단계를 거친 후, 사용자 입력을 기준으로 전체 시스템이 목적에 맞게 제대로 작동하는지에 대한 테스팅을 수행한다.

4. 결론 및 향후 연구 과제

본 연구에서는 안전에 대한 치명도가 Safety 1E class로 가장 높은 RPS를 디지털화 하려는데 있어, 이 시스템에서 사용되는 내장형 소프트웨어의 안전성과 신뢰도 보장을 위한 테스팅 방법론을 제안했다.

이 방법에서 제안하는 방법은 크게 두 가지로 나누어 진다. 먼저 제공학 단계로, 절차중심의 프로그램으로부터 객체를 추출하기 위하여 대상 시스템의 하드웨어와 소프트웨어를 표현한 다이어그램으로부터 소프트웨어에 관련한 부분과 하드웨어에 관련한 부분으로 나누어, 그 관련성 여부에 따라 객체를 추출하고, 이렇게 추출된 객체 그룹을 바탕으로 클래스를 추출한다. 다음 단계는 레벨별 테스팅 단계로, 이 단계에서는 이미 추출한 클래스들을 이용하여 레벨별로 테스트 아이템을 추출하고, 이것을 토대로 레벨별 테스트 케이스를 선정한 후, 레벨별로 테스팅을 수행 한다.

현재 내장형 소프트웨어의 테스팅에 관한 연구는 초기 단계이며, 본 논문에서 제안한 알고리즘을 적용하여 실질적으로 내장형 소프트웨어를 테스팅 하기 위해서는 내장형 시스템의 특성 중에서도 특히 실시간적인 요소가 추가되어야 하며, 이를 테스팅 하기 위한 효과적인 방법론을 모색중이다.

5. 참고 문헌

[1]Patrick R.H. Place, Kyo C.Kang , "Safety-Critical Software Status Report and Annotated Bibliography", CMU/SEI-92-TR-5, ESC-TR-93-182

[2] EPRI TR-103331-V1 Research project 3093-01, "Guidelines for the verification and validation of expert systems *software and conventional software*", Vol. 1, 1995

[3] E. A. Lee, "What's Ahead for Embedded Software?", Computer, September, 2000

[4] Maier, T., "FMEA and FTA To Support Safe Design of Embedded Software in Safety-Critical Systems, " In CSR 12th Annual Workshop on Safety and Reliability of Software Based Systems, Bruges, Belgium, 1995.

[5]Mattias O'Nils and Axel Jantsch, "Communication in Hardware/Software Embedded Systems - A Taxonomy and Problem Formulation", Proceedings of the 15th NORCHIP Conference, November, 1997

[6] Robert S. Arnold, "Software Reengineering", IEEE Computer Society Press, 1994

[7] J.A Zimmer "Restructuring for style", Software Practice and Experience vol20(4), April, 1990