정형 소프트웨어 요구사항으로부터 PLC 디자인의 체계적 생성

(Systematic Generation of PLC-based Design from Formal Software Requirements)

유 준 범 [†] 차 성 덕 [††] 김 창 회 [†††] 송 덕 용 [††††]

(Junbeom Yoo) (Sungdeok Cha) (Chang Hui Kim) (Deokyong Song)

요 약 원자력 발전소의 디지털 제어 시스템은 안전성이 중요시되는 safety-critical 소프트웨어로서 충분한 수준의 안전성을 보장하기 위해서 여러 기법들이 적용되고 있다. 특히, 정형명세 기법은 개발의 초기 단계에서 소프트웨어 요구 사항들을 명확하고 완전하게 명세하도록 유도함으로써 안전성을 크게 향상시킬 수 있는 기법으로 인정받고 있다. 본 논문에서는 정형명세 기법인 NuSCR을 이용해서 작성된 요구사항 명세로부터, 설계 단계의 내용으로 사용될 수 있는 PLC 기반의FBD 프로그램을 체계적으로 생성하는 기법을 제안하고 있다. 제안된 기법은 기존의 수동 명세 작업에서 발생할 수 있는 오류들을 크게 줄일 수 있으며, 소프트웨어의 개발 비용과 기간을 줄일 수 있다. 또한, 제안된 기법의 유용성을 증명하기 위해서, 현재 KNICS에서 개발 중인 DPPS RPS의 BP를 구성하는 트립 논리 중의 하나인 고정 설정치 상승 트립을 예제로 설명하고 있다.

키워드 : 정형 요구사항 명세, 디자인 명세, PLC, 원자력 발전소 디지털 제어 시스템

Abstract The software of the nuclear power plant digital control system is a safety-critical system where many techniques must be applied to it in order to preserve safety in the whole system. Formal specifications especially allow the system to be clearly and completely specified in the early requirements specification phase, therefore making it a trusted method for increasing safety. In this paper, we discuss a systematic method, which generates PLC-based FBD programs from the requirements specification using NuSCR, a formal requirements specification method. This FBD programs takes an important position in design specification. The proposed method can reduce the possible errors occur in the manual design specification, and the software development cost and time. To investigate the usefulness of our proposed method, we introduce the fixed set-point rising trip example, a trip logic of BP in DPPS RPS, which is presently being developed at KNICS.

Key words : formal requirements specification, design specification, PLC, nuclear power plant digital control system

1. 서 론

최근의 10여 년 동안 원자력 발전소 제어 시스템 분야에서는 소프트웨어 안전성[1]을 특히 중요하게 요구하고 있다. 이는 기존에 제작되었던 RLL(Relay Ladder Logic) 기반의 아날로그 하드웨어 제어 시스템들을 소프트웨어 기반의 디지털 제어 시스템으로 교체하고 있기 때문이다[2]. 따라서, 정형명세 기법[3]을 통해서 제어 소프트웨어의 안전성을 초기 단계에서 향상시키기 위한 노력이 요구되고 있으며, 원자력 발전소의 제어 시스템 소프트웨어를 명세 하는데 적합하게 수정보완된 정형명세 기법을 개발하기 위한 노력도 꾸준히 진행중이다[4-6]. NuSCR[7]은 이와 같은 목적으로 개발된 정형명세 기법으로서 현재 KNICS[8]에서 개발하고 있는 디지털 발전소 보호시스템(Digital Plant Protection System: DPPS)의 노심보호시스템(Reactor Protection System: RPS)을 정형명세 하는데 사용되고 있는 기법

† 학생회원 : 한국과학기술원 전자전산학과
jbyoo@dependable.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과 교수
cha@dependable.kaist.ac.kr
††† 비 회 원 : 한국원자력연구소 I&C-HMI 팀
chkim2@kaeri.re.kr
†††† 비 회 원 : ㈜액트 엔지니어링 사업팀
dysong@actbest.com
논문접수 : 2003년 6월 25일
심사완료 : 2004년 11월 18일

이다. NuSCR은 원자력 발전소의 제어 시스템 SW를 명세 하는데 유용한 정형명세 기법으로 평가 받고 있다 [9].

원자력 발전소의 제어 시스템 소프트웨어를 개발하기 위한 개발 단계를 살펴 보면 다음과 같다. 개발 단계는 일반적인 소프트웨어 개발 단계와 마차가지로 분석, 설계, 구현, 테스팅으로 구분할 수 있다. 먼저 분석 단계에서는 자연어로 요구사항 명세를 작성한다. 다음의 설계 단계에서는 실제로 구현하기에 앞서 요구사항 명세의 내용을 보다 자세하게 기술하는 단계이다. 원자력 발전소의 제어 소프트웨어들은 모두 PLC(Programmable Logic Controller)를 기반으로 하는 내장형 소프트웨어이므로, 설계 명세에서는 하드웨어인 PLC에서 소프트웨어 요구 사항들이 어떻게 구현되는 가를 명세하여야 한다. 이를 위해서는 PLC를 구동하기 위해서 사용되는 프로그래밍 언어인 SFC(Sequential Function Chart), LD(Ladder Diagram), FBD(Function Block Diagram) [10] 등이 주로 이용된다. 구현 단계에서는 PLC에 하드웨어 구성을 결정한 후, PLC 제작사들이 제공하는 엔지니어링 도구를 이용해서 SFC, LD, FBD 등으로 작성된 프로그램을 PLC가 인식할 수 있도록 변환하는 작업이 수행된다. 이는 컴파일러에 의해서 여러 단계를 거쳐 자동으로 수행되는 작업이므로, 소프트웨어 개발자는 설계 단계에서 PLC 대상의 프로그램을 완성했을 때, 실질적으로 소프트웨어의 개발이 완료된 것으로 생각할수 있다. 마지막으로는 테스팅 단계에서는 설계 단계의 PLC 프로그램에 대한 직접 테스팅 보다는 구현 단계에서 컴파일러에 의해서 생성되는 중간 산출물인C 프로그램을 대상으로 간접 테스팅을 수행한다. 이는 아직까지는 PLC 프로그램 자체를 대상으로 하는테스팅 기법에 대한 연구가 부족하기 때문이다.

이와 같이 PLC를 대상으로 하는 소프트웨어는 설계 단계가 완료되면 실질적인 소프트웨어의 개발이 완료된 것으로 간주할수 있다. 이와 같이 중요한 역할을하는 설계 단계는 미리 작성된 요구사항 문서를 바탕으로 개발자가 수동으로 작성하는 방법이 주로 사용되는데, 이는 전 단계의 요구사항 문서가 자연어로 기술되어 있음에 기인한다. 앞서 언급한 바와 같이, 현재 개발중인 KNICS의 제어 소프트웨어들은 정형명세 기법인 NuSCR을 이용해서 정형명세 되었다. NuSCR은 자연어와는 달리 정해진 형식과 의미를 지니므로, NuSCR 명세를 기준으로 이와 동일한 행위를 하는 PLC 프로그램을 체계적으로 생성할 수 있다면, 기존의 수동으로 하던 명세 작업에서 발생하던 오류들을 크게줄일 수 있으며 또한, 소프트웨어의 개발 비용과 기간을 크게 줄일 수 있게 된다.

본 논문에서는 정형명세 기법인 NuSCR을 이용해서 작성된 정형 요구사항 명세로부터, PLC 프로그래밍 언어 중에서 가장 널리 사용되는 FBD 프로그램을 체계적으로 생성할 수 있는 기법을 제시하고 있다. 생성된 FBD 프로그램은 설계 명세로서 유용하게 사용될 수 있다. 또한, 제안하는 기법의 유용성을 보다 명확하게 하기 위해서, 현재 KNICS에서 개발 중인 DPPS RPS의 BP(Bistable Processor)를 구성하는 트립 논리중의하나인 고정 설정치 상승 트립 논리를 예제로 하여 설명하고 있다. 논문의 추후 구성은 다음과 같다. 2장에서는 원자력 발전소의 제어 소프트웨어들이 구동 되는 환경인 PLC의 특성과 PLC 프로그래밍 언어인 FBD에 대해서 소개하겠다. 3장에서는 정형명세 기법인 NuSCR에 대해서 간단히 소개하겠다. 보다 자세한 정보는 [9,15]을 통해 얻을 수 있다. 4장에서는 NuSCR 정형명세로부터 FBD 프로그램을 생성하는 기법을 실제 예제를 중심으로 소개하겠다. 마지막으로 5장에서는 결론 및 제안된 기법을 바탕으로 계획하고 있는 향후 연구에 대해서 언급하겠다.

2. Programmable Logic Controller

PLC(Programmable Logic Controller)는 철도 교차로 차단기, 교통신호 제어 시스템, 생산라인 등과 같이 실시간의 문제 처리가 요구되는 산업에서 널리 사용되고 있는 시스템이다. PLC는 다음과 같이 그 특징을 설명할 수 있다[11,12].

간단한 하드웨어 아키텍쳐: PLC는 다양한 입출력의 가감을 용이하게 하기 위해서 비교적 간단한 구조로 구성되어 있다. 특히, 이러한 하드웨어 구조를 바탕으로 프로그램의 수행 시간을 정확하게 분석할 수 있도록 구성되어 있다. 이러한 PLC에는 이를 구동할 수 있는 OS가 제공되며, OS 상에서 동작하는 응용 프로그램은 SFC, LD, FBD 등을 이용해서 작성된다.

프로그램 실행 체계: PLC와 기존의 일반적인 시스템과의 가장 큰 차이점은 바로 program execution mechanism일 것이다. PLC 프로그램은 주기적으로 무한히 수행되게 되는데, 입력들을 읽고, 필요한 연산을 수행한 후, 출력을 내보내는 작업을 각 주기(scan cycle) 마다 반복하게 된다. 또한, PLC 프로그램들은 자신의 수행 시간을 정확하게 ms 단위로 분석할 수 있어야 하며, 전체 수행 시간을 모두 포함할 수 있는 시간이 주기로서 사용된다.

IEC 61131-3에 의해 정의된 프로그래밍 언어들: PLC 프로그램은 IEC 61131-3에서 지정된 ST, LD, IL, SFC, FBD의 5 가지 프로그래밍 언어를 사용해서 작성된다. 각각의 프로그래밍 언어들은 나름대로의 장단점이

존재하며, 이 중에서 가장 널리 사용되는 것은 LD와 FBD이다. 현재 KNICS에서는 FBD만을 사용하고 있으므로, FBD의 특징을 설명하면 다음과 같다. FBD는 전자회로 다이어그램(electrical circuit diagram)과 유사한 형태의 graphical diagram으로서 단위 기능을 수행하는 각각의 FB(Function Block)들의 network으로 표현된다. 따라서, FBD는 시스템을 FB을 이용한 정보의 흐름으로 표현하고 있는 PLC 프로그래밍 언어로 볼 수 있다. 다음의 그림 1은 FBD 프로그래밍에서 널리 사용되는 기본적인 FB들의 예로서 각 분류의 대표적인 FB를 설명하고 있다. 기본적인 FB에는 논리 연산, 사칙 연산 및 선택 기능, 타이머 등이 모두 포함되며, 원자력 발전소의 제어 소프트웨어에 대한 FBD 프로그램은 아래의 기본적인 14 종류의 FB를 이용하면 충분히 작성할 수 있는 것으로 인식되고 있다. 이는 원자력 분야에서는 기본적인 FB만을 사용함으로써 해석 및 분석을 용이하게 하여 생성된 FBD의 안전성을 제고하도록 권고하기 때문이다.

3. NuSCR

NuSCR[7]은 기존에 AECL(Atomic Energy of Canada Limited)에서 제시한 SCR에 기반 하는 방법론[5]을 수정, 보완하여 원자력 발전소의 제어 시스템 소프트웨어를 명세 하는데 보다 유용하게 사용될 수 있도록 개발된 정형명세 기법이다. 기본적으로 SCR[13]과 마찬가지로 Parnas′ Four-Variable Model[14]에 기초하며, 이 모델에서 정의되는 네 변수 간의 관계를 보다 명확하게 명세하기 위해서 세 가지의 추가적인 변수 모델을 사용한다. 이는 function variable, history variable, timed-history variable로서 각각SDT(Structured Decision Table), FSM(Finite State Machine), TTS (Timed Transition System)[15]를 이용해서 표현된다.

각각의 변수 모델들은 원자력 발전소의 제어 소프트웨어의 다양한 행위를 명세 하는데 유용하게끔 특징지어져 있다. Function variable은 수학적인 함수 관계를 표현하는데 사용되며 테이블 형태의 SDT를 사용한다. History variable은 수학적인 함수 보다는 상태의 흐름을 중심으로 명세할 때 보다 쉽게 명세 되는 내용을 표현할 때 사용되며, 오토마타의 일종인 FSM으로 표현된다. 또한, Timed-history variable은 시간제약 조건이 있는 내용을 명세할 때 사용되며, 시간 개념이 추가된 오토마타의 일종인 TTS를 사용한다. NuSCR은 이와 같이 서로 다른 특성을 지니는 변수들을 혼용함으로써 명세 대상을 보다 이해하기 쉽고 정확하게 명세할 수 있다는 특징을 지닌다[9]. 각 변수 노드들 간의 상관관계는 DFD(Data Flow Diagram)의 일종인 FOD (Function Overview Diagram)에 의해서 표현된다. 각각의 변수들은 FOD 상에서 하나의 독립된 노드로 표현되며, 각 노드들은 입력과 출력을 갖는다. 또한 FOD는 계층적으로표현된다. 각 변수들과 FOD를 포함한 전체 시스템에 대한정형 정의는 [7]에 자세히 제시되어 있다.

FOD(Function Overview Diagram) 그림 2는 *g_Fixed_Setpoint_Rising_Trip_with_OB*에 대한 전체 FOD 로서, *"g_"*는 FOD의 계층 구조 내의 중간 단계 노드임을 의미한다. 그림 2의 FOD는 고정설정치 상승 트립 논리에 대한 FOD이며, 이 FOD가 지니는 정확한 의미에 대해서는 4장에서 설명하겠다. 좌우의 사각형으로 표현된 노드들은 FOD의 hierarchy 상에서의 입출력을 의미한다. NuSCR의 FOD에서는 각 노드들의 특성을 이름과 노드의 모양을 이용해서 분류할 수 있게 하고 있다. 화살표는 DFD에서와 마찬가지로 source의 출력값이 target의 입력으로 사용되는 것을 의미한다.

Function variable node 아래의 표 1은 그림 2의 FOD 중에서 function variable에 해당하는 *f_X_Valid* 노드에 대한 SDT이며, 다음과 같이 해석된다. *"f_X*의 값이 *k_X_MIN*과 *k_X_MAX* 사이에 존재하면 입력값이 올바르다는 의미의 출력값으로서 *f_X_Valid*에 0을 발생한다. 만약, 두 상수값 사이에 존재하지 않으면 1을 내보낸다." NuSCR에서는 관련이 있는 조건문은 하나의 복합문으로서 한 줄에 표기하도록 하고 있다.

Timed-history variable node 아래의 그림 3은 위의 FOD에서 timed-history variable node인 *th_X_Trip*에

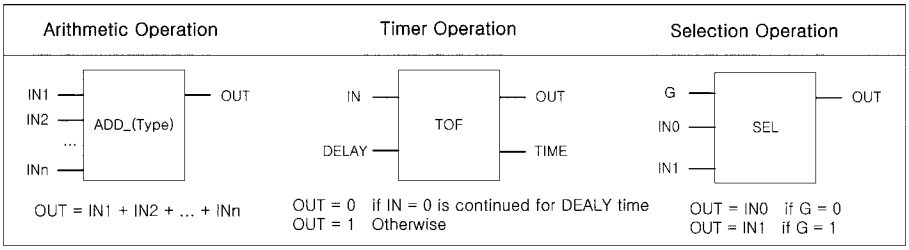


그림 1 원자력 발전소의 디지털 보호 시스템에서 사용되는 IEC 61131-3 FBD의 예

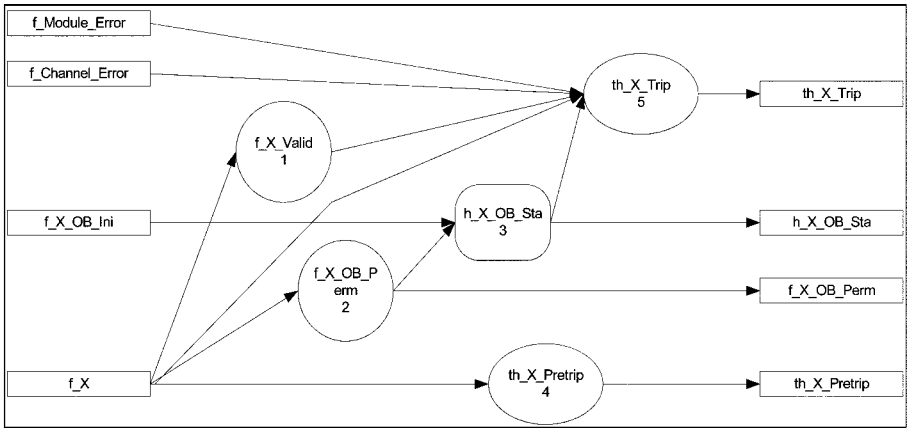


그림 2 *g_Fixed_Setpoint_Rising_Trip_with_OB*에 대한 FOD

표 1 *f_X_Valid*에 대한 SDT

Conditions		
k_X_MIN <= f_X <= k_X_MAX	T	F
Actions		
f_X_Valid := 0	X	
f_X_Valid := 1		X

대한 TTS이다. TTS는 변수의 상태 흐름 정보와 시간 제약 조건을 모두 표현할 수 있는 오토마타이다. TTS는 history variable node의 정의에서 사용되는 FSM에서 커다란 의미의 변화 없이 원자력 발전소 제어 시스템에서 사용되는 시간 조건인 타이머 기능을 충분히 명세할수 있다는 특징이 있다. 그림 3의 TTS는 다음과 같이 해석된다. *"Normal* 상태에서 조건 *f_X* >=

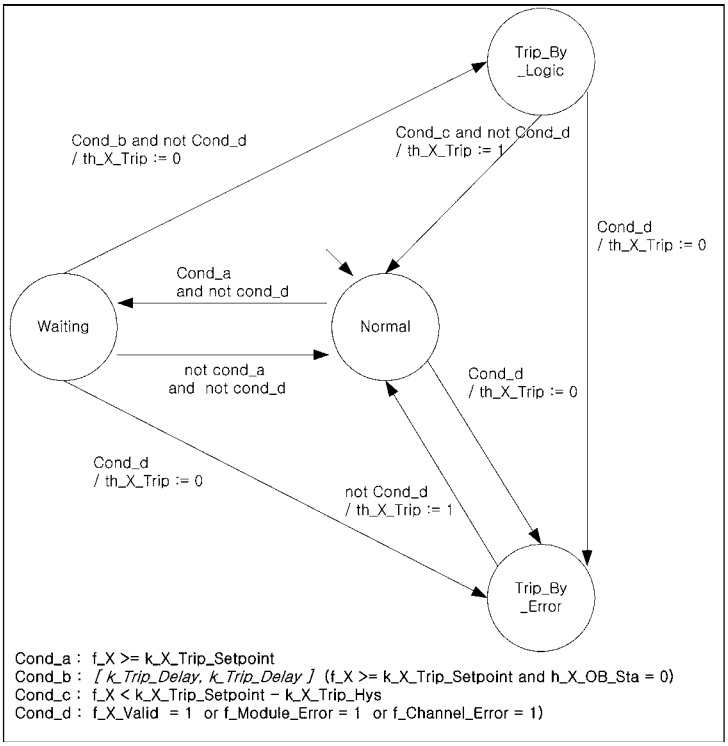


그림 3 *th_X_Trip*을 위한 TTS

*k_X_Trip_Setpoint*를 만족하면 *Waiting* 상태로 전이한 후, 이 조건이 *k_Trip_delay* 시간동안 지속되면 출력으로 트립을 의미하는 0을 발생한다. 만약 *f_X_Valid*, *f_Module_Error*, *f_Channel_Error*와 같은 에러가 발생하면 바로 트립을 발생시킨다. 트립 상태인 *Trip_By_Error*이나 *Trip_By_Logic*에서는 트립 조건이 해제되면, 출력값을 정상을 의미하는 1로 발생시키면서 다시 *Normal* 상태로 돌아온다." *Cond_b*에서 *[k_Trip_Delay, k_Trip_Delay]*가 *k_Trip_Delay* 시간동안 시간이 지속됨을 의미하는 TTS의 표현이다.

History variable node 그림 2 FOD에서 *h_X_OB_Sta* 노드는 history variable node이며, FSM으로 정의된다. FSM은 시간 조건만을 제외하면 그림 3의 TTS와 동일하다.

NuSCR에서는 테이블 형태의 SDT 외에 FSM과 TTS와 같은 오토마타를 제공함으로써 개발자가 정의하려는 대상의 특징에 적합한 변수 모델을 선정하여 명세할 수 있게 하고 있다. 따라서, 기존에 제시되었던 기법들에 비해 명세하기가 수월하며 이해 및 해석도 용이한 것으로평가 받고 있다[9].

4. NuSCR로부터 FBD를 생성하는 과정

본 장에서는 정형명세 기법인 NuSCR을 이용해서 작성된 분석 단계의 요구사항 명세로부터, 설계 단계의 내용으로 사용될 수 있는 PLC 기반의 FBD 프로그램을 체계적으로 생성하는 기법에 대해 설명한다. 제안된 기법은 기존의 수동 명세 작업에서 발생하던 오류들을 크게 줄일 수 있으며, 소프트웨어의 개발 비용과 기간을 줄일 수 있다는 장점을 지닌다. 변환 과정을 보다 효과적으로 설명하기 위해서 본 장에서는 현재 KNICS에서 개발중인 DPPS RPS의 BP를 구성하는 트립 논리 중의 하나인 고정 설정치 상승 트립을 예제로 사용하고 있다.

예제 시스템 예제로서 사용되는 시스템은 원자력 발전소의 보호계통(Digital Plant Protection System) RPS(Reactor Protection System)의BP(Bistable Processor)를 구성하는 트립 논리 중의 하나인 고정설정치 상승 트립(Fixed set-point rising trip with operating bypass)이다. 이 트립 논리 부분은 외부로부터 입력 받은 대수 출력값(log power: *f_X*)이 정상적인 값(*f_X_Valid*)일 때, 정해진 트립 설정치 이상으로 증가하면 트립(*th_X_Trip*)을 발생하는 논리이다. 또한, 운전원이 운전우회(operating bypass) 신호(*f_X_OB_Ini*)를 보내면 트립 조건이 만족해도 트립을 발생시키지 않는 논리(*h_X_OB_Sta)*도 포함되어 있으며, 운전우회는 입력 받은 대수 출력값이 일정한 범위(*f_X_OB_Perm*) 내에 있을 때에만 가능하다. 또한 트립 신호 외에 미리 예비로 출력되는 예비트립(*th_X_Pretrip*)도 포함되어 있다. 예비 트립은 트립 보다 트립 설정치가 일정수준 낮게 설정되어 있어서 실제 트립에 발생하기에 앞서 운전원에게 알려주는 역할을 한다. 이 트립 논리에 대한 NuSCR FOD는 앞 장의 그림 2이며, NuSCR의 세 가지 변수 모델에 해당하는 *f_X_Valid*, *th_X_Trip*에 대한 정의는 각각 표 1과 그림 3이 해당된다.

FBD 생성 과정 NuSCR 요구사항 명세로부터 FBD 프로그램을 생성하는 전체 과정은 다음의 그림 4와 같다. FBD 생성 과정은 네 단계로 구분된다. 첫 단계에서는 SDT, FSM 및 TTS로 정의되어 있는 모든 단일 노드들에 대해서 consistency와 completeness를 분석을 수행한다. 이러한 분석 과정을 통해서 모든 노드들을 complete하고 consistent하게 수정한 후에는 FSM과 TTS로 정의된 노드들에 대해서 2C-Table을 작성한다. 2C-Table은 FSM이나 TTS로부터 FBD를 생성하는데 사용되는 중간단계의 테이블이다. 다음으로는 이와 같이 생성된 2C-Table이나 원래의 SDT를 이용해서 FBD를 생성한다. 이 단계에서는 FOD 상에서SDT, FSM, TTS로 정의된모든 노드들에 대한 FBD를 독립적으로 생성하게 된다. 마지막 단계에서는 FOD에 표현된 각 노드들간의 의존관계를 분석하여 생성된 FBD들에게 일정한 실행 순서(execution order)를 부여한다. PLC는 순차적으로 수행되므로 이와 같이 각각의 생성된 FBD들에게 FOD를 분석한 결과에 따라 실행 순서를 지정해주는 작업이 필요하다. FBD 생성 과정에 대한 정형 정의와 생성 알고리즘에 대한 설명은 [16]에 자세히 설명되어 있다.

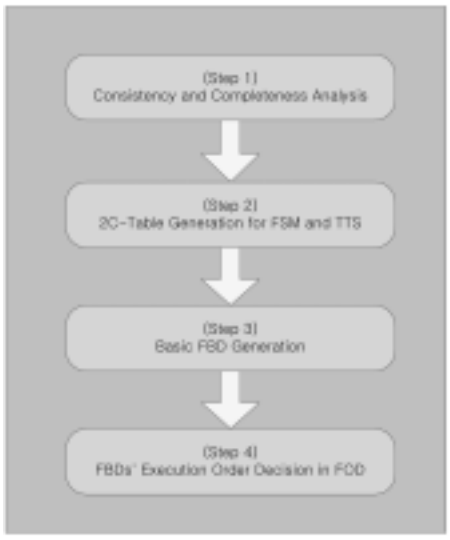


그림 4 FBD 생성의 전체 단계

Step 1. 완전성 및 일관성 분석 SDT, FSM, TTS로 정의된 모든 노드들은 효과적인 즉, 적은 개수의 FBs를 가지는 FBD의 생성을 위해서 먼저 완전(complete)하고 일관(consistent)[7]되게 수정해주는 작업이 선행된다. SDT는 Condition문과 Action문으로 구성된 테이블이므로 정의된 조건들이 가능한 모든 경우를 표현하고 있으며, 중복되는 조건이 없는가를 조사해야 한다. 또한, 하나 이상의 서로 다른 조건이 동시에 만족됨으로써 상이한 출력값이 동시에 발생하는 경우에 대해서도 분석해 보아야 한다. 아래의 그림 5는 SDT를 위한 예로서, SDT에 대해서 완전성과 일관성을 보정하는 예를 보이고 있다. (a)에서는 *Cond_a = 1* 이고 *Cond_b = 1*인 경우에는 *Output* 값으로서 0 또는 1이 모두 발생할 수 있으므로 일관적이지 않다. 또한 *Cond_a ≠ 1* 이고 *Cond_b = 1* 인 경우에 대해서는 고려하고 있지 않으므로 완전하지 않다. 이러한 SDT를 완전하고 일관되게 보정한 결과가 (b)의 SDT이다.

그림 6은 그림 3에서 제시된 timed-history variable node인 *th_X_Trip*을 정의하고 있는 TTS를 완전하고 일관되게 수정한 결과이다. History variable node를 정의할 때 사용되는 FSM도 동일한 방법으로 수정될 수 있다. 오토마타는 현재 상태에서 다른 상태로 전이하는

Conditions			
Cond_a = 1	T	T	F
Cond_b = 1	T	-	F
Actions			
Output := 0	X		
Output := 1		X	X

(a) Original SDT

Conditions				
Cond_a = 1	T	T	F	F
Cond_b = 1	T	F	T	F
Actions				
Output := 0	X			
Output := 1		X	X	X

(b) Complete and consistent SDT

그림 5 *f_X_Valid*를 위한 SDT의 완전성과 일관성에 관한 예

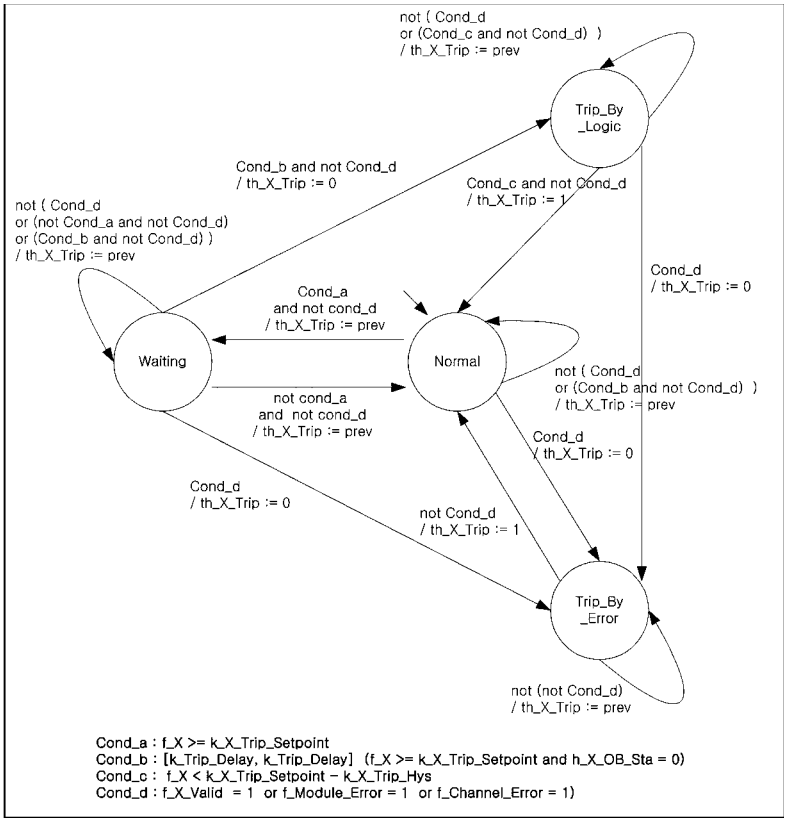


그림 6 수정된 완전하고 일관된 *th_X_Trip*에 대한 TTS

조건이 만족되지 않으면 현재 상태를 계속 유지하게 된다. 따라서, 이러한 경우에는 완전성을 위해서 적절한 action을 추가해 주는 작업이 필요하게 된다. NuSCR 방법론의 FSM과 TTS에서는 Action 부분이 명시되지 않을 경우에는 출력값으로서 전 상태의 값(*prev*)를 출력하는 것으로 가정하므로, 이러한 부분에 대한 추가적인 작업도 요구된다. 또한, 현재 상태에서 다른 상태들로 전이하는 여러 조건들이 중복됨으로써 두 개 이상의 상태 전이가 동시에 발생하지 않도록 하는 검사 및 수정하는 작업도 수행되어야 한다.

Step 2. FSM와 TTS에 대한 2C-Table 생성 FSM과 TTS로 정의된 모든 노드들은 Step 1의 작업을 수행한 후에 2C-Table 형태로 변환해 준다. 2C-Table은 FSM 및 TTS와 동일한 행위를 표현한 테이블로서 SDT와 유사하다. SDT와의 차이점은 테이블의 action 부분에 output을 정하는 부분 외에 오토마타의 상태 전이를 지정하는 부분이 추가되어 있다는 점이다. 이와 같이 추가적으로 생성된 2C-Table은 오토마타로 표현된 노드들로부터 FBD를 기계적으로 생성하기 위한 중간단계의 역할을 한다. 그림 6의 수정된 TTS를 바탕으로 하여 완성된2C-Table은 다음의 표 2와 같다. 변수 *SV*는 오토마타의 상태들을 의미하는 상태변수이며, *Output*은 노드 *th_X_Trip*의 출력값을 의미한다.

생성된 2-C Table은 완전하고 일관되게 수정된 오토마타 즉, FSM과 TTS로부터 생성되었으므로 Condition 문들이 서로 배타적인(exclusive) 특성을 지닌다. 이러한 2-C Table의 특성으로 인해서 이를 기반으로 생성된 FBD는 적은 수의 FBs 만으로도 정의될 수 있게 된다. 또한, 2-C Table은 생성될 FBD에서 사용될 FB들의 종류와 개수를 미리 예측 및 분석할 수 있게 한다. SDT는 이미 이러한 테이블의 형태이므로 동일한 작업을 수행할 수 있다.

Step 3. 기본 FBD 생성 다음 단계에서는 수정된 SDT나 추가적으로 생성된 2C-Table을 기반으로 FBD를 생성한다. 다음의 그림 7은 SDT와 이에 대한 분석을 바탕으로 생성된 FBD이다. 생성된 FBD는 먼저 SDT의 Condition들만을 분류해서 미리계산해 놓는 preprocessing part와 구체적인 계산을 수행하는 output processing part로 구분되어서 작성된다. 이 FBD는 Schneider Automation GmbH에서 제공하는 PLC 프로그래밍 도구인 Concept version 2.2 XL SR2를 이용해서 작성된 것으로 FB 위에 있는 숫자는 해당하는 FB의 생성 순서와 실행 순서를 의미한다.

다음의 그림 8은 표 2의 2-C table을 바탕으로 생성된 *th_X_Trip* 노드에 대한 FBD이다. 생성된 FBD는 먼저 SDT의 Condition들만을 분류해서 미리 계산해 놓는 preprocessing part (a)와 구체적인 계산을 수행하는 output processing part (b) 및 오토마타의상태를 계산하기 위한 state-variable processing part (c)로 구분되어 생성된다. 그림 8(a)는 사용되는 전체 Conditions

표 2 *th_X_Trip*를 정의하고 있는 TTS에 대한 2C-Table

SV = Normal	T	T	T									
Cond_a and not Cond_d	T	-	-									
cond_d	-	T	-									
Otherwise	-	-	T									
SV = Waiting				T	T	T	T					
not Cond_a and not Cond_d				T	-	-	-					
Cond_d				-	T	-	-					
Cond_b and not Cond_d				-	-	T	-					
Otherwise				-	-	-	T					
SV = Trip_By_Logic								T	T	T		
Cond_c and not Cond_d								T	-	-		
Cond_d								-	T	-		
Otherwise								-	-	T		
SV= Trip_By_Error											T	T
not Cond_d											T	-
Otherwise											-	T
Output := 0		X			X	X			X			
Output := 1								X			X	
Output := prev	X		X	X			X			X		X
SV:= Normal(0)			X	X				X			X	
SV:= Waiting(1)	X						X					
SV:= Trip_By_Logic(2)						X				X		
SV:= Trip_By_Error(3)		X			X				X			X

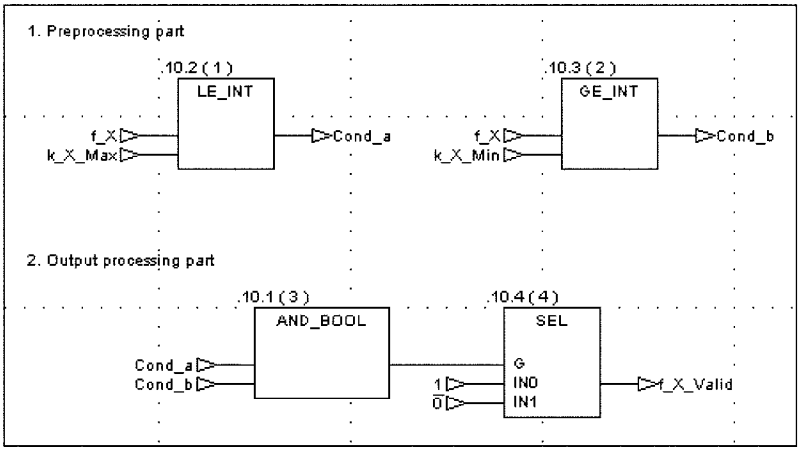
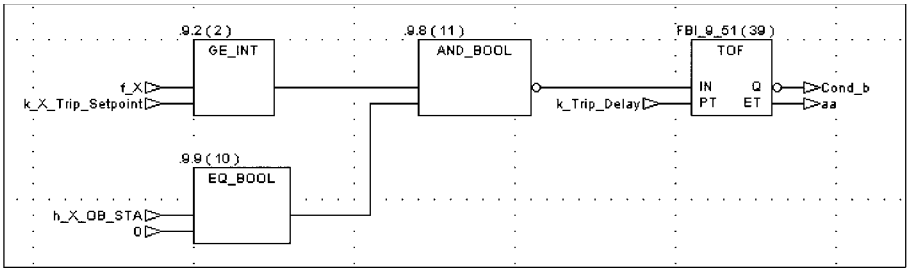
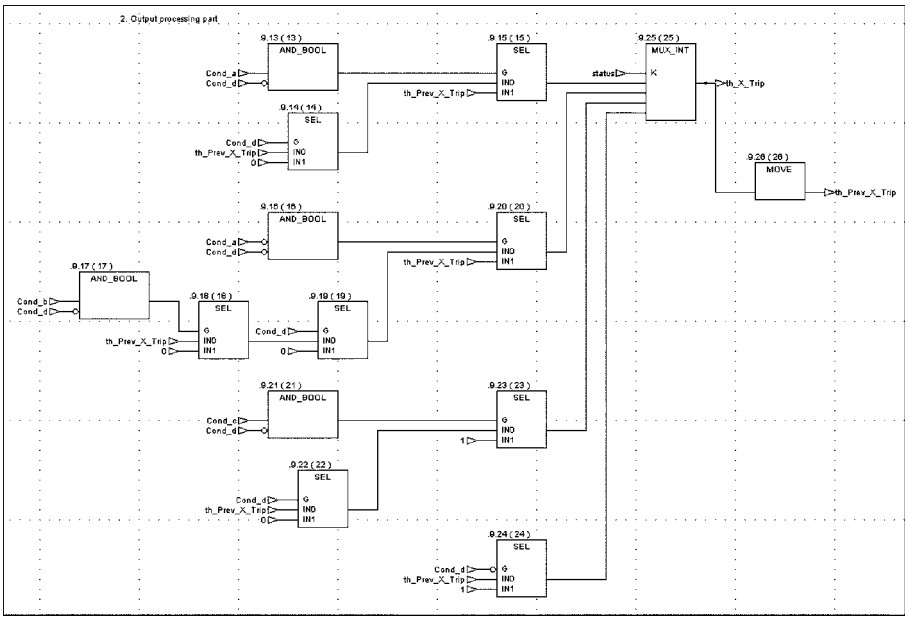


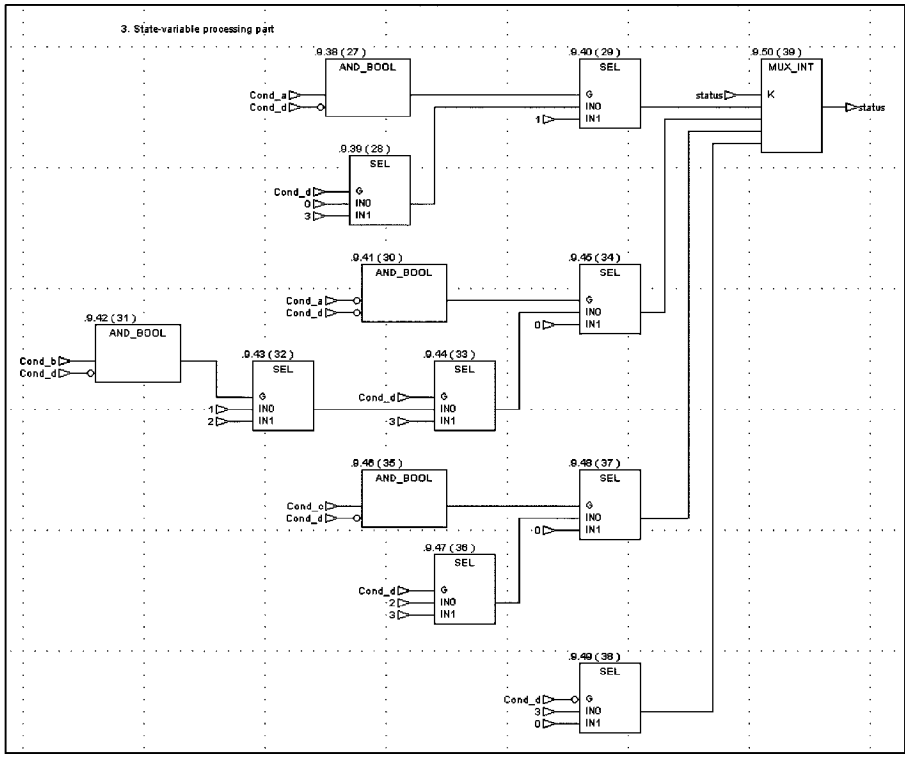
그림 7 *f_X_Valid*에 대한 SDT로부터 생성된 FBD



(a) Processing part FBD



(b) Output processing part FBD



(c) State-variable processing part FBD

그림 8 *th_X_Trip*에 대한 TTS로부터 생성된 FBD

중에서 *Cond_b*에 대한 FBD이다. *Cond_b*는 *[k_Trip_Delay, k_Trip_Delay](f_X >= k_Trip_Setpoint and h_X_OB_Sta = 0)*을 의미하는 매크로이다. *Cond_b*는 *k_Trip_Delay* 시간동안 조건 *(f_X >= k_Trip_Setpoint and h_X_OB_Sta = 0)*이 지속적으로 만족되면 *Cond_b*가 참이 됨을 정의하고 있다.

그림 8(b)는 output processing part에 대한 FBD이다. SDT와 2C-Table에 정의된 각각의 Condition/Action 들은 SEL FB에 의해서 선택되어 진다. 각각의 Condition들은 FBD의 간결한 의미 전달 및 사용되는 FB의 개수를 줄이기 위해서 그림 8(a)와 같이 미리 정의되어서 그 결과값만을 사용한다. SEL FB에 의해서 선택된 출력값은 현재 오토마타가 어떤 상태에 있는가에 따라서 선별되어서 출력되는 작업이 수행되는데 이러한 역할은 MUX FB와 MUX의 입력으로 사용되는 *status* 변수에 의해서 수행된다. MUX FB는 *status* 값에 따라서 해당하는 입력을 출력으로 내보내며, *status* 변수는 2C-Table에 따라 그림 8(c)와 같이 생성된 FBD에 의해서 정의된다. 출력값인 *th_X_Trip* 외에 별도로 출력되는 *th_Prev_X_Trip* 변수는 내부적으로만 사용되는 내부 변수이다.

이와 같은 과정이 완료되면, FOD 상에 존재하는 모든 function variable node 및 history variable node와 timed history variable node에 대한 개별적인 FBD가 완성된다.

Step 4. FBD의 실행순서 결정 마지막 단계에서는 각각의 FBD에 실행 순서를 지정해 주는 작업이 진행된다. 그림 2와 같이 5개의 노드로 구성된 FOD를 대상으로, 가능한 모든 실행 순서를 고려해 보면, 먼저 다음과 같은 두 종류의 부분적인 순서 관계가 있음을 알 수 있다. 또한 4번 노드는 다른 노드들과 연관 관계가 없이 독립적으로 수행할 수 있음을 알 수 있다.

Partial execution order 1: (1→5)

Partial execution order 2: (2→3→5)

이러한 부분적인 순서 관계와 독립적인 관계를 고려했을 때, 전체 FOD에 대한 가능한 실행 순서는 다음과 같다. 노드 4번은 독립적이므로 어느 자리에 위치해도 무관하다.

Execution Order 1: (입력)→1→2→3→5→(4)→(출력)

Execution Order 2: (입력)→2→1→3→5→(4)→(출력)

표 3 사용된 FB의 개수에 대한 비교

	f_X_Valid	th_X_Trip	th_X_Pretrip	f_X_OB_Perm	h_X_OB_Sta	Total
System atically generated from NuSCR	3	39	16	2	11	71
Manually generated by experts	3	12	8	1	3	27

Execution Order 3: (입력)→2→3→1→5→(4)→(출력)

생성된 FBD에 대한 평가 다음의 표 3은 KNICS RPS BP의 트립 논리 중에 하나인 고정설정치 상승트립 논리에 대해서 실제 엔지니어가 작성한 FBD와 본 논문에서 제안한 기법을 이용해서 생성한 FBD를 비교한 결과이다. 정형명세 기법인 NuSCR을 이용해서 작성된 요구사항 명세로부터 제안된 과정을 통해 체계적으로 생성된 FBD는 NuSCR로 명세를 기반으로 개발자가 수동으로 작성한 FBD와 비교해 볼 때, 사용된 FB의 개수의 비가 #(Systematic generated FBs) : #(FBs generated by experts) = 3 : 1의 관계가 성립함을 실험적으로 확인할 수 있다. 이는 전문가에 의해 수동으로 작성된 FBD는 전문가의 최적화(optimization) 작업이 함께 수반되기 때문이다.

PLC 소프트웨어의 동작 주기는 30 - 50 ms인 반면에 소프트웨어의 실제 실행 시간은 5 - 10 ms 정도가 보통이다. 따라서, 3배 정도의 FBs 수의 증가는 소프트웨어의 수행 시간을 크게 증가시키지 않으며, 주기적으로 동작하는데 영향을 끼치지 않는다. 또한, FBD 프로그램을 제안된 기법을 통해서 기계적으로 생성할 수 있으므로 기존의 수동으로 하던 명세 작업에서 발생하던 오류들 즉, 자연어나 정형문서로 작성된 요구사항 명세를 읽고 이해하고, FBD를 직접 작성하는 과정에서 발생할 수 있는 문제들을 크게 줄일 수 있으며, 소프트웨어의 개발 비용과 기간 또한 크게 줄일 수 있을 것으로 예상된다.

5. 결론 및 향후 연구

원자력 발전소의 제어 소프트웨어는 SFC, LD, FBD 등의 PLC 프로그래밍 언어를 기반으로 하는 설계 단계가 완성되면, 실질적인 소프트웨어의 개발이 완료되었다고 간주 할 수 있다. 이와 같이 중요한 역할을 하는 설계 단계는 미리 작성된 요구사항 문서를 바탕으로 개발자가 수동으로 작성하는 방법이 주로 사용되는데, 이는 전 단계의 요구사항 문서가 자연어로 기술되어 있기 때문이다.

본 논문에서는 정형명세 기법인 NuSCR을 이용해서 작성된 정형 요구사항 명세로부터, PLC 프로그래밍 언어 중에서 가장 널리 사용되는 FBD 프로그램을 체계적으로 생성할 수 있는 기법을 제시하고 있다. 제안된 생성 기법을 이용하면, NuSCR 명세를 기준으로 이와 동일한 행위를 하는 PLC 프로그램을 체계적으로 생성할 수 있으며, 기존의 수동으로 하던 명세 작업에서 발생하던 오류들도 크게 줄일 수 있다[7,9]. 또한, 소프트웨어의 개발 비용과 기간을 크게 줄일 수 있을 것으로 예상된다. 본 논문에서는 NuSCR 정형 명세로부터 FBD를 체계적으로 생성하는 기법을 현재 KNICS에서 개발 중인 DPPS RPS BP 모듈을 구성하는 트립 논리중의 하나인 고정 설정치 상승 트립 부분에 적용한 결과를 실례로서 제시하고 있다.

향후 연구로는 먼저, NuSCR을 구성하는 요소들인 SDT, FSM, TTS, FOD에 대해서 완전성과 일관성을 자동으로 검사 및 수정해 주는 연구가 있으며, 이 연구는 현재 진행 중에 있다. 또한, 생성된 FBD에 대해 NuSCR로 작성된 요구 명세를 기준으로 하여 fault-tree analysis나 backward reachability analysis와 같은 안전성 분석을 수행하는 연구도 고려할 수 있다. 특히, 제안된 기법을 이용하면, 설계 단계의 FBD 프로그램은 분석 단계의 NuSCR 정형명세로부터 생성된 것이므로, 두 단계 사이의 일관성 있는 안전성 분석[17]을 수행할 수 있을 것이다. 즉, 설계 단계에서 FBD를 기준으로 작성된 fault-tree의 구성 요소들이 상위 단계인 분석 단계의 NuSCR 명세를 기준으로 작성된 fault-tree에서도 같은 의미로 인식될 수 있다면, 분석 단계와 설계 단계 간에 추적 가능한 일관된 안전성 분석이 가능하게 된다. 또한, FBD를 대상으로 기존의 coverage criteria를 응용하여 직접 testing을 수행하는 연구도 진행 중이며, NuSCR 명세를 기준으로 명세 기반의 테스팅을 수행했을 때의 결과를 FBD에 직접 테스팅 할 때의 결과와 비교해 봄으로써 두 개발 단계 간의 차이점을 확인해 보는 연구도 추가로 수행할 계획이다.

참 고 문 헌

[1] Nancy G. Leveson, SAFEWARE: System safety and Computers, Addison Wesley, 1995.
[2] U.S. NRC, Digital Instrumentation and Control Systems in Nuclear Power Plants: safety and reliability issues, National Academy Press, 1997.
[3] Doron A. Peled, SOFTWARE RELIABILITY METHODS, Springer, 2001.
[4] UK MoD, The procurement of safety critical software in defense equipment, Define Standard 00-55, 1997.
[5] D. Parnas, A. J. Schouwen Van, and J. Maday, "Documentation of requirements for computer

systems," In Proc. RE'93: IEEE International Symposium on Requirements Engineering, pp.198-207, 1993.
[6] Wolsong NPP 2/3/4, Software requirements specificationfor shutdown system 2 PDC, 86-68350-SRS-001, June, 1993.
[7] Junbeom Yoo, Taihyo Kim, Sungdeok Cha, Jangsu Lee, Han Sung Son, "A Formal Software Requirements Specification Method for Digital Nuclear Plants Protection Systems," Journal of Systems and Software, Vol. 74, No. 1, pp.73-83, 2005.
[8] KNICS, Korea nuclear instrumentation and control system research and development center, http://www.knics.re.kr.
[9] Junbeom Yoo, Sungdeok Cha, Changhui Kim, Younju Oh, "Formal Requirements specification for digital reactor protection systems," Journal of KISS, Vol.31, No.6, pp.750-759, 6, 2004.
[10] IEC, International standard for programmable controllers: Programming languages, Technical Report IEC 1131 part 3, IEC (International Electrotechnical Commission), 1993.
[11] Henning Dierks, "PLC-Automata: A new class of implementable real-time automata," Theoretical Computer Science, 1997.
[12] Angelika Mader, "A Classification of PLC Models and Applications," In Proc. WODES 2000: 5th Workshop on Discrete Event Systems, 2000.
[13] K. L. Heninger, "Specifyingsoftware requirements for complex systems: New techniques and their application," IEEE Trans. Software Engineering, SE-6(1), pp.2-13, 1980.
[14] D. Parnas and J. Madey, "Functional documentation for computer systems engineering," CRL 237, Telecommunications Research Institute of Ontario(TRIO), McMaster Univ., Hamilton, Ontario, 1991.
[15] Zphar Manna, Thomas A. Hensinger, and Amir Pnueli, "Timed transition systems," In Proc. REX Workshop, pp.226-251, 1991.
[16] Junbeom Yoo, Hojung Bang, Sungdeok Cha, "Procedural Transformation from Formal Software Requirement to PLC-based Design," KAIST CS/TR-2004-198.
[17] Y. Papadopoulos, J. McDermid, R. Sasse, and G. Heiner, "Analysis and synthesis of the behavior of complex programmable electronic systems in conditions of failure," Reliability Engineering and System Safety, Vol. 71, No. 3, pp.229-247, 2001.

유 준 범
1999년 홍익대학교 컴퓨터공학과 학사 2001년 KAIST 전자전산학과 전산학전공 석사. 2001년~현재 KAIST 전자전산학과 전산학전공 박사과정. 관심분야는 소프트웨어공학, 안전성분석, 정형검증 및 명세

차 성 덕
1983년 University of California, Irvine 전산학 학사. 1986년 University of California, Irvine 전산학 석사. 1991년 University of California, Irvine 전산학 박사. 1994년~현재 KAIST 전자전산학과 전산학전공 부교수. 관심분야는 정형기법 및 명세, 정보보호, 침입탐지

김 창 회
1995년 충남대학교 전자공학과 박사. 현재 원자력연구소 계측제어인간공학연구부 책임연구원. 연구세부분야: 원자로보호계통 개발

송 덕 용
2000년 전주대학교 컴퓨터공학부 학사 현 ㈜액트 엔지니어링 사업팀 과장. 연구세부분야: 디지털온라인자동시험기 개발