정형기법을 이용한 Safety-Critical System 개발방법론 적용사례+

ᵁ성창훈* 이주용* 이나영** 최진영*

고려대학교 컴퓨터학과*

서울대학교 원자력공학과**

Case Study on Development Methodology of Safety-Critical System Using Formal Method

ᵁSung, ChangHun* Lee, JooYong* Lee, Na-Young** Choi, Jin-Young*

Dept. of Computer Science and Engineering, Korea University*

Dept. of Nuclear Engineering, Seoul National University**

오늘날 우리가 사용하고 있는 시스템은 날이 갈수록 규모 면에서 대형화되고, 기능 면에서 복잡해지고 있다. 이런 복잡성의 증가로 시스템 에러 발생가능성은 더욱 높아졌다. 특히 safety-critical 시스템의 경우 에러가 발생했을 때 인간과 생태계에 엄청난 영향을 미치기 때문에 더욱 신중한 개발 과정이 필요하다. 따라서 정형기법을 이용한 safety-critical 시스템의 개발방법론이 나오게 되었다. 본 논문에서는 대표적인 safety-critical 시스템인 원자력 발전소 시스템에서 Digital Plant Protection System(DPPS)를 가지고 어떻게 방법론이 적용되는지를 보여준다. Software Cost Reduction(SCR)이라는 정형 명세 도구로 명세를 하였고, SPIN이라는 정형 검증 도구로 그 특성(property)을 검증하였다.

1. 서론

오늘날 우리가 사용하고 있는 시스템은 날이 갈수록 규모 면에서 대형화되고, 기능 면에서 복잡해지고 있다. 그러나 이러한 복잡성의 증가로 인하여 시스템에서의 에러 발생가능성은 더욱 높아졌다. 실수 변환 에러로 인한 Arian 5 rocket의 폭발, 덴버 공항의 지하 화물 처리 시스템의 실패 등은 발생하는 에러들이 인간의 생명과 재산에 얼마나 치명적인 결과를 가져오는지 알 수 있다. 이에 따라 safety-critical 시스템의 개발에 있어서 정형기법(Formal Method)이 강조되었고, 정형기법을 이용하여 safety-critical 시스템의 개발방법론까지 나오게 되었다.[1,2]

특히 원자력 발전소 시스템의 경우 기계의 낙후로 인하여 아날로그 시스템의 부품보급이 원활하지 않게 되자 디지털 시스템으로의 변환을 시도하고 있고, 이에 따라 에러발생을 없앨 수 있는 시스템 개발 방법론이 개발되었다.[3][4]

본 논문은 safety-critical 시스템인 원자력 발전소의 Digital Plant Protection System(DPPS)를 가지고 safety-critical 시스템 개발 방법론을 적용시켜본다. 2장에서는 DPPS와 정형기법도구인 SCR과 SPIN에 대해 알아보고, 3장에서는 SCR과 SPIN으로 DPPS를 어떻게 명세, 검증하였는지를 알아본다. 5장에서는 결론 및 향후 과제에 대해 제시한다.

2. DPPS 및 정형기법도구(SCR, SPIN)

2.1 Digital Plant Protection System

현재 운전 중인 발전소 중 Pressurized Water Reactor(PWR) 형태의 대부분의 원자력 발전소는 relay circuit으로 구성된 Reactor Protection System(RPS)을 가지고 있다. 기기 낙후와 교체 부품의 단종 등에 의해 RPS를 디지털화하려는 시도가 있으며 여기에서는 이 중 CE type의 Digital Plant Protection System에 대해 연구하였다[5].

Digital Plant Protection System(DPPS)은 다중성을 고려하여 A, B, C, D 4개의 독립적인 채널로 구분된 장치 구조로 구성되어 있다. 각 장치구조는 BISTABLE, COINCIDENCE, INTERFACE &TEST 프로세서로 구성되어 있으며, 연구에 적용하기 위하여 간략화 한 구조는 그림1과 같다.

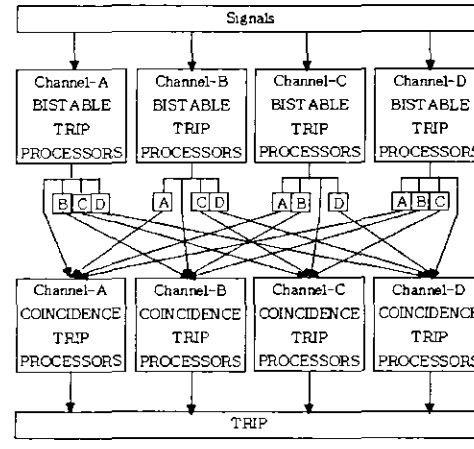

그림 1 Digital Plant Protection System

각각의 네 개의 채널에서 BISTABLE 프로세서는 입력 신호들을 모니터를 하고 있는 프로세스로부터 값들을 전달받는다. 입력 값은 10개의 아날로그 신호와 2개의 디지털 신호로 구성된다. 원전의 가동 시작 단계나 가동 중단 등의 상태에서는 이러한 한계 값을 바로 적용할 수 없기 때문에 수동 조작이 가능하게 하며, 몇 가지 복잡한 단계를 거치게 되지만, 여기에서는 이에 대한 내용은 생략하여 단순화하였다.

원전의 상태를 모니터링하는 센서들로부터 10개의 아날로그 입력을 받게 되며, 몇 가지 입력의 조합으로 계산한 2개의 디지털 입력을 받게 된다. 이는, 각 채널의 BISTABLE 프로세서로 전달된다. BISTABLE 프로세서는 입력 신호들의 값들이 기존에 결정되어 있는 한계 값들과 비교하여 trip상태를 결정한다. 만약 BISTABLE 프로세스 단계에서 입력 값들 중에 하나의 값이라도 한계 값을 넘어서거나 낮아지게 되면 BISTABLE 프로세서는 각 채널을 거쳐 COINCIDENCE 프로세서에게 trip신호를

+본 논문은 한국 전력 공사의 지원에 의하여 기초전력공학공동연구소의 주관으로 수행되었음.

날리게 된다.

BISTABLE 프로세서가 입력 신호를 받을 때 operating bypass 신호를 확인하여 bypass신호가 세팅되어 있는 경우에는 입력된 센서의 신호를 무시하고 계속 입력 신호를 받아들인다. bypass 신호는 원자로 제어에 관련하여 관리자가 인위적으로 값들을 조작하는 경우나, 시스템을 처음 켰을 경우에 세팅된다.

COINCIDENCE 프로세서는 BISTABLE의 output을 조사한다. BISTABLE 프로세서는 입력 값이 trip condition에 적용되는 경우에 신호를 날려주게 되며 COINCIDENCE 프로세서는 독립적인 네 개 채널의 BISTABLE 프로세서 출력을 모두 확인하여 2개 이상이 trip신호를 가질 때 trip initiation 프로세서 쪽으로 출력을 내게 된다. 즉, 각 BISTABLE 프로세서 채널에서 입력 값이 한계 값을 넘어설 경우 각 프로세서는 trip신호를 각각의 COINCIDENCE 프로세서에게 전달을 하고, 각 COINCIDENCE 프로세서는 전달되는 값들을 모두 살펴 2개 이상이 trip일 때 trip신호를 날리게 된다. 그럼 DPPS는 결국 trip 신호를 trip initiation 프로세서로 전달하게 되고, 제어봉을 지지하고 있는 회로의 입력을 끊어 중력에 의한 제어봉을 낙하하여 원자로가 정지되게 한다.

2.2 Software Cost Reduction

SCR[6]은 NRL에서 A-7항공기 비행 프로그램의 요구명세를 위해 개발되었다. 즉, SCR은 시스템이나 소프트웨어의 요구명세를 도와주기 위해 만들어진 정형검증도구이다. 이 NRL의 연구는 실제적인 정형명세에 초점이 맞추어져 있다. SCR은 embedded , real-time 소프트웨어와 시스템을 위한 software 요구를 명세하고, 그러한 명세를 위한 tool 과 method를 평가한다.

SCR의 하드웨어 인터페이스는 입력과 출력데이터 아이템과 소프트웨어 함수로 기술된다. 이러한 소프트웨어 함수는 mode transition tables, event tables, condition table과 같은 표의 집합으로 이루어져 있다. 또한 SCR은 condition, events, modes, mode classes와 같은 높은 단계의 구조를 지원한기 때문에 더욱 명세를 이해하기 쉽다. 또한 SCR을 이용한 정형명세 방법은 시스템이나 소프트웨어의 상태와 그에 관련된 변수들을 table로 표현하여 요구명세를 검증한다. SCR은 단순히 시스템이나 소프트웨어의 요구를 명세할 뿐만 아니라 명세한 요구의 일관성(consistency)과 완전성(completeness)을 체크한다. 또 SCR은 명세한 시스템이나 소프트웨어를 시뮬레이션 할 수도 있다.

2.3 SPIN

SPIN[7,8]은 AT&T Bell 연구소에서 1995년에 발표한 LTL(Linear temporal Logic) 모델체커이다. 기존의 모델체킹 방법을 개선하기 위해 On-the-Fly 방식을 사용하여 메모리의 효율적인 사용을 가능하도록 하였다. SPIN은 비동기 프로세스 시스템(asynchronous process system)의 설계와 검증을 지원하는 가장 일반적인 tool이다. SPIN은 또한 분산 소프트웨어(distributed software)와 통신 프로토콜(communication protocol) 검증에 아주 유용하게 사용되고 있다. SPIN 검증 모델은 프로세스 상호작용(Process interactions)의 정확성(correctness)에 초점을 둔다.

SPIN은 분산 시스템에서의 논리적 설계 오류를 찾는데 사용될 수 있다. 특히 운영체제, 데이터 통신 시스템, 교환 시스템(Switching Systems), 동시성 알고리즘(concurrent algorithms), 철도 신호 프로토콜(railway signaling protocols) 등 실제적인 시스템의 디자인에 매우 적합한 것으로 평가되고 있다. 이 도구를 이용해서 명세의 일관성을 검사할 수 있는데 데드락(Deadlock), 명세되지 않은 반응(Unspecified receptions), flags incompleteness, 경쟁 조건(race conditions)등을 잘 보여준다. 선형시제논리(LTL)이나, next-time free LTL, 또는 Buchi Automata를 이용해서 정확성(correctness properties)을 검증한다.

SPIN은 PROMELA(Process Meta Language)라는 검증언어를 사용하여 설계를 하고 PROMELA는 LTL의 구문으로 명세화 하여 정확성을 증명한다. PROMELA는 검증 모델 언어(Verification modeling language)이고, 분산 시스템이나 프로토콜의 추상모델(abstraction)을 제공하고 프로세스(processes), 메시지 채널(message channels), 그리고 변수들로 구성되어 있다.

3. 적용사례

3.1 DPPS의 state diagram

DPPS는 각 네 개의 채널에서 BISTABLE과 COINCIDENCE, TRIP프로세스로 구성되고, 각자의 일을 별개로 수행하게 된다. 간략화하면 그 기능은 10개의 analog신호와 2개의 digital신호를 입력받아 하나라도 정해진 범위를 벗어나면 신호를 보내고 두 채널이상에서 한계 값을 벗어난 신호가 발견되면 trip을 발생한다. 그 state diagram을 간략화하여 표현하면 그림 2와 같다.

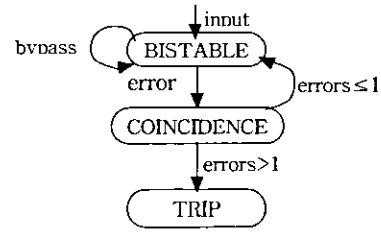

그림 2 DPPS를 간략화한 state diagram

3.2 SCR 명세

DPPS를 명세함에 있어서 DPPS의 모든 변수와 모든 채널을 모두 명세 하였을 경우 SPIN으로 변환하여 검증할 경우 상태 폭발(state explosion)이 발생하여 모든 변수를 모두 명세할 수 없음을 알았다. 그래서 입력 변수가 한계 값을 넘어서는지는 다른 모듈로 체크할 수 있다고 판단하고 입력 값을 각 채널 당 하나로 잡았다. 각 변수 table은 다음과 같다.

name	Initial	Class	Discription
input_a	False	Monitored	채널A input변수
input_b	False	Monitored	채널B input변수
input_c	False	Monitored	채널C input변수
input_d	False	Monitored	채널D input변수
bypass	False	Monitored	bypass signal
bis_out_a	False	Controlled	Bistable A output
bis_out_a	False	Controlled	Bistable B output
bis_out_a	False	Controlled	Bistable C output
bis_out_a	False	Controlled	Bistable D output
coin_out	False	Controlled	Coincidence output

표 1 Variable Dictionary

각 변수들에 대해 살펴보면 input_a, input_b, input_c, input_d는 각 채널로 들어가는 입력 값들을 묶어놓은 것이다. 하나의 값이라도 한계 값을 넘어서면 이 값은 true가 된다. bypass는 bypass 신호를 처리하기 위해 만든 것으로 이것이 true인 경우는 입력 값이 한계 값을 넘어

서더라도 BISTABLE 프로세서는 출력을 false로 내보내게 된다. bis_out_a, bis_out_b, bis_out_c, bis_out_d는 각 채널의 BISTABLE 프로세서의 출력 값으로 bypass 신호가 false일 때 입력 값이 한계 값을 넘어선 경우 true가 된다. coin_out은 COINCIDENCE 프로세서의 출력 값으로 *bis_out*의 값을 살펴보아 *2개 이상의 값이 true*가 되면 true로 된다. 각 controlled variable을 살펴보면 다음과 같다.

```
if (input_* and (not bypass))
  then bis_out_*=true
// bis_input_*는 bypass가 false일 때
// input_*가 참이면 true가 된다.
```

```
if (not (((bis_out_a=FALSE) and (bis_out_b=FALSE) and
          (bis_out_c=FALSE) and (bis_out_d=FALSE))
      or ((bis_out_a=TRUE) and (bis_out_b=FALSE) and
          (bis_out_c=FALSE) and (bis_out_d=FALSE))
      or ((bis_out_a=FALSE) and (bis_out_b=TRUE) and
          (bis_out_c=FALSE) and (bis_out_d=FALSE))
      or ((bis_out_a=FALSE) and (bis_out_b=FALSE) and
          (bis_out_c=TRUE) and (bis_out_d=FALSE))
      or ((bis_out_a=FALSE) and (bis_out_b=FALSE) and
          (bis_out_c=FALSE) and (bis_out_d=TRUE))))
  then coin_out=true
// coin_out은 bis_out_*의 값들이
// 2개 이상이 참이 되면 true가 된다.
```

각 state의 mode class table은 표2와 같이 표현된다.

Source	Event	Destination
bistable	@T((bis_out_a=FALSE) and (bis_out_b=FALSE) and (bis_out_c=FALSE) and (bis_out_d=FALSE))	bistable
bistable	@T(not ((bis_out_a=FALSE) and (bis_out_b=FALSE) and (bis_out_c=FALSE) and (bis_out_d=FALSE)))	coincidence
coincidence	@T(coin_out)	trip
coincidence	@T((bis_out_a=FALSE) and (bis_out_b=FALSE) and (bis_out_c=FALSE) and (bis_out_d=FALSE))	bistable

표 2 Mode Class Table

3.3 SPIN 검증

SPIN을 검증도구로 삼은 것은 시스템의 일관성과 완전성을 검증해 줄 수 있을 뿐만 아니라 SCR에서 SPIN language로 바로 변환이 이루어진다는 것이다. 사람의 손을 덜 거치는 것이 시스템에서 발생할 수 있는 에러가 더욱 줄어들기 때문에 될 수 있는 한 자동화방법을 사용하는 것이다. (SCR에서 SPIN으로 변화되는 것은 [9]을 참조바람.)

자동화된 SPIN 코드를 검증하기 위하여 DPPS의 가장 중요한 특성을 살펴보자면 다음과 같이 정리할 수 있다.

> • bypass가 false인 경우 두 채널 이상에서 입력 값이 한계 값을 넘어서면 trip이 발생하여야 한다.

이것을 SPIN에서 검증하기 위하여 LTL로 나타내면 다음과 같이 나타낼 수 있다.

```
#define p1 (input_a_NEW==FALSE && input_b_NEW==FALSE &&
            input_c_NEW==FALSE && input_d_NEW==FALSE)
#define p2 (input_a_NEW==TRUE && input_b_NEW==FALSE &&
            input_c_NEW==FALSE && input_d_NEW==FALSE)
#define p3 (input_a_NEW==FALSE && input_b_NEW==TRUE &&
            input_c_NEW==FALSE && input_d_NEW==FALSE)
#define p4 (input_a_NEW==FALSE && input_b_NEW==FALSE &&
            input_c_NEW==TRUE && input_d_NEW==FALSE)
#define p5 (input_a_NEW==FALSE && input_b_NEW==FALSE &&
            input_c_NEW==FALSE && input_d_NEW==TRUE)
#define dpps        (DPPS_NEW==trip)
#define bypass      (bypass_NEW==TRUE)

[] (!(p1 || p2 || p3 || p4 || p5) && !bypass) -> <> dpps)
```

LTL로 나타낸 이 특성은 SPIN에서 valid하다고 나왔고, DPPS의 명세는 올바르게 시스템의 요구를 만족한다고 볼 수 있다.

4. 결론

safety-critical 시스템은 에러가 발생하였을 때 인간 생명에 치명적인 영향을 미칠게 된다. 특히 DPPS의 경우 원자로의 상태를 입력 값으로 받아 처리하게 때문에 단 하나의 에러가 발전소 주변 생태계에 엄청난 영향을 미치게 된다. 따라서 개발 시작부터 끝까지 정형기법을 사용하여 단계별로 정형 명세하고 정형 검증하여야 한다. 즉 시스템을 정확하게 명세하고 그 특성(property)을 확인하는 V&V과정을 거쳐야 한다.

본 논문은 정형기법을 사용하여 safety-critical 시스템을 개발할 때 정형기법 도구를 어떻게 사용하는지에 대해 살펴보았다. SCR을 사용하여 시스템을 명세할 때 어떻게 상태 변수들과 제어변수를 나누는지 제어변수는 어떻게 적용되어 Mode Class Event에 적용되는지를 알아보았고, 자동화하여 SPIN code가 나왔을 때 어떻게 특성을 검증하는지도 알아보았다.

이는 앞으로 진행될 모든 safety-critical 시스템의 개발에 있어서 어떻게 시스템 개발이 이루어지는지 그 한 예를 든 것이다.

향후 과제로 검증된 명세로 코드를 만들어 내었을 때 어떻게 검증하고 테스트 할 것인가가 있다. 검증된 safety-critical 시스템을 코드로 만들어 냈을 때 이것이 정말 믿을 수 있는 코드인가에 대한 검증과 테스트하는 과정을 알아보는 것이다.

5. Acknowledgement

본 논문에서 작성된 SCR자료는 University of Pennsylvania의 Real-Time Group에서 작성된 것입니다.

6. 참고 문헌

[1] Dolores R. Wallace and Laura M. Ippolito, "A Framework for the Development and Assurance of High Integrity Software", NIST500-233, Dec. 1994

[2] *Edmund M. Clark and Jeannette M. Wing, "Formal* methods : State of the Art and Future Directions". ACM Computing surveys, Dec.1996

[3] 성창훈 외 3명, "Development Methodology of Safety-Critical System Using Formal Method", 한국정보과학회 추계학술대회 학술지, 2000

[4] Na Young Lee et al. "Methodological Approach to Software V&V for the Reactor Protection System", 2000 춘계 원자력 학술대회

[5] 한국전력공사 "UCN 3&4 Final Safety Analysis Report" Vol. 11

[6] James Kirby, "SCR* Toolset : The User Guide", Jan.1997

[7] Gerald J. Holzmann, "The model Checker SPIN", IEEE Transactions on Software Engineering, Vol. 23, pp279~295, May. 1997

[8] Gerard J. Holzmann, "Design and Validation Computer Protocols", Prentice Hall, 1991

[9] R. Bharadwaj and C. Heitmeyer, "Model Checking Complete Requirements Specifications Using Abstraction", ASE1999, Vol. 6, pp 37~68, 1999