2002 추계학술발표회 논문집
한국원자력학회

원전계측제어시스템 안전소프트웨어 검증을 위한 모델체커 VIS의 활용성 고찰

A Usability Review of a Model Checker VIS for the Verification of NPP I&C System Safety Software

손한성, 권기춘

한국원자력연구소 MMIS팀
대전광역시 유성구 덕진동 150

요 약

본 논문은 원전계측제어시스템 안전소프트웨어가 개발되는 환경의 예시를 통해 모델체커 VIS가 소프트웨어 개발환경 상에서 어떻게 검증에 활용될 수 있을 것인가에 대하여 고찰해 본다. 요구사항분석과 설계에 STATEMATE 환경이 적용되고, 실시간 운영체제나 통신과 같이 하드웨어를 직접 구동하는 기능을 구현하거나 PLC 프로그램과 같은 내장형 소프트웨어의 특성을 갖는 경우에는 VIS의 활용성이 매우 높은 것으로 판단되었다. 향후 코드의 추상화 및 VIS 입력으로의 자동 변환에 대한 연구가 진행된다면, VIS의 활용성은 더욱 증대될 것으로 기대된다.

Abstract

This paper discusses the usability of a model checker VIS in the verification of safety software of NPP I&C systems. The software development environment exemplified in this paper is for PLC and ESF-CCS which are being developed in KNICS project. In this environment, STATEMATE is used in requirement analysis and design phases. PLC is expected to be implemented using C language and an assembly language because it has many interfaces with hardware like CPU, I/O devices, communication devices. ESF-CCS is supposed to be developed in terms of PLC programming languages which are defined in IEC 61131-3 standard. In this case, VIS proved to be very useful through the review. We are also able to expect greater usability of VIS if we further develop the techniques for code abstraction and automatic translation from code to verilog, which is the input of VIS.

1. 서론

원전계측제어시스템은 전형적인 내장형(Embedded) 실시간(Real-time) 시스템으로서 소프트웨어가 시스템 구동의 중심 역할을 담당하고 있다. 일반적으로 컴퓨터 시스템을 제어하는 소프트웨어는 그 특성상 시간적 보장이 비결정적(Non-deterministic)이며, 또한 여러 개의 프로세스를 운용하는 멀티 프로세스(Multi-process) 시스템인 경우 그 비결정성(Non-determinism)으로 인해 데드록(Deadlock) 등의 결정적인 오류를 가질 수 있다. 따라서 원전계측제어시스템의 경우, 내장되는 소프트웨어에 대하여 다양한 측면으로 다양한 기법을 이용하여 확인 및 검증(Verification and Validation)을 수행할 필요가 있다. 특히 규제기준에서 채택하고 있는 IEEE Std. 7-4.3.2[1]이나 IEEE Std 1012[2]나 IEEE Std 1074[3] 등과 같은 산업표준에서는 소프트웨어 개발 주기의 각 단계 별로 확인 및 검증을 수행할 것과 또한 안전에 중요한 정도에 따른 철저한 확인 및 검증 수행을 권장하고 있다.

원전계측제어시스템개발사업단에서는 원자로보호계통과 같은 안전에 매우 중요한 시스템을 개발하면서 소프트웨어에 대한 철저한 검증을 수행하는 것을 지향하고 있다. 캐나다 AECL이 일정한 수준의 엄격함으로 검증을 수행한 것이 가장 높이 평가되고 있는 실정임을 감안하면 원전계측제어 소프트웨어 검증 분야에서는 세계적인 수준의 기술력을 확보할 수 있을 것으로 기대된다. 소프트웨어에 대하여 철저한 검증을 수행하기 위해서는 요구사항분석, 설계, 구현 등의 활동을 체계적으로 수행하는 것이 전제되어야 한다. 검증 가능한 소프트웨어를 생산할 때 철저한 검증이 이루어질 수 있기 때문이다. 본 연구진은 수학적 명세 언어로 설계하여 수학적인 증명 방법으로 검증을 수행하는 정형기법을 이용하여 소프트웨어에 대한 철저한 검증을 수행하고 이를 통하여 소프트웨어의 안전성 및 신뢰성에 대한 확신을 높이려 하고 있다.

정형기법은 정형명세기법과 정형검증기법으로 분류할 수 있으며, 정형검증기법은 모델체킹과 정리증명의 두 가지 기법이 있다. 모델체킹은 유한 상태 시스템에 대한 분석방법으로서, 시스템이 만족해야 할 속성을 시제 논리 공식으로 표현하고, 시스템의 실제 행위를 병렬 상태-전이 그래프로 나타내어 모델체커의 입력으로 전달한다. 모델체커는

시스템이 그 속성을 만족하는지 여부를 자동적으로 검증해 준다. 만일 속성이 만족되지 않는 경우는, 반례를 생성하는데 이 정보가 소프트웨어에 대한 철저한 검증에 매우 유용하게 쓰일 수 있다. 모델체커를 통한 검증을 통해 우리는 소프트웨어가 만족해야 할 다양한 특성에 대한 만족여부를 분석할 수 있고, 소프트웨어에서 잘못된 부분을 찾아낼 수도 있으며, 검증 결과를 분석함으로써 잘못된 부분을 정확하게 보정하는 정보들을 얻을 수도 있다.

본 논문에서는 모델체커 VIS가 원전계측제어시스템 소프트웨어 검증에 어떻게 활용될 수 있을 것인가에 대하여 고찰한다. 이러한 고찰은 원전계측제어시스템 소프트웨어 검증 절차 개발에 활용하기 위한 것이며 모델체킹 외의 다른 검증 기법들에 대한 활용 범위 및 방안을 도출하기 위한 것이기도 하다.

2. 모델체커 VIS의 소개

VIS(Verification Interacting with Synthesis)는 검증, 시뮬레이션, 유한 상태 하드에어 시스템의 합성을 종합적으로 수행하는 도구이다 [4]. 이 도구는 입력 언어를 Verilog를 이용하고 fair CTL 모델체킹, 조합 및 순차 동등 검사(Combinational and sequential equivalence checking) 등의 검증 기능을 제공하며, 계층적 합성(Hierarchical synthesis) 기능도 지원한다. 그림 1이 VIS의 기능을 다이어그램으로 보여주고 있다.

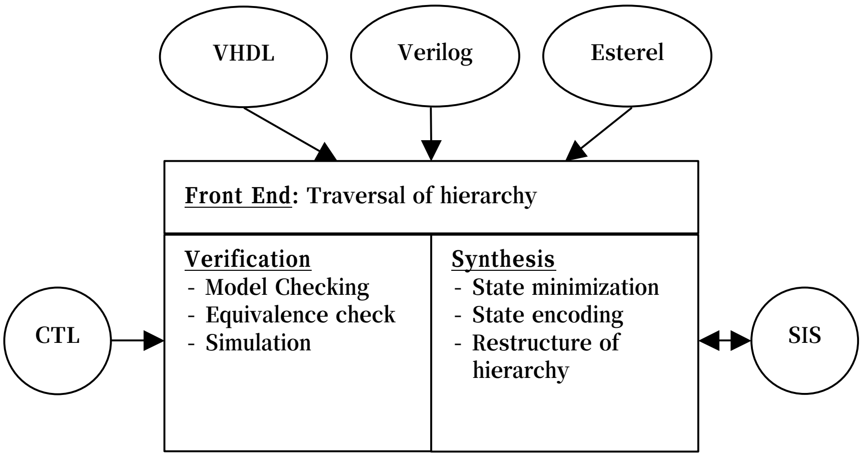

그림 1. 모델체커 VIS의 기능

VIS는 BLIF-MV라는 중간 형태에서 작동되며 BLIF_MV 파일은 vl2mv 라는 컴파일러에 의해 생성된다. VIS가 사용하는 입력언어는 Verilog로서 일반적인 Verilog와 다른 점은 비결정성과 기호적 변수를 사용한다는 것이다. vl2mv가 이러한 Verilog 언어를 BLIF-MV

형태의 상태기계로 변환해 주면, 검증자가 검증하고 싶은 특성을 CTL이라는 논리언어 형태로 입력하여 모델체킹을 수행한다.

CTL은 상태전이 그래프로부터 유도되며, 상태전이 그래프는 초기 상태로부터 무한한 경우를 나타내는 트리로 표현된다. 이 트리에서 나타나는 모든 경로(path)는 모델링 되는 모든 가능한 계산을 나타낸다. CTL 표현식은 단위 명제와 명제 논리의 부울 합성자(Boolean connective), 시제 연산자로 구성된다. 각 시제 연산자는 두 부분으로 구성되어지는데, 경로 한정자(Path quantifier: A,E)와 시제 구성자(Temporal modality: F,X,G,U)이다. 각 공식(Formula)이 q라는 상태에서 적용된다고 가정하며 경로 한정자 중 A는 "q"에서 시작하는 모든 경로를 의미한다. 한편, E는 "q"에서 시작하는 어떤 한 경로를 나타낸다. 시제 구성자 F는 "해당 경로의 어떤 한 상태"를, X는 "다음에", G는 "해당 경로의 모든 상태"를 의미하며, U는 "~일 때까지"를 나타낸다. 그래서 AG safe가 q에서 만족되려면, q에서 시작하는 모든 경로(A)의 모든 상태(G)들이 safe라는 공식을 만족해야 한다. 또한 AF safe 가 q에서 만족되려면, q에서 시작하는 모든 경로(A)에 대해, 각 경로에서 적어도 하나의 상태(F)가 safe 를 만족해야 하며, EG safe가 q에서 만족되려면, q에서 시작하는 경로 중 적어도 하나의 경로(E)의 모든 상태(G)가 safe를 만족해야 한다. 한편 EF safe가 q에서 만족되려면, q에서 시작하는 경로 중 적어도 하나의 경로(E)에서 적어도 하나의 상태(F)가 safe를 만족해야 한다.

그러나 시스템이 여러 사용자 사이에 공유된 자원을 할당한다면 사용자가 보유하지 않은 자원 사이에 있는 경로들도 고려되어야 한다. CTL로만은 Fair 경로 사이(고려되어야만 하는 path)의 정확성에 관해 나타낼 수 없다. Fair 경로는 무한하게 자주 만족되는 각각의 fair 조건 사이에 있는 경로이다. 이에 대한 문제 해결을 위해서 fairness를 다루는 CTL의 변형인 Fair CTL이 제안되었으며, VIS는 fair CTL을 지원한다.

3. 원전계측제어사업단의 소프트웨어 개발 방향

본 논문에서는 원전계측제어사업단에서 개발하고 있는 논리제어기(Programmable Logic Controller, 이하 PLC) 소프트웨어와 공학적안전설비작동계통(Engineered Safety Feature Actuation System) 소프트웨어를 예로 들어 소프트웨어 개발 방향을 소개하고 이에 따른 VIS의 활용성을 고찰한다. 논리제어기는 원자로보호계통 및 공학적안전설비작동계통의 핵심 구성요소로서 내장형 실시간 시스템의 전형적인 예가 될 수 있다. PLC를 제어하는 가장 중요한 소프트웨어는 실시간 운영체제라 할 수 있다. 이러한 실시간 운영체제는 잘 정의된 시간적 제약을 가져야 한다. 프로세스는 정의된 시간 내에 수행이 완료되어야 하며, 그렇지 못할 경우 예상하지 못한 결과가 발생 할 수 있다. 실시간 운영체제는 주기적으로 발생하는 작업과 비정기적으로 발생하는 작업을 순서에 맞게 실행시켜 주어 모든 작업이 주어진 시간 안에 성공적으로 수행할 수 있도록 해준다. 이러한 실시간 운영체제가 갖는 여러 가지 특성, 즉 빠른 문맥교환, 우선순위에 근거한 스케줄링, 클럭의 실시간성, 프로세스의 강제 배제

특성 등이 기본적인 검증 및 시험 대상이라고 할 수 있다.
한편, 공학적안전설비작동계통은 원자력발전소의 상태에 따라 적절하게 반응하도록 설계되는 전형적인 응답형 시스템(Reactive System)이다. 공학적안전설비작동계통은 원전의 안전에 매우 중요한 기능을 수행하므로 무엇보다도 소프트웨어의 안전성이 가장 중요한 검증 및 시험의 목표가 된다.

3.1. 소프트웨어 요구사항분석

원전계측제어사업단에서 개발하고 있는 PLC와 공학적안전설비작동계통에 대한 소프트웨어 요구사항은 Statechart라는 정형명세 언어를 기반으로 명세되고 분석되며, 자동화된 Statechart 기반의 모델링 및 분석을 위하여 STATEMATE[5]라는 소프트웨어공학도구가 활용된다. STATEMATE가 제공하는 Activity Chart와 Statechart의 조합으로 소프트웨어의 기능과 그에 따른 행위를 명세하고 분석하는 일이 소프트웨어 요구사항분석 단계에서 이루어 진다.

3.2. 소프트웨어 설계

STATEMATE는 Activity Chart와 Statechart 외에도 Module Chart도 제공하여 소프트웨어 설계를 지원한다. 소프트웨어 요구사항분석 단계를 통하여 명세된 Activity Chart와 Statechart 모델들을 상세화하고, 이렇게 상세화된 모델을 Module Chart에 대응하여 소프트웨어의 구조와 구체적인 구현 방법을 결정하는 일이 소프트웨어 설계단계에서 이루어 진다.

3.3. 소프트웨어 구현

STATEMATE가 제공하는 모델링 및 분석 기능을 이용하여 요구사항분석과 설계가 진행되는 PLC와 공학적안전설비작동계통은 구현 단계에 와서는 전혀 다른 도구를 사용하여 구현된다. PLC의 경우, 예를 들어 실시간운영체제는 C 언어와 어셈블리 언어와 같은 프로그래밍언어를 이용하여 구현되는 반면, 공학적안전설비작동계통은 이렇게 개발된 PLC 상에서 응용 프로그램을 작성할 수 있도록 하는 Ladder Diagram, Function Block Diagram 등과 같은 PLC 고유의 언어를 이용하여 구현되기 때문이다.

4. VIS의 활용성 고찰

4.1. 요구사항분석에서의 활용성

앞서 언급한대로 요구사항분석은 STATEMATE 환경에서 이루어진다. 주목할 것은 STATEMATE가 최근 Model Checker/Model Certifier를 STATEMATE에 내장하여 사용할 수 있도록 출시하였다는 것이다. 여기서 Model Checker/Model Certifier가 VIS를 모델체킹 엔진으로 사용하고 있다는 점이 중요하다. 즉, STATEMATE 환경에서

소프트웨어의 요구사항분석을 수행하는 경우, 자연스럽게 VIS를 모델체킹에 활용할 수 있게 되는 것이다. 따라서 원전계측제어시스템개발사업단에서 개발 중인 소프트웨어 중 PLC 소프트웨어와 공학적안전설비작동계통 소프트웨어에 대해서는 요구사항분석 단계에서 VIS의 활용성이 매우 높다고 할 수 있다.

STATEMATE의 Model Checker/Model Certifier에 대한 간략한 소개는 다음과 같다. Model Checker/Model Certifier는 STATEMATE로부터 관계되는 모델의 정보를 모두 얻을 수 있다. 이러한 정보는 모델체커 VIS의 입력으로 자동 변환되며 사용자는 검증할 특성을 정의하여 모델체킹을 수행하게 된다. Model Checker가 사용하기 쉬운 사용자 연계를 제공하므로 어렵지 않게 모델체킹을 수행할 수 있다는 큰 장점이 있다. Model Certifier에서는 미리 정의된 특성의 패턴들을 사용하여 검증할 특성을 표현할 수 있도록 지원한다. VIS의 출력으로서 반례가 나올 경우, 이것을 표현하고 분석할 수 있게 해 주는 기능도 있다. 비결정성에 대한 분석과 데드 코드 분석(Dead Code Analysis) 등과 같은 견고성 점검(Robustness Check)에 활용될 수 있으며, 안전성 특성에 대한 검증에도 활용할 수 있다.

4.2. 설계에서의 활용성

PLC 소프트웨어와 공학적안전설비작동계통 소프트웨어의 경우 설계 단계에서도 STATEMATE 환경이 그대로 적용되므로 요구사항분석 단계에서와 똑 같은 이치로 VIS의 활용성은 매우 높다.

앞서 설명한 바대로 VIS는 Verilog를 입력으로 받아들여 CTL 모델체킹을 수행하는 검증 도구이다. Statechart가 VIS와 연동한다는 점에 착안한다면, STATEMATE에서 나온 VIS 입력언어와 Verilog로 설계 명세되어 변환된 VIS 입력언어로 등가관계를 보이는 일이 가능함을 알 수 있다. 이렇게 함으로써 소프트웨어 요구명세와 설계명세 사이의 등가관계를 입증하여 보일 수 있게 된다. 또한, 요구명세에서 산출된 요구사항 역시 CTL 형태로 만들어져 있으므로 VIS가 검증하게 될 요구사항으로 입력될 수 있다. 이러한 특성을 활용한다면 요구사항과 설계 사이의 일관성 검증에 VIS가 활용될 수 있을 것이다.

4.3. PLC 구현에서의 활용성

PLC 구현은 C언어와 어셈블리 언어로 이루어진다. PLC 구현 자체가 실시간 운영체제, 통신 기능, 입출력 제어 기능 등을 구현하는 것이고, 이러한 기능들은 필연적으로 하드웨어와의 연계가 있다. 따라서 하드웨어 설계를 위하여 사용되는 Verilog가 실시간 운영체제, 통신, 입출력 제어 등을 잘 기술할 수 있다는 점이 중요하게 부각된다. Verilog가 PLC 구현을 잘 모델링할 수 있다면 이는 곧 VIS가 PLC 구현 단계에서 검증에 활용될 수 있다는 것을 의미한다. 앞서 언급한대로 VIS는 Verilog Front-end를 가지고 있기 때문이다. 문제는 C언어와 어셈블리 언어로 구현되는 PLC 기능들을 어떻게 Verilog로 변환하는가 하는 것이다. 이는 코드 기반의 모델체킹에 대한 연구와 밀접한 관련이 있는 문제이다.

코드기반 모델체킹 기법은 구현된 코드에 대하여 직접적으로 모델체킹을 수행하는 것을 말한다. 이 때 코드의 경우는 일반적으로 사용되는 변수들이 많이 존재하게 되어 변수가 가질 수 있는 상태, 즉 시스템이 가질 수 있는 상태가 급격히 증가하는 문제가 발생한다. 또한 코드에서는 동적인 데이터 구조를 사용함으로써 무한대의 상태공간을 가지는 경우도 발생할 수 있다. 따라서 코드를 적절히 추상화하는 것이 코드기반 모델체킹의 핵심적인 문제가 된다. VIS는 실제 코드에 가까울 뿐 아니라 마이크로프로세서 기반의 하드웨어와 같이 복잡한 시스템에 대하여 잘 정의할 수 있는 Verilog를 입력모델로 활용할 수 있어 코드 추상화에서 매우 유리한 측면이 있다.

4.4. 공학적안전설비작동계통 구현에서의 활용성

공학적안전설비작동계통의 구현은 PLC 프로그래밍 언어로 이루어진다. 참고로 원자로보호계통의 구현도 같은 상황이므로 구현단계에서의 VIS의 활용성은 공학적안전설비작동계통 뿐만 아니라 원자로보호계통에도 그대로 적용된다. PLC 프로그래밍 언어는 IEC 61131-3 표준에 의하여 규격화되어 있다. IEC 61131-3 표준은 네 가지 PLC 프로그래밍 언어와 구조 언어인 Sequential Function Chart(SFC)에 대한 문법(Syntax)과 의미론(Semantics)을 정의한다. 네 가지 언어는 Ladder Diagram(LD), Function Block Diagram(FBD), Structured Text(ST) 및 Instruction List(IL)를 말한다. LD는 릴레이 래더 논리 다이어그램에 기반을 두고 있으며, 부울 기능을 서술한다. FBD는 특성 및 기능을 그래픽 블록으로 표현한다. ST는 파스칼과 비슷하며, 프로시저, 조건, 루프 선언 등을 제공한다. IL은 어셈블리와 비슷한 언어이다. SFC는 PLC 프로그램 및 기능 블록의 내부 조직을 구조화하기 위하여 정의되었다. SFC는 순차적, 병렬적, 또는 선택적 행위를 표현할 수 있도록 확장한 상태 기계이다.

PLC 프로그램의 검증은 다음의 두 단계로 이루어지는 것이 일반적이다.

- PLC 프로그램의 행위를 메시지 교환에 의하여 동기화되는 상태전이 시스템으로 모델링한다. 이 모델링은 작동의미론(Operational Semantics)을 정의하는 것을 말한다.
- 모델링된 시스템을 안전성 등의 특성을 검증하기 위하여 모델체킹 도구에 구현한다. 이 구현은 작동의미론에 근거를 둔다.

작동의미론은 PLC 실행 사이클을 모델링하는 상태전이 시스템과 각 언어에 대한 행위 규칙 및 프로그램 요소를 표현하는 상태전이 시스템으로 구성된다. 특히, SFC에서는 SFC의 구조가 또 하나의 상태전이 시스템으로 변환된다.

4.4.1. PLC 실행 사이클

하나의 상태전이 시스템이 PLC 실행 사이클을 모델링하며, 그것은 입력 읽기, 계산, 출력 쓰기의 세 가지 단계로 모델링된다. 계산 단계는 여러 가지 부프로그램(Sub-program)들을 구동한다. SFC는 PLC 프로그램을 구조화하는데 사용되며 SFC 부프로그램은 오직 또

하나의 SFC 프로그램에 의해서만 호출된다. ST나 LD와 같은 언어는 PLC 프로그램의 전단과 후단에서 사용될 수 있으며, 전이의 조건 등을 기술하기 위해 사용될 수도 있다.

4.4.2. SFC 모델링

SFC에서는 상태전이 시스템의 작동의미론이 프로그램이 어떻게 변이되어 가는가를 변이 규칙의 관점에서 모델링한다. 프로그램은 몇 가지 상태전이 시스템으로 모델링되는데, 각 스텝 변수에 대한 시스템, 각 전이에 대한 시스템, 각 행동에 대한 시스템 등으로 구성된다. 스텝과 전이에 대한 모델링은 문법에 영향을 많이 받게 된다.

4.4.3. LD 모델링

LD 프로그램은 대개 그래픽 요소로 표현된다. 이 문법에 대하여 정의가 된 후에 각 LD 요소에 대한 수학적 해석이 정의되어야 LD 프로그램의 행위에 대한 정형적 기술이 가능하다. LD의 접점은 부울 변수에 값을 할당하는 것이며, 부울 변수 값은 접점의 변수와 접점 사이의 조합 명령에 의존하게 된다. 행위에 대한 상태전이 시스템을 구축하는 간단한 방법은 프로그램의 내부적 동특성, 즉 상세한 프로그램의 실행 과정을 모델링하는 것이다. 상세한 내부적 상태전이 시스템에서 상태는 모든 프로그램 변수의 값과 제어 변수 값으로 나타난다. 제어변수는 실행 패턴의 현 단계를 저장하는 것이 가능하게 한다.

4.4.4. ST 모델링

ST 언어에 대해서는 이제까지와는 다른 접근 방식이 요구된다. 언어의 정형적 문법이 정의된 후, 각 언어 요소들이 단순한 오토마타로 기술되는 것이다. 요소들은 부울 값을 생성하는 선언을 구성하기 위해 집단으로 결합된다. ST의 절차적 부분에서는 선언들이 일반적으로 단순 종결형 순서를 구축하기 위하여 결합된다. 선언들을 하나의 순서로 결합하기 위해서는 상태전이 시스템을 확장하기 위한 합성 및 축소 규칙들이 사용될 수 있다.

이상에서 알 수 있듯이, PLC 프로그래밍 언어들은 상태전이 시스템으로 표현될 수 있다. 또한 PLC의 작동 원리가 기존 하드웨어 기반 시스템의 작동 원리와 크게 다르지 않다는 점도 특기할 만하다. 따라서 VIS가 PLC 프로그래밍 언어로 구현된 시스템의 소프트웨어에 대하여 모델체킹을 적용할 때 원할하게 활용될 수 있을 것으로 사료된다.

5. 결론

본 논문에서는 원전계측제어시스템 소프트웨어가 개발되고 있는 하나의 방안을 소개하고 거기에 모델체커 VIS가 어떻게 활용될 수 있을 것인가에 대하여 고찰해 보았다. 요구사항분석과 설계에 STATEMATE 환경이 적용되고, 실시간 운영체제나 통신과 같이

하드웨어를 직접 구동하는 기능을 구현하거나 PLC 프로그램과 같은 내장형 소프트웨어의 특성을 갖는 경우에는 VIS의 활용성이 매우 높은 것으로 판단된다. 향후 코드의 추상화 및 VIS 입력으로의 자동 변환에 대한 연구가 진행된다면, VIS의 활용성은 더욱 증대될 것으로 기대된다.

알림

(1) 이 연구는 원자력 기반 분야 원전계측제어시스템 개발 사업으로 수행되는 디지털 계측 제어 인허가 확보기술 개발 과제에서 수행되었음.

(2) 모델체커 VIS는 공개 소프트웨어임.

참고문헌

[1] IEEE, "IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations", IEEE Std 7-4.3.2-1993, Sep. 15, 1993.

[2] IEEE, "IEEE Standard for Software Verification and Validation Plans", ANSI/IEEE Std. 1012-1986, Feb. 10, 1987.

[3] IEEE, “Standard for Developing Software Life Cycle Processes,” IEEE Std. 1074-1997.

[4] R. Alur and T. Henzinger . VIS: A system for Verification and Synthesis", The VIS Group, In the Proceedings of the 8th International Conference on Computer Aided Verification, p428-432, Springer Lecture Notes in Computer Science, #1102, New Brunswick, NJ, July 1996.

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