2001 춘계학술발표회 논문집
한국 원자력 학회

원자력 발전소 내장형 Protection 시스템의 설계에 대한 정형기법의 적용+

Applying Formal Method to Design of Nuclear Power Plant Embedded Protection System

[0]김진현* 김일곤* 성창훈* 이나영** 최진영*

고려대학교 컴퓨터학과*
서울시 성북구 안암동 5가 1번지
서울대학교 원자핵공학과**
서울시 관악구 신림동 산 56-1

요 약

원자력 발전 내장형 Protection 시스템은 원자력 발전의 과부하를 검출하여 원자로의 가동을 중단시키는 대표적인 Safety-critical 시스템이다. 이러한 시스템은 그 위험성으로 인해 안전성과 신뢰성이 절대적으로 요구되는 시스템이다. 따라서 원자력 발전 내장형 Protection 시스템은 그 위험성을 가만하여 설계 단계부터 V&V가 철저하게 이루어져야 한다. 이러한 내장형 시스템을 설계하기 위해 다양한 V&V 방법이 제시되고 있으며 특히 선진국에서는 정형기법을 이용한 설계가 연구되고 있다. 본 연구에서는 원자력 발전소 소프트웨어의 가이드라인에 맞추어 다양한 측면에서의 정형기법을 이용한 원자력 발전 내장형 Protection 시스템의 설계에 대한 기법을 소개한다.

Abstract

Nuclear power embedded protection systems is a typical safety-critical system, which detects its failure and shutdowns its operation of nuclear reactor. These systems are very dangerous so that it absolutely requires safety and reliability. Therefore nuclear power embedded protection system should fulfill verification and validation completely from the design stage. To develop embedded system, various V&V method have been provided and especially its design using Formal Method is studied in other advanced country. In this paper, we introduce design method of nuclear power embedded protection systems using various Formal-Method in various respect following nuclear power plant software development guideline.

1. 서론

오늘날 대부분 선진국에서는 원자력 발전소에서 전기의 30퍼센트나 그 이상을 공급하고 있다. 핵

+ 본 연구는 한국전력공사의 지원에 의하여 기초전력공학공동연구소의 지원으로 수행되었음

에너지는 더욱더 환경 친화적인 에너지 원천 중 하나이다. 그러나, 1979년 Three Mile 사고와 1986년 체르노빌 1 사고는 오랜 기간 영향을 미치는 위험이 잠재적으로 존재함을 보여주는 사례라고 볼 수 있다. 그 결과 원자력 발전소 산업은 대중의 안전을 보호하기 위해 국가 정보 차원에서 통제되고 있다. 원자력 발전소 산업은 안전성과 신뢰성 공학의 영역에서 오랜 역사를 가지고 있다. 최근에는 기계와 통제(I&C)시스템에 대한 기술이 소프트웨어 통제로 바뀌고 있다. 이러한 타겟 머신에 들어가는 소프트웨어 역시 일반적인 원자력 발전 시스템과 같이 높은 수준의 안정성과 신뢰성을 요구하고 있다. 특히 Plant Protection 시스템과 같이 사고를 대비하는 타겟 머신에 들어가는 소프트웨어라면 그 안정성이나 신뢰성은 더욱 강조될 수 밖에 없다. 왜냐하면 이러한 시스템의 사고는 대형 인명 및 재산 피해를 가져올 수 있기 때문이다. 따라서 원자력 발전 시스템의 내장형 소프트웨어의 설계는 그 설계 초기부터 다양한 방법으로 Validation&Verification(이하 V&V) 기법으로 검사되고 검증되어야 한다. 본 논문은 기존의 원자력 발전 소프트웨어 시스템에 대한 설계 가이드라인에 근거한 정형기법을 원자력 발전 내장형 시스템에 적용시키는 기법에 대한 연구이다. 2장에서는 원자력 발전 시스템의 소프트웨어에 대한 가이드라인을 기술하고 3장에서는 이를 충족시킬 수 있는 정형기법의 적용에 대해 논한다. 4장부터 7장은 2장의 가이드라인을 중심으로 3장에서 제안한 정형기법을 토대로 Digital Plant Protection 시스템에 적용한 예를 보일 것이며 8장에서는 결론과 향후 연구 방향을 논한다.

2. 원자력 발전 내장형 시스템의 가이드라인

원자력 장비에 대한 책임을 관장하고 있는 국제 전기 위원회(IEC) 부속 위원회(SC) 45A는 IEC 60880을 제정하였다.[1] IEC SC 45A는 1977년 처음 표준인 IEC568을 작성하였으며 오늘날 까지도 이용되고 있다. IEC회원국들은 일반적 비평이나 투표 과정에 참여하게 된다. 원자력 발전소 산업의 소프트웨어 안전성과 신뢰성을 다룬 최초 표준들 중 하나인 IEC 60880은 1986년 승인되고 발행되어졌다. 그 이후 몇 번의 재검사 과정을 거쳤으며 오늘날 그 효력을 발생하고 있다. IEC60880은 원자력 발전소에 사용되는 소프트웨어에 대한 하드웨어 안전성 디자인 원리들을 해석하고 확장하는 목적을 표현하기 위해 작성되어졌다. 예를 들면, 표준은 다음과 같이 신뢰성 공학으로 확장되어 가고 있다. IEC 60880의 목적은 생산, 운영, 유지에 대한 지침서를 제공하는 것이다. IEC 60880은 요구사항 분석 과정과 정확하고 완벽한 요구사항에 대한 설명에 대해 크게 강조하고 있다. N. Thuy와 F. Ficheau-Vapne에 따르면 정확한 요구사항에 대해 강조하고 있는 한 이유는 원자력 발전소에 있는 소프트웨어의 생명 기간이 대체적으로 길다는 것이다. 컴퓨터 안전 시스템에 할당된 기능을 위한 소프트웨어 요구 사항들은 시스템 요구 명세로부터 도출된다. 그러한 표준이 기술하는 것은 요구 사항들은 이해할 수 있어야 하고, 간결하고, 장황하지 않아야 하며, 통용성을 가져야 하며, 모호성을 갖지 않아야 하며, 테스트할 수 있고, 검증 수 있으며, 실행할 수 있어야 한다. 명세의 명료성과 완전성을 높이기 위해 정형 명세 언어의 사용이 권장되어진다. 요구사항은 “프로젝트가 아닌, 제품에 대해 기술”해야 하고 “구현에 대한 상세성이 없어야 한다” 즉, 무엇을 만드는지를 설명해야지, 어떻게 만드는지를 설명해서는 안 된다. 요구 사항들은 이용할 수 있는 자원의 측면에서 구현의 우선순위를 정하는데 도움을 주기 위해 의무적으로나 선택적으로 레이블화 되어야만 한다. 그러한 표준은 요구사항이 소프트웨어가 라이센스 문제를 언급하지 말아야 할 것을 포함하고 있다. 완전성

의 목적을 충족시키기 위해, 표준의 요구 사항들을 다음과 같은 9가지 분류로 그룹화 하였다.

- *기능 요구사항*
- *시스템 수행 요구사항*
- *신뢰성 요구사항*
- *에러 처리 요구사항*
- *지속적인 모니터링 요구사항*
- *인간 컴퓨터 상호작용(HCI) 요구사항*
- *시스템 인터페이스 요구사항*
- *운영 환경 제한사항과*
- *데이터 요구사항*

기능 요구사항은 소프트웨어에 부여된 시스템 안전성 기능에 대해 기술하고 있다. 각각의 기능은 그 목적상, 요구되는 입력, 요구되는 출력, 다른 기능과의 상호작용 그리고 시스템 신뢰성에 대한 관계에 의해 기술되어져야 한다. 기능 요구사항은 동작 조건, 태스크 시퀀싱, 기능 검증 그리고 에러 탐지를 기술하는 그래픽 표현 , 수학적 표현, 로직 다이어그램과 진리표에 의해 설명되어져야만 한다. 입력 영역, 입력 정확성과 노이즈 한계성, 내부 표현성, 출력 정확성 그리고 요구되어지는 입력과 출력의 출력 범위는 상세히 정의 되어져야만 한다.

시스템 수행 요구사항은 어떻게 시스템이 낮은, 정상적인, 높은 범위 아래에서 그리고 높은 조건 위의 상황에서 수행되어야만 하는지를 설명해야만 한다. 의무적인 용량(mandatory volume), 출력, 응답시간, 통신 동기화 그리고 정확성 요구사항이 각각의 이러한 상황들에 대해 정의되어져야만 한다.

소프트웨어 신뢰성 요구사항은 시스템 신뢰성 요구 사항으로부터 도출되어야 하거나 특히 시스템 요구사항 명세의 소프트웨어에 적용되어야 한다. 이것은 프로젝트마다 다양하다. 소프트웨어 신뢰성 요구사항은 운영 환경과 올바른 동작 프로파일과 관련되어져야 한다. IEC 60880은 양적인 소프트웨어 신뢰성 목표에 그 초점을 맞추고 있다. 표준이 본래 발행되었을 때, 질적인 소프트웨어 신뢰성 측정의 개념이 상대적으로 알려지지 않았기 때문에 이것은 이해할 수 있어야 한다. 그러나, Tate와 B. Jennings는 양적인 소프트웨어 신뢰성 측정 증명의 어려움을 지적하였다.

에러 처리 요구사항은 어떻게 소프트웨어가 사고를 방지하거나 포함하는 목적을 갖는 에러 상황을 탐지하고, 복구하고 그리고 반응해야만 하는지를 명세하기 때문에, 직접적으로 신뢰성 요구사항과 관련 있다. 따라서, 초기 상황들은 다음에 대해 명세 되어져야만 한다.

- 사고 상황시 방사능 노출을 방지하기 위한 기능
- 비상 차단 상황
- 발전소 피해를 막기위한 기능

본 연구에서는 위의 요구 사항 가운데 기능적 요구사항, 시스템 수행의 요구사항, 소프트웨어 신뢰성의 요구사항 및 에러처리 요구사항을 충족시키는 설계 방법론에 대해 논하고 있다. 특히 2장에서 논한 가이드라인에 맞는 다음과 같은 정형기법[2,3]을 적용하고자 한다.

다음 장에서는 이러한 정형기법을 지향하는 개발 방법론을 소개하고 계속해서 이러한 정형적 개발 방법론을 기반으로 실제 어떻게 적용 할 것인지 논한다.

3. 원자력 발전 내장형 시스템의 정형적 설계

<그림 1>은 미국의 National Institute of Standards and Technology(NIST)에서 제안한 High Integrity Software개발 방법론이다.[4] 이 개발 방법론은 높은 신뢰성과 안정성을 요구하는 소프트웨어에 대한 방법론을 지향하고 있다. 따라서 본 논문의 2장에서 언급한 가이드라인이 요구하는 신뢰성이나 안정성을 측면의 설계 및 개발에도 적합할 것이라 생각하고 원자력 발전 내장형 시스템 설계 과정을 이에 맞추어 보고자 한다. 이 방법론은 정형기법을 기본으로 V&V를 수행하여 safety-critical 내장형 시스템을 개발 제시한다. 즉 시작 단계부터 끝까지 정형기법을 사용하여 단계 별로 정형 명세하고 정형 검증을 하여, 시스템의 특성(property)을 확인하고 검증(verification)과 확인(validation) 과정을 거치는 것이다. 설계 및 개발의 전체 과정은 <그림 1>과 같은 방법으로 진행된다. <그림 1>에서 SRS는 System Requirement Specification이고, SDD는 System Design Description을 가리킨다. V&V는 Verification & Validation과정을 이야기하며, R V&V는 Requirement V&V와 SD V&V는 System Design의 V&V를 가리킨다. 전체 개발과정은 일반 시스템의 개발 방법론처럼 Requirement에서 specification, design, code, integration, operation & maintenance과정으로 개발이 되지만 각 단계마다 V&V과정이 추가되어야 한다. 그리고, 필요에 따라 바로 이전 단계로 돌아가 그 단계의 작업을 다시 수행하기도 한다. 또한 전체 개발과정에서 management 과정과 quality assurance과정이 필요하다. SRS 과정은 시스템의 요구사항을 특성별로 구분해야 한다. 이때 시스템의 요구사항으로 완전(completeness)하고, 일관(consistency)되고, 옳고(correctness), 테스트 가능(testable)하고, 이해할 수 있는(understandable) 정보를 제공해야 한다. 그러기 위해서는 사람의 손을 최소한 적게 거치도록 개발된 정형명세 도구를 사용하는 것이 바람직하다.

SDD 과정은 명세 된 요구사항에 기반을 두고, 규명할 수 있는 시스템 설계를 개발하는 것이다. 또 시스템 코드를 생성하기 위해 완전하고, 일관되고, 옳고, 테스트 가능하며, 이해할 수 있는 정보를 제공하는 것이다. 코드(code)생성과정의 중요한 목적은 SRS와 SDD에 기반을 두고, 규명할 수 있는 소스코드를 개발하는 것이다. 또한 통합(integration) 과정은 실행 가능한 코드를 생산하고, 다른 소프트웨어나 시스템 컴포넌트에서 실행 가능한 코드를 통합한다. 이 과정은 시스템을 설치하기 위해 준비하는 단계에서 다른 소프트웨어 컴포넌트와 하드웨어를 가지고 시스템 컴포넌트를 통합하기 위해 소스코드를 사용하게 된다. 수행과 유지보수(operation and maintenance)과정의 목적은 시스템이 그

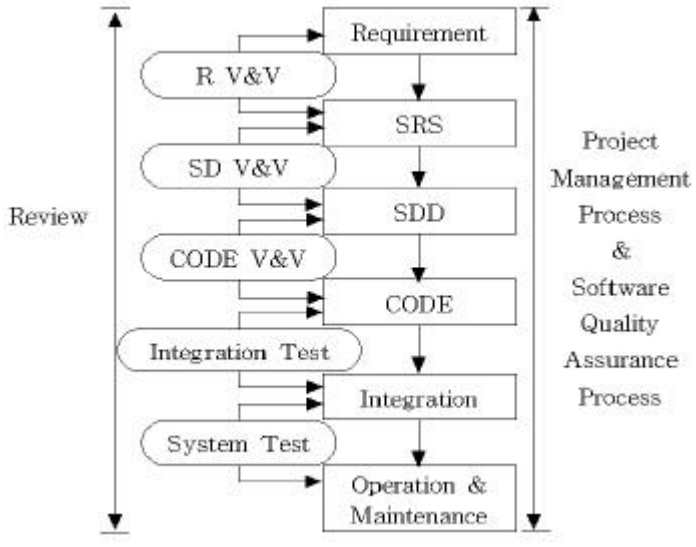

그림 1 NIST에서 제안한 High-Integrity System의 개발 과정

수행동안, 또 그것이 수정되었을 경우 시스템의 요구사항을 잘 만족하는지 확실히 하는 것이다. 이 과정은 시스템이 설치되어 있는 모든 곳을 필요(에러, 수행 환경의 변화나 강화)에 따라 시스템을 수정한다. 본질적으로, 이 과정은 선행하는 모든 과정들을 반복한다. 관리(management)와 품질보증(quality assurance)과정의 주목적은 시스템 개발 과정이 시스템보증과정의 계획과 규정에 만족하는가 확인하고 개발 과정의 개선점을 제시하는 것이다. 즉, 시스템의 개발이 계획대로 진행되고 있고, 기준에 맞게 개발되고 있는지를 확인한다. 또 개발 과정에 문제점이 있다면 그것을 보안하고 개선할 수 있는 방안을 제시하는 과정이다.

본 논문에서는 특히 설계 과정에서의 V&V에 초점을 맞추고자 한다. 좋은 설계도에서 좋은 구현이 나온다는 것은 거듭 강조해도 지나치지 않다. 따라서 앞서 언급한 가이드라인에 부합되는 기준에 따라 NIST에서 제안 방법을 구현하기 위해 본 연구에서는 정형기법이 적합하다고 제안한다.
먼저 STATECHART를 통한 도식화된 표현과 SCR을 통한 입력 영역 및 입력 정확성, 노이즈의 한계성들을 표현하고, ESTEREL을 통해서 행위에 대한 전반적인 명세를 할 수 있다. 또한 각각 SMV, SPIN, XEVE를 통한 모델 체킹 기법을 이용하여 시스템의 기능에 대한 완벽한 V&V를 할 수 있다. 이러한 정형적 설계와 검증 및 검사(V&V)를 통해서 기능에 대한 요구 사항에 대한 분석, 시스템의 행위에 대한 요구사항 분석, 신뢰성의 요구 사항에 대한 분석 및 에러 탐지에 대한 요구 사항을 설계도 측면에서 보다 정확하게 분석할 수 있다. 따라서 본 논문에서는 정형기법을 통한 설계의 각종 예를 보임으로 어떻게 정형기법을 적용함으로써 원자력 발전 시스템 개발의 가이드라인에 맞는 시스템을 설계할 수 있는지를 보일 것이다. 다음 장에서는 우선 정형기법이 무엇인지 그 특징을 논한다.

3.1. 정형기법

정형기법[2,3]이란 소프트웨어 공학의 일종으로 오류가 없는 시스템을 설계하여 시스템의 신뢰성을 높이려는데 목적이 있다. 정형기법에서는 시스템의 구성 요소를 수학적 객체 모들로 취급하며, 이러한 객체의 성질과 동작을 묘사하고 예측하기 위해 수학적 모델을 제공한다. 이러한 수학적 기호를 사용함으로 시스템의 명세를 작성하는데 있어서 자연어가 일으킬 수 있는 애매모호함이나 불확실성을 최소한으로 줄일 수 있고, 설계된 사용자의 요구조건과 동일한지 수학적인 성질을 이용하여 증명할 수 있다. 정형기법은 크게 정형명세(Formal Specification)와 정형검증(Formal Verification)으로 나눌 수 있는데 정형명세는 시스템이 달성해야 하는 요구 사항과 그러한 요구 사항을 만족시키는 설계를 묘사하는데 그 목적이 있고, 정형 검증은 그 설계가 요구사항을 만족하는지를 검사하는 일련의 과정을 의미한다. 따라서 정형기법은 정형명세를 통하여 2장에서 언급한 설계 및 요구 사항에 대한 명확한 이해를 하게 할 뿐 아니라 정형검증을 통하여 설계도 자체를 명확하기 분석할 수 있는 도구를 제시한다. 정형검증 가운데에는 논리적 증명을 이용한 Theorem Proving기법과, 시스템이 취하는 모든 상태를 분석하는 모델 체킹(Model-Checking)기법[3], 그리고 Process Algebra 의 Bi-simulation의 기법이 있다. 본 논문에서는 정형기법을 2장과 3장에서 언급한 방법대로 적용하기 위해 다음 세가지 정형명세 및 검증 기법을 사용하기로 한다. 먼저 용이한 설계와 이해를 위한 도식적 명세를 위한 STATECHART를 이용한 설계와 이에 대한 검증을 위해 모델 체킹 기반의 SMV를 이용할 것이며, 테이블 기반의 SCR을 통한 설계와 Validation 그리고 그 설계에 대한 행위적 분석을 위한 SPIN으로의 검증을 시행할 것이

다. 또한 함수형 언어를 통해 마치 프로그램을 짜듯이 Reactive 시스템을 설계하게 하는 설계 시스템인 ESTEREL 설계 시스템을 이용하여 설계 및 V&V를 시행할 것이다.

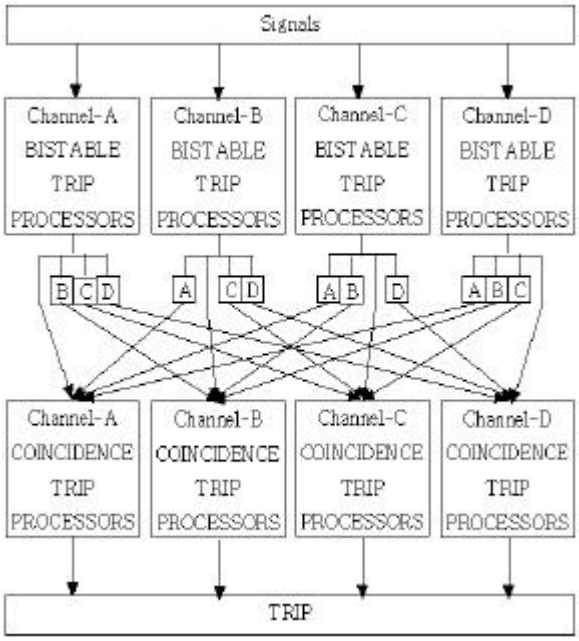

그림 2 DPPS 시스템 구조

4. 원자력 발전 내장형 Digital Plant Protection System의 명세

4.1. DPPS 시스템 및 요구사항

본 논문에서 설계하고자 하는 시스템은 <그림 2>와 같다.

DPPS 시스템은 크게 A,B,C,D 4 개의 고립된 각각의 Equipment room 들로 구성되어 있다. 각각의 Equipment room 에는 PPS(Plant Protection System)이 있으며 이것은 Core Protection Calculator 와 Plant Protection Calculator(PPS) 가 있다. 이 PPS 는 Bistable 과 Coincidence 와 Interface & Test Processor 가 있다. 각각의 PPS 채널은 모니터링되는 별개의 측정값을 받게된다. 다음은 PPS 내의 각 부분의 기능을 명세한 것이다.

1) *Bistable Trip Processor :* 입력 신호가 정해진 Set-point 를 넘는지를 검사한다. 즉 각각의 입력 변수에 대해 그 변수가 Set-point 를 넘는지를 확인하여 만약 Set-point 를 넘는다면 Error 신호를 다음 단계로 넘겨준다.

2) *Coincidence Processor* : Bistable Trip Processor 로부터 넘어온 Error 가 발생한 신호에 대해 다른 채널의 동일한 신호를 기다려 그 신호 역시 잘못 되었는지를 확인하여 4 개의 채널 중 2 개의 채널에 문제가 있으면 시스템을 Trip 시키라는 신호를 하위 단계로 넘겨준다.

3) *RT/ESF* : Coincidence Processor 에서 넘어온 Trip 신호를 받아 실제 원자로의 가동을 중지시키라는 명령을 내리는 과정을 시작하게 한다.

특히 Coincidence Processor 에서는 자신의 채널에서 들어온 오류를 처리하기 위해 나머지 채널과 voting 하게 되며 이 voting 은 임의의 채널에서 적어도 하나의 오류가 발견되고 이러한 voting 이 연속으로 4 번 이상 오류임을 감지할 경우 Trip 신호를 하위 단계로 보내게 된다.

만약 이러한 voting 단계에서 나머지 모든 채널에서 오류 신호 발견하지 못할 경우, Coincidence Processor 는 Reset 이 되고 Bistable Trip Processor 의 신호를 기다리게 된다.

본 논문에서 명세 하는 DPPS 는 다음과 같은 가정으로 추상화 시켜 검증을 시행할 것이다. 이는 설계의 Prototype 을 제시하기위해 단순화 시킨 것이다. 실제 시스템은 12 개의 신호(아날로그 10 개, 디지털 2 개)를 처리하지만 본 논문에서 디지털 신호 하나 만을 처리한다고 가정한다.

따라서 Bistable Trip Processor에서는 특정 아날로그 값을 Set-point와 비교하지 않고 디지털 신

호로 오류인지 오류가 아닌지를 내보내 주게 명세한다.

DPPS 가 만족해야 하는 요구 사항은 다음과 같을 수 있다.

Requirement Specification :

Property : Safety_01

$$Always(((ERROR_i \grave{U} ERROR_j) \rightarrow (ERROR_i \grave{U} ERROR_j) \rightarrow (ERROR_i \grave{U} ERROR_j) \rightarrow (ERROR_i \grave{U} ERROR_j)) \rightarrow TRIP)\ (i \neq j,\ 0 \le i,j \le 3))$$

Property : Fairness_01:

$$Always((NOT\_ERROR_i \grave{U} NOT\_ERROR_j \grave{U} NOT\_ERROR_k) \rightarrow SYSTEM\_RESET)\ (i \neq j \neq k,\ 0 \le i,j,k \le 3)$$

Property Safety_01은 만약 2개 이상의 채널에서 에러가 검출된다면 그 즉시 trip신호가 발생해야 함을 요구조건으로 둔 것이며 두 번째 Property Fairness_01은 만약 어떤 채널의 coincidence processor 에서 trip을 판가름하기 위해 기다리고 있다 하더라고 다른 모든 채널에서 에러가 검출되지 않는다면 bistable coincidence와 coincidence processor RESET 시키게 하는 Fairness에 대핸 요구사항을 명세한 것이다. 위와 같은 방법 외에 시스템이 취하게 될 행위에 대한 명세는 시제논리(Temporal Logic)을 이용해서도 명세할 수 있다. 즉 시제 논리를 이용하여 시스템이 취해야 할 상태를 명확하게 명세할 수 있다. 시제 논리로 표현한 요구 사항은 6절의 SPIN의 LTL 명세에서 자세히 볼 수 있다.

5. STATECHART 및 SMV 통한 정형적 설계 및 검증

먼저 도식적인 정형명세 언어인 STATECHART를 이용하여 설계하고 검증하는 것을 논하겠다. STATECHART[5,6]는 Reactive system을 설계하기 위한 가장 널리 알려진 도식적 명세 언어이다. 하지만 각종 도구를 이용하여 이를 설계하는 기법은 논의되어 왔지만 원자력 발전 시스템에 대한 설계에서 요구하는 기능상의 검증은 소원하다. 따라서 이러한 검증 기법을 본 논문에서 보이고 있다.

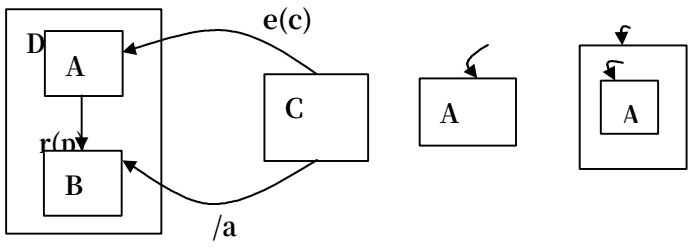

그림 3 STATECHART

그림 4 default state

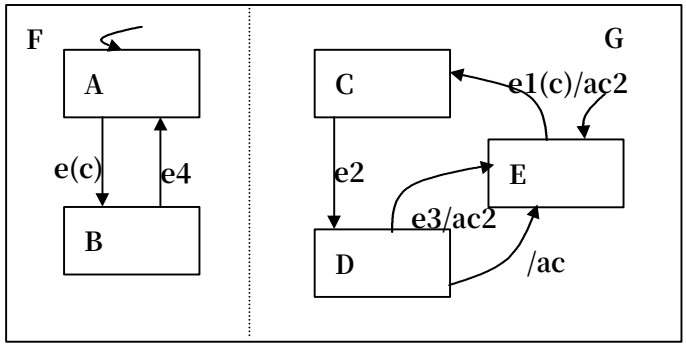

그림 5 병렬적인 STATECHART의 예

5.1. STATECHART

STATECHART(Harel 1987)는 reactive 시스템의 명세를 위하여 개발된 정형명세 언어로써 기존

의 유한 상태 오토마타와 상태 전이 다이어그램의 확장된 형태의 상태 기반 도식적 정형 명세 언어이다. 그 근본구조는 state transition diagram에 높이(hierachy),동시성(concurrency),통신(communication)의 개념을 도입한 것이다. 높이는 트리에서 sub-tree가 파생되듯, 하나의 스테이트가 서브 스테이트들을 가질 수 있게 하는 것으로 계층적 구조를 표현한다. 동시성은 스테이트들 간에 병렬성을 제공하는 것인데, 하나의 스테이트가 여러 개의 병렬 스테이트로 구성된. 통신은 시스템의 컴포넌트들간에 방송(broadcasting)을 통해 정보를 주고받으면서 시스템이 진행하는 것을 의미한다. STATECHART에는 3종류의 스테이트가 있다. OR-state는 스테이트에 높이를 제공하는 것으로서 하나의 스테이트 하나의 스테이트가 여러 서브 스테이트들을 가질 수 있다. 그 서브 스테이트들간에는 'exclusive-OR'관계가 성립하는데 이것의 의미는 서브 스테이트들 중 단 하나만 active 될 수 있음을 나타낸다. <그림 3>은 간단한 STATECHART를 보여준다. 이 다이 다이아그램에서 사각형은 각 상태를 나타내고 화살표는 전이를 나타낸다. STATECHART에서 전이 레이블(label)의 문법은 e(c)/a이다. e는 전이를 발생시키는 이벤트이고 (c)는 이벤트가 발생했을 때 전이를 가능하게 하는 조건문이다. a는 전이가 일어날 때 발생되는 액션을 가리킨다. 각 상태간의 계층은 encapsulation을 사용한다. <그림 3>에서 상태 A와 상태 D는 서로 자식과 부모간의 계층관계를 가지고 있고 서로 exclusive-or관계로 같은 시간에 서로 다른 상태에 제어가 있어서는 안 된다. <그림 4>에서 각 상태 붙어있는 작은 화살표는 default 상태가 된다. <그림5>에서는 동시성을 보여주고 있다. 즉 가운데 점선은 상태 F와 G가 "AND"관계에 있음을 보여준다.

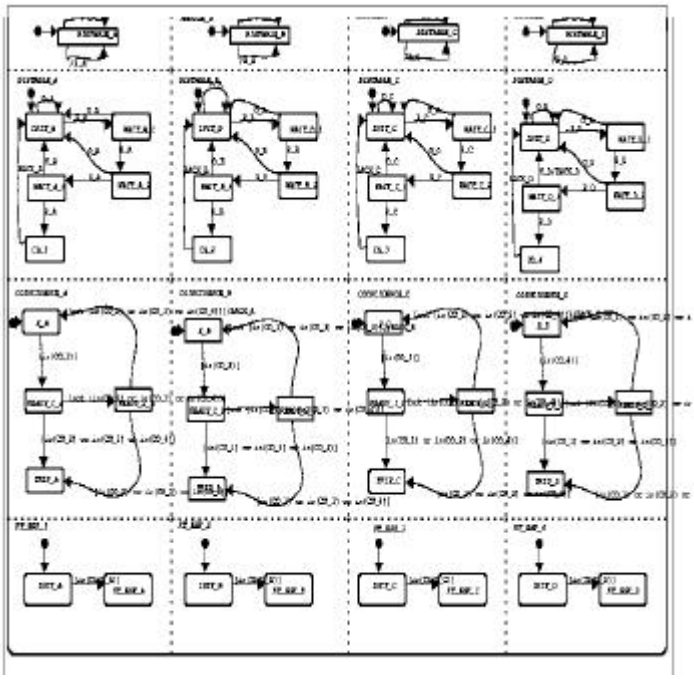

그림 6 STATECHART를 이용한 DPPS명세

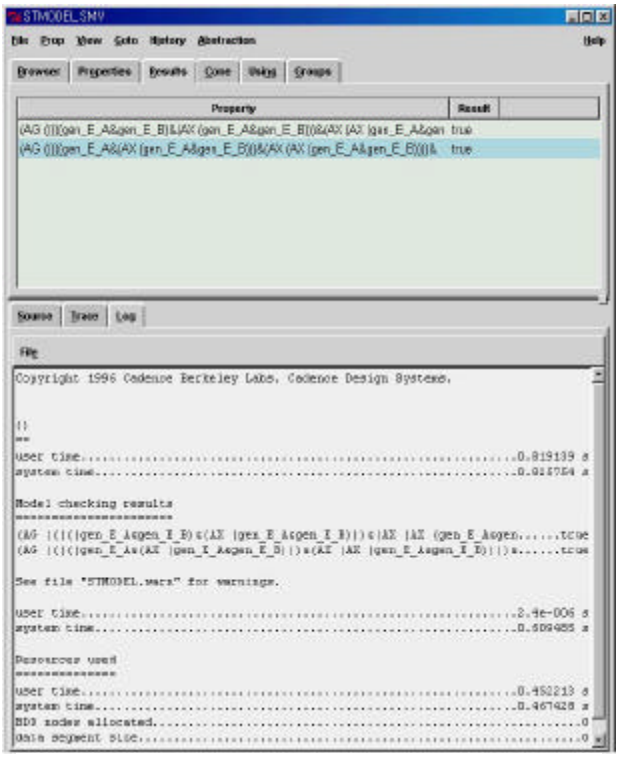

그림 7 SMV를 이용한 검증 결과

즉 상태 F와 G가 동시에 있는 것을 의미한다.

5.2 모델체킹과 SMV

5.2.1 모델체킹

본 논문에서 STATECHART로 설계된 시스템을 검증하기 위해 사용하는 검증 기법이 모델 체킹이다. 모델 체킹(Model-Checking)[3]이란 유한 상태 동시 시스템을 검증하기 위한 오토마타 기반의 기술이다. 상태 전이 시스템(state transition system)과 특성(property)이 주어지면, 모델 체킹 알고리즘은 시스템이 원하는 특성을 만족하는지를 알아 보기 위해서 전체 상태 공간(state space)를 검사한다. 즉 시스템이 가질 수 있는 모든 상태를 오토마타로 표현하고 이 오토마타가 사용자의 요구 사항을 만족시키는 검사하는 자동화된 기법을 가리킨다. 따라서 기존의 모의 실험, 테스팅 및 연역적 추리를 기반 하여 시스템을 검사하는 방법보다 뛰어난 검증 방법이라 할 수 있다. 본 논문에서 STATECHART를 검증하게 될 모델체킹 시스템은 SMV[7]로서 CTL(Computational Tree Logic)의 시제논리를 기반으로 한 모델 체커이다. 시제 논리(temporal logic) 모델 체킹은 검증하려고 하는 소프트웨어나 하드웨어 시스템을 모델링한 상태 전이 시스템과 특정 시제 논리로 표현된 특성을 입력 받아서, 시스템이 그 특성을 만족하는지를 검증하는 것이다.

5.3 DPPS의 STATEHCHART-SMV의 설계 및 검증

본 논문에서는 이 DPPS에 대한 STATECHART의 명세는 3개의 부분으로 나누어져 있다. 첫째, 입력 시그널로 들어오는 아날로그 변수와 디지털 변수에 대한 이상여부를 확인하는 bistabple trip processors 부분. 둘째, 다른 채널과 brocasting하면서 실제적으로 시스템에 이상이 있는지 확인해 주는 coincidence processors 부분, 마지막으로 RT, ESF 부분은 실제로 어떤 역할을 수행하느것이 아니라 단지 coincidence processor로 발생한 trip signal을 받는지 까지만 명세에 포함시켰으며 bypass또한 시스템의 초기단계에서부터 영향을 미치는 부분으로 DPPS에 포함시키지 않았다. 또한 이 논문에서 세운 가정은 A, B, C, D 4개의 채널이 입력시그널을 받아들일 때의 각각의 채널에 대한 지연시간을 50ms까지만 허용하는 것으로 설정하였다. STATECHART에서 25ms는 한 step으로 처리하였다. <그림6>는 이 DPPS를 STATECHART를 이용하여 명세한 부분이다. 본 논문에서는 검증 코드를 만들기 위해 원래의 STATEMATE의 STATECHART의 문법에 얼마의 제약을 가하여 이를 SMV로 바꾼 후 시스템이 요구하는 사항을 CTL로 바꾸어 검증하였다. 단 이미 각각의 채널이 처음 입력 신호를 받을 때 가질 수 있는 시간 지연은 명세상에서 두 step까지 허용하는 것으로 하였다. 예를 들어 bistable에서 입력신호를 받을 때, C, D채널에서는 모두 정상 신호가 들어오고 A에서는 B채널에서 보다 에러 신호를 2 step 먼저 입력을 받게 되면 A채널에 있는 coincidence processor는 trip 을 발생시키지 않게 된다. <그림 7> 에서 나타난 바와 같이 SMV로 검증한 결과 위의 요구사항을 만족시킴을 알 수 있다.

6. SCR 및 SPIN통한 정형적 설계 및 검증

6장에서는 테이블 기반의 정형명세 언어인 SCR[8]을 통한 설계와 이에 대한 검증의 예를 보일 것이다.

정형적 명세를 위해 Requirement와 SRS, SDD 사이의 관계를 살펴보면 <그림 8>과 같이 나타낼 수 있다. Requirement는 자연어로 많이 기술되어 있기 때문에 그 모호성과 불완전성이 많이 존재하게 된다. 그렇기 때문에 정형 명세를 통해 SRS로 표현하고, 이때 requirement와 SRS간의 완전성과 일관성 검사가 꼭 필요하다. 이를 위해 정형명세 도구들이 사용된다. 또한 그 명세에 requirement를 정확하게 표현하고 있는지를 검증하여야 한다. 이를 위해 정형 검증 도구들이 사용된다. 다시 말하면, 자연어로 기술되어 모호하고 불완전한 requirement를 정형명세도구로 완전성과 일관성을 검사하고, 그 특성을 확인(property check)한 다음 정형검증도구로 V&V를 하는 것이다. 자연어의 모호성과 불완전성을 해결하고 나면 시스템의 설계를 시작하게 된다. 물론 이때도 설계에 대한 완전성과 일관성 검사와 특성 확인 및 V&V 과정이 필요하다. 때문에 정형기법 도구들이 사용된다. 하지만 이렇게 정형 명세와 정형검증을 하였다 하더라도 그것은 SDD에 관한 명세와 검증일 수밖에 없기 때문에 SRS와 SDD간의 compliance를 검사해야 한다. 즉 SRS에서 SDD로 변환되면서 SRS와 SDD가 서로 일치하는지에 대해 확인할 필요가 있다는 것이다.

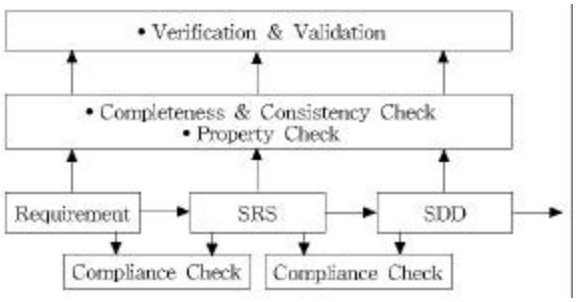

그림 8 정형명세를 위한 SRS와 SDD와의 관계

이러한 방법을 사용하면 자동화된 다양한 정형기법을 적용하기 때문에 빠른 시간에 여러 특성들을 여러 방면으로 검토할 수 있고, SRS와 SDD와의 일관성까지 확인할 수 있기 때문에 각 단계에서 오류가 없음을 효율적으로 증명할 수 있다.

6.1 SCR(Software Cost Reduction)

SCR은 NRL에서 A-7 항공기 비행 프로그램의 요구명세를 위해 개발되었다. 즉, SCR은 시스템이나 소프트웨어의 요구명세를 도와주기 위해 만들어진 정형검증 도구이다. 이 NRL의 연구는 실제적인 정형명세에 초점이 맞추어져 있다. SCR은 내장형 시스템, 실시간 시스템과 소프트웨어를 위한 시스템 요구사항을 명세하거나 그러한 명세를 위한 도구와 방법들을 평가한다.

SCR에서 하드웨어 인터페이스는 입력과 출력 데이터와 시스템 함수로 기술된다. 이러한 시스템 함수는 mode transition tables, event tables, condition table과 같은 표의 집합으로 이루어져 있고 condition, events, modes, mode classes와 같은 높은 단계의 구조를 지원한다. 또한 SCR을 이용한 정형검증 방법은 시스템이나 소프트웨어의 상태와 그에 관련된 변수들을 표로 표현하여 요구명세를
검증한다. SCR은 단순히 시스템이나 소프트웨어의 요구사항을 명세 할 뿐만 아니라 명세한 요구사항들의 완전성과 일관성을 체크한다. 또한 SCR은 명세한 표의 변수를 가지고 시스템이나 소프트웨어를 시뮬레이션 할 수도 있다.

SCR의 DPPS를 명세함에 있어서 DPPS의 모든 변수와 모든 채널을 모두 명세 하였을 경우 본 논문에서 SCR 설계의 검증을 위해 사용할 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 채크할 수 있다고 판단하고 입력 값을 각 채널 당 하나로 잡았다.

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 변수 Table

각 변수 table은 표1과 같다. 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 넘어서더라도 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_out의 값을 살펴보아 2개 이상의 값이 true가 되면 true로 된다. 각 controlled variable을 살펴보면 <표 2>와 같다. 각 state의 mode class table은 <표 3>와 같이 표현된다.

```
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.
```

표 2 Controlled Variable

6.2. SPIN

SPIN[9]은 AT&T Bell 연구소에서 1995년에 발표한 LTL 모델체커이다. 기존의 모델 체킹 방법을 개선하기 위해 On-the-Fly 방식을 사용하여 메모리의 효율적인 사용을 가능하도록 하였다. SPIN은 비동기 프로세스 시스템(asynchronous process system)의 설계와 검증(verify)을 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 통 신 시스템, 교환 시스템(Switching Systems), 동시성 알고리즘(Concurrent algorithms), 철도 신

Source	Event	Destinatio
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 3 Mode Class Table

호 프로토콜(railway signaling protocols) 등 실제적인 시스템의 디자인에 매우 적합한 것으로 평가되고 있다.

이 도구를 이용해서 명세(specification)의 일관성(consistency)을 검사할 수 있는데 데드락(Deadlock), 명세되지 않은 반응(Unspecified receptions), flags incompleteness, 경쟁 조건(race conditions)등을 잘 보여준다. 선형시제논리(Linear Temporal Logic:LTL)이나, next-time free LTL, 또는 Buchi Automata 를 이용해서 정확성(correctness properties)을 검증한다.

SPIN은 PROMELA(Process Meta Language)라는 검증 언어를 사용하여 설계를 하고 PROMELA는 LTL의 구문으로 명세화 하여 정확성을 증명한다. PROMELA는 검증 모델 언어(Verification modeling language)이고, 분산 시스템(distributed system)이나 프로토콜의 추상모델(abstraction)을 제공하고 프로세스(processes), 메시지 채널(message channels), 그리고 변수들로 구성되어 있다. 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. 자동화된 SPIN 코드를 검증하기 위하여 DPPS의 가장 중요한 특성을 살펴보자면 다음과 같이 정리할 수 있다. 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)
```

bypass가 false인 경우 두 채널 이상에서 입력 값이 한계값을 넘어서면 trip이 발생하여야 한다.

이러한 요구 사항을 시제논리의 일종인 LTL로 나타낸 이 특성은 SPIN에서 valid하다고 나왔고, DPPS의 명세는 올바르게 시스템의 요구를 만족한다고 볼 수 있다.

그림 9 LTL로 표현한 요구 사항

7. ESTEREL을 통한 정형적 설계 및 검증

7.1. ESTEREL 설계 시스템

ESTEREL[10]은 프랑스의 Sophia-INRIA에서 개발 연구 되고 있는 정형 명세 및 검증 시스템이다. ESTEREL 명세 언어는 하드웨어나 프로토콜 등을 명세하기 위해 개발된 Reactive 시스템 모델링 언어로써 Reactive 시스템의 동기적 프로그램을 작성하는데 사용하는 동기적 언어(Synchronous Language)이다. 동기적 언어는 완벽한 동기적 동시성 모델에 근거하며, 이러한 모델 내에서는 동시적 프로세스가 0 시간 내에 계산과 정보의 교환이 일어난다. ESTEREL은 현재 프랑스 INRIA 연구소에서 연구되고 있으며 다양한 정형 검증 도구가 개발되고 있다. 이러한 정형검증 도구로는 XEVE [11]있는데 이 검증도구는 모델 체킹 기법을 이용하여 시스템을 검증하고 있다. ESTEREL의 설계 시스템 가운데는 XES[12]라는 검사 도구가 있는데 이 시스템은 명세 된 시스템을 설계자나 구현자가 Interactive 하게 시뮬레이션 하는 기법을 제공한다. ESTEREL을 이용하여 <그림 10> 같이 정형기법을 사용하여 시스템의 설계할 수 있다. <그림 10>에서와 같이 설계된 설계도는 설계 단계에서 오류를 찾을 수 있기 때문에 오류가 없는 설계를 통해 구현자는 보다 완벽한 구현을 할 수 있다. 이러한 기법은 특히 안전성과 신뢰성이 필요한 곳에 적용될 수 있다.

7.2 DPPS의 검사 및 검증

<그림 11>과 같이 명세한 DPPS는 XES와 XEVE를 통해 검사하고 검증할 수 있다. XES는 시스템이 올바로 구현되었는지 검사하는 도구이다. 즉 이 단계에서 Simulation을 통한 검사가 가능하게 된다. 이렇게 검사된 시스템은 사용자의 요구에 따라 특정 성질(property)을 만족하는지를 검증하는 도구가 XEVE다.

XES는 사용자는 임의 테스트 값을 집어 넣음으로 시스템이 올바른 신호를 내는가를 검사한다. <그림 11>과 같이 명세한 DPPS는 사용자가 신호 입력 부분에 임의 신호를 입력함으로 시스템이 올바로 작동하는 지를 검사하게 된다. 즉 시스템이 올바르게 만들어졌는가를 검사하게 된다. 본 논문에서 각각의 모듈을 검사하고 전체를 concurrently하게 놓고 검사한 결과 올바르게 작동하였다. XEVE로는 시스템을 검증할 수 있다. XEVE는 시스템이 만족해야 하는 성질을 위배하는 경우 발생하는 위배신호(Illegal Signal)가 나오지 않음을 보여줌으로 검증할 수 있다. 즉 위배 신호가 “절대 나올 없다(NEVER EMIITTED)”를 보여줌으로 위의 성질을 위배하는 경우가 위의 설계에서는 존재하지 않음을 보여준다. 만약 특정 상황에 도달하면 위배 신호(Illegal Signal)가 발생하면 시스템이 사용자의 요구 조건을 충족시키지 않음을 보여주는 것이다.

본 논문에서 검증한 DPPS는 두 요구 조건 Safety와 Fairness를 위배하는 어떤 행위도 하지 않음을 확인할 수 있었다.

본 논문을 통해서 원자력 발전소의 내장형 시스템을 설계하는데 있어서 정형적으로 설계함으로 설계 상의 오류를 제거함으로 완벽한 시스템을 설계한 설계도를 만드는 기법을 제시한다.
본 논문에서 제시한 ESTEREL을 이용한 정형적 설계는 다음과 같은 특징을 지니고 있다.

첫째, 원자력 발전소의 내장형 시스템인 DPPS와 같이 안정성과 신뢰성이 절대적으로 요구되는 시스템을 설계 하는데 있어서, 설계 단계에서 설계도 측면에서의 안정성과 신뢰성을 완벽하게 검사하고 검증하는 정형기법적 설계를 제시하여 이러한 기법을 적용시킨 설계도를 통해 구현자는 보다 수월하게 신뢰성과 안정성이 있는 시스템을 개발할 수 있다.

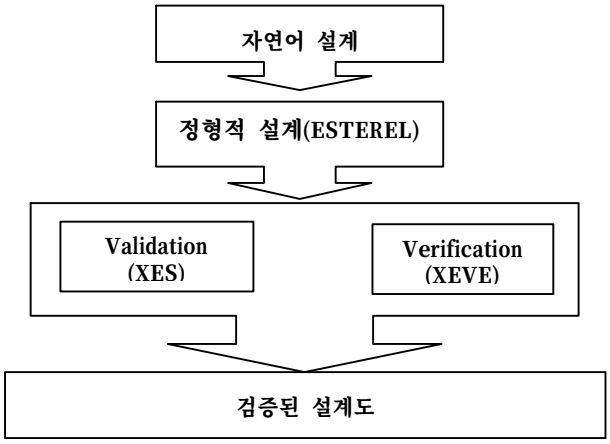

그림 10 ESTEREL을 이용한 정형 설계

```
module DPPS:
input A_NOR, A_ERR;
input B_NOR, B_ERR;
input C_NOR, C_ERR;
input D_NOR, D_ERR;
output SYSTEM_TRIP;
output SYSTEM_RESET;
…

loop
    run A_BIST/BIST [signal A_NOR/BIST_NOR,
                           A_ERR/BIST_ERR,
                           A_NOR_O/BIST_NOR_O,
                           A_ERR_O/BIST_ERR_O]
||
    run A_COINCIDENCE/COINCIDENCE
                    [signal  A_ERR_O/COIN_ERR,
                           B_ERR_O/OTH_ERR_1,
                           C_ERR_O/OTH_ERR_2,
                           D_ERR_O/OTH_ERR_3,
                           B_NOR_O/OTH_NOR_1,

C_NOR_O/OTH_NOR_2,
                           D_NOR_O/OTH_NOR_3,
                           SYSTEM_TRIP/SYS_TRIP,
                           SYSTEM_RESET/RESET]
||
    run B_BIST/BIST [signal B_NOR/BIST_NOR,
                           B_ERR/BIST_ERR,
                           B_NOR_O/BIST_NOR_O,
```

그림 11 ESTEREL로 명세한 DPPS의 일부

둘째, 본 논문에서 제시한 설계 기법은 설계자와 구현자가 사용자의 요구를 보다 정확히 애매모호함 없이 이해 할 수 있게 한다. 따라서 설계자는 요구 사항을 정확하게 이해하여 설계할 수 있고 그러한 구현자는 보다 명확한 설계도를 가지고 구현할 수 있다.

셋째, 본 논문은 기존의 원자력 발전 내장형 시스템의 정형적 설계 기법과는 다른, 용이하면서도 안정된 시스템을 개발하는 정형적 설계 기법을 제시하고 있다. 원자력 발전소 시스템을 설계하는데 있어서 사용되는 설계 기법으로는 SCR을 이용한 시스템 검사(validation) 기법과 PVS의 Theorem Proving을 이용한 시스템의 행위적 측면에서의 검증이 있다. SCR의 경우는 시스템이 서술적인 명세에서 정형화된 설계로 옮겨진 후 정형화된 설계가 올바른지를 판단할 수 있는 기능을 제시한다. 특히 Duplicate variable names, Unspecified dependent variables(i.e., controlled variables, terms, mode classes), Incorrect syntax의 검사 혹은 Definition, specification, 또는 변수의 사용에 있어서 타입의 일관성을 검사함으로 설계를 검사하게 된다. 따라서 행위적 검증은 상대적으로 용이하지 않다. 또한 PVS는 Theorem Proving을 이용하기 때문에 설계자가 설계를 검증하기 위해서는 상당한 논리학적 지식이 필요하다. 하지만 본 논문에서 사용한 ESTEREL 설계 기법은 함수형 언어로서 설계한 다음 검사 및 검증(V&V)을 실시할 수 있으므로 보다 용이하게 설계자가 좋은 설계도를 만들 수 있다는 이점을 가지고 있다.

단점으로 ESTEREL은 모델 체킹을 기반으로 검증을 수행하기 때문에 시스템의 복잡도에 따라 상태 폭발로 인한 ESTEREL의 검증 시간이 커질 수 있다는 단점을 가지고 있다.

또한 행위적인 명세에 시간을 표현할 경우 시간의 흐름을 표현하기에는 어려움을 가지고 있다.

8. 결론 및 향후 연구 방향

본 논문에서는 논하고 있는 원자력 발전 내장형 Protection 시스템에 대한 설계의 특징은 다음과

같다.

첫째로 국제 원자력 발전 시스템의 개발 가이드라인에 근거한 시스템의 설계 및 분석에 기법을 제안하고 있다. 두 번째로 미국의 공인된 High-integrity 소프트웨어의 설계 기법을 근거로 정형기법을 적용하였으며 따라서 설계상의 애매모호함을 줄이며 또한 설계도의 오류를 발견함으로 실제 구현 단계에서의 오류 발생을 최소화하는 설계도의 V&V 기법을 제안하고 있다. 세 번째로 이러한 설계 상의 정형기법을 통한 V&V의 적용은 시스템의 전반에 걸친 오류를 극소화하여 내장형 소프트웨어의 안정성화 신뢰성을 극대화할 수 있으며 이를 통해 전반적인 시스템 개발의 안정화를 기대할 수 있다.

향후 연구 방향으로는 원자력 발전 내장형 Protection 시스템의 설계에 대해 보다 심도 있는 정형기법의 적용을 들 수 있으며, 그로 인해 실제와 가까운 시스템을 설계하여 분석함으로 그 가능성을 연구해야 할 것이다. 또한 설계도를 통한 구현자의 구현으로 생기는 오류를 최소화하기 위해 가능한 설계도를 구현에 이르게 하는 자동화된 구현 기법이 연구되어야 할 것이다.

9. 참고 문헌

[1] Debra S. Herrmann, “Software Safety and Reliability”, IEEE Computer Society, 1999, p 226-269

[2] 임성묵, ACSR-VP를 이용한 실시간 시스템의 정형 명세와 검증, p 3-4, 고려대학교 컴퓨터학과, June. 1998

[3] Edmund M. Clake, Orna Grumberg, Doron A. Peled. Medel Checking. The MIT press. 1999

[4] Dolores R. Wallace and Laura M. Ippolito, "A Framework for the Development and Assurance of High Integrity Software", *NIST 500-223*, Dec. 1994.

[5] David Harel, STATECHART: A VISUAL FORMALISM FOR COMPLEX SYSTEMS, Science of Computer Programming 8 (1987) pp231-274

[6] David Harel and Ammon Naamad, The STATEMATE Semantics of STATECHARTs, ACM Trans. Soft. Eng. Method. Oct. 1996

[7] K. L. McMillan. Symbolic model checking - an approach to the state explosion problem. PhD thesis, SCS, Carnegie Mellon University, 1992

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

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

[10] Gerard Berry. The Foundations of Esterel, INRIA, France, Milner Lecture at Edinburgh University. 1996

[11] Amar Bouali. XEVE: an Esterel Verification Environment(Version v1_3)

[12] G.Berry and The Esterel Team. The Esterel v5_21 System Manual. INRIA. France. March. 11. 1999