KIPS Tr. Software and Data Eng.
Vol.6, No.4 pp.167~176 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.4.167

Using Model Checking to Verify an Automotive Electric Parking Brake System

Jun Yeol Choi[†] · Joon Hyung Cho[††] · Yun Ja Choi[†††]

ABSTRACT

There are increasing policies and safeguards to prevent various human resource losses with the development of automotive industry. Currently ISO26262 1st edition has been released in 2011 to ensure functional safety of electrical and electronic systems and the 2nd edition will be released in the second half of 2016 as part of a trend. The E/E (Electrical & Electronics) system requirements verification is required through walk-through, 인스펙션, semi-formal verification and formal verification in ISO 26262. This paper describe the efficiency of model checking for the E/E system requirements verification by applying the product development project of ASIL (Automotive Safety Integrity Level) D for the electrical parking brake system.

Keywords : Model Checking, Automotive Verification, Formal Verification, ISO26262, Safety Requirement

자동차 전자식 주차 브레이크 시스템 안전 요구사항 검증을 위한 모델검증 적용

최 준 열[†] · 조 준 형[††] · 최 윤 자[†††]

요 약

자동차 산업이 급격히 발달하면서 각종 인명손실을 예방하기 위한 정책 및 안전장치가 늘어나고 있다. 트렌드의 일환으로 2011년에 차량의 전기전자시스템의 기능안전성을 확보하기 위한 ISO26262 1st edition이 릴리즈 되었으며, 2016년 하반기에 2nd edition이 릴리즈 될 예정이다. ISO 26262에서는 안전 요구사항에 대해 Walk through, 인스펙션, 준정형 검증, 정형 검증을 통해 전기전자시스템 요구사항에 대한 검증을 요구하고 있다. 본 논문에서는 ASIL (Automotive Safety Integrity Level) D등급의 전자식 주차 브레이크 양산 프로젝트의 전기전자시스템 요구사항 검증에 모델검증을 적용함으로써 전기전자시스템 요구사항 검증 시 모델검증의 효율성을 기술한다.

키워드 : 모델검증, 자동차 검증, 정형 검증, ISO26262, 안전 요구사항

1. 서 론

자동차 제어 시스템은 결함발생 시 인명 손실을 유발할 수 있는 안전필수 시스템으로 자동차 전기전자 시스템의 안전관련 국제표준인 ISO26262에서는 전기전자시스템의 요구사항에 대한 검증 및 확인을 필수적으로 요구하고 있다[1, 2]. ISO26262에서 요구하는 전기전자시스템 요구사항 검증의 필요성을 많은 고객 및 개발자들이 인지하고 있지만 현재 대부분의 자동차 회사에서는 전기전자시스템의 요구사항에 대해서 효율적인 검증을 수행하지 못하고 있다. 실제로 Walk through, 인스펙션 수준으로 동료검토를 수행하더라도 시스템이 거대해지고 복잡해질수록 사람이 논리적으로 요구사항들을 생각하는 것에는 한계가 존재하는 등 전기전자시스템의 요구사항 검증을 위해 투자되는 시간과 효율성은 떨어진다. 잘못되거나 미흡한 전기전자시스템 요구사항은 주로 테스트 단계에서 결함 및 설계오류로 발견되며, 이로 인해 소프트웨어의 구조가 비효율적으로 되거나 뒤늦은 설계변경으로 인한 추가자원이 필요하게 되는 행위가 자주 발생되고 있다.

대부분 양산 자동차에 적용되는 시스템은 선행연구나 벤치마킹을 통해 무엇을 구현해야 하는지 목적과 대상을 명확히 아는 상태에서 설계를 시작한다. 하지만 많은 OEM(Original Equipment Manufacturing)의 서로 다른 고객 요구사항을 적

† 준 회 원 : 경북대학교 컴퓨터학부 박사과정
†† 비 회 원 : 만도 CBS 2실 설계1팀 팀장
††† 정 회 원 : 경북대학교 컴퓨터학부 교수
Manuscript Received : September 22, 2016
Accepted : October 27, 2016
* Corresponding Author : Jun Yeol Choi(trustcjy@gmail.com)
Joon Hyung Cho(Joonhyung.cho@halla.com)
Yun Ja Choi(yuchoi76@knu.ac.kr)

용하다 보면 요구사항간의 충돌로 인하여 테스트단계에서 예상과는 다른 성능을 보여주는 경우가 많이 있다. 이러한 경우 시스템 요구사항 명세서 수정부터 소프트웨어 구조변경까지 수정이 필요하게 되며, 구현된 소스코드는 유지보수가 어려운 복잡한 구조를 가질 확률이 높아진다. 이러한 문제점을 개선하기 위해 소프트웨어 구현 이후 단계가 아닌 고객 요구사항 분석 및 설계 단계에서부터 전기전자시스템 요구사항 검증은 ISO26262에서 요구하지 않더라도 반드시 필요하다는 것을 알 수 있다.

본 연구에서는 실제 양산되는 ASIL D등급의 자동차 전자식 주차 브레이크 시스템 프로젝트에서 OEM의 고객 요구사항과 안전목표를 바탕으로 도출 된 기능 안전 요구사항이 반영 된 시스템 요구사항과 기술적 안전 요구사항 검증을 위해 모델검증을 적용하였다. 모델검증의 적용 시점 및 방법, 프로젝트 수행에 있어 모델검증을 수행함으로써 얻을 수 있었던 개선점을 기술한다. 시스템 요구사항은 시스템의 기능을 구현하기 위해 설계 되어 도출 된 요구사항을 의미하며 기술적 안전 요구사항은 시스템의 기능과는 별개로 기능 안전 요구사항을 충족하기 위한 요구사항이다. 시스템 요구사항과 기술적 안전 요구사항은 모두 시스템 설계 단계에서 도출되는 요구사항이다.

본 논문의 구성은 다음과 같다. 2장에서는 전자전기시스템 개발과정과 모델검증에 대해 서술한다. 3장에서는 전자식 주차브레이크 시스템의 기본기능을 설명하고, 시스템 요구사항을 모델링 한 뒤 모델검증을 통해 요구사항의 속성을 검증하고 결과를 바탕으로 시스템 요구사항을 개선하는 과정에 대해 서술한다. 4장에서는 본 연구에 대한 평가를 기술하며 5장에서는 결론 및 향후 계획에 대해 서술한다.

2. 관련배경

2.1 시스템 개발과정

기본적으로 자동차 산업은 Fig. 1과 같이 ISO26262, A SPICE, CMMI 등에서 언급하는 V모델기반으로 시스템을 개발한다[3]. V모델은 소프트웨어 개발 프로세스 중 폭포수 모델을 확장한 모델로 폭포수 모델에 검증부분을 강화한 방법론이다. 특별히 V 모델에 안전을 특화 시킨 Spiral 모델을 사용하는 개발 프로세스도 간혹 사용되지만 대부분의 시스템은 V 모델에 기반하여 개발을 진행 한다. 본 논문에서 언급하는 전자식 주차 브레이크 시스템의 양산 프로젝트도 V 모델을 기반으로 진행되었다.

먼저 고객으로부터 고객 요구사항을 제공받아 어떠한 기능을 수행하는 시스템인지 명확히 하고 고객 요구사항을 바탕으로 개념을 설계한다. 안전목표 및 기능 안전 요구사항 명세를 도출하고 도출한 명세를 바탕으로 시스템, 하드웨어 및 소프트웨어를 설계한다[5]. 설계가 완료되면 설계한 시스템 요구사항 명세서, 구조 설계서, 인터페이스 명세서 등의 문서를 참조하여 하드웨어와 소프트웨어 구현을 한다. 시스템 요구사항 명세서는 시스템 요구사항과 기술적 안전 요구사항 2가지 성격의 요구사항을 포함한다. 하드웨어 및 소프트웨어를 구현 한 뒤 테스트를 통해 설계명세서에서 요구하는 기능이 제대로 구현 되었는지 검증을 한다. 소프트웨어 및 하드웨어 검증이 완료되면 통합을 하여 시스템 통합시험을 수행하며 시스템 검증이 완료되면 고객이 요구하는 기능이 구현되었는지 확인하는 작업을 수행한다. 시스템을 개발하는 Supplier에서 시스템 검증을 완료하면 OEM은 OEM에서 제공한 고객 요구사항을 바탕으로 인수시험을 수행한다. 인수시험을 거쳐 시스템을 확인하는 단계가 완료되면 비로소 시스템 양산을 시작한다[3, 4].

2.2 모델검증

정형 검증은 최근 안전성이 필요한 분야에서 요구사항 검증을 위해 권장된다. Fig. 2는 ISO26262 문서 내에 존재하는 요구사항 검증 방법에 대한 Table 이다.

‘o’ 기호는 권장하지 않는다는 의미이다. ‘+’ 기호는 수행을 권장한다는 의미이며 ‘++’ 기호는 강력하게 권장한다는 의미이다. 제시된 내용처럼 ISO 26262에서는 모든 시스템에 대해서는 요구사항 검증을 수행을 요청하고 있다.

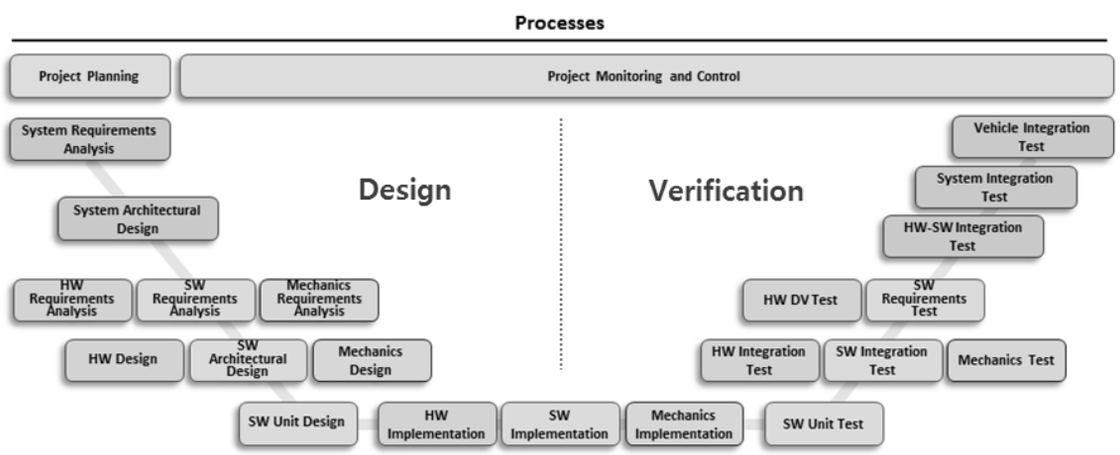

Fig. 1. V Model

Table - Methods for the verification of requirements

Methods		ASIL A	B	C	D
1a	Informal verification by walkthrough	++	+	o	o
1b	Informal verification by inspection	+	++	++	++
1c	Semi-formal verification (a)	+	+	++	++
1d	Formal verification	o	+	+	+
a Method 1c can be supported by executable models.					

Fig. 2. Methods for the Verification of Requirements

ASIL C ~ D에 대한 시스템에 대해서는 인스펙션뿐아니라 Semi-formal verification도 강력하게 권장하고 있으며 Formal verification도 '+'로 권장하고 있다. 이러한 요구사항에 대한 검증필요성은 자동차 분야의 ISO 26262뿐만 아니라 항공관련 DO-178, 철도관련 IEC 62278, 전기전자 임베디드 관련 IEC61508등의 문서에서도 동일하게 언급된다.

모델검증은 정형 검증 기법의 하나로 정형명세로부터 생성된 유한모델과 시스템이 만족해야 할 요구사항이 갖추어졌을 때, 시스템이 요구사항을 만족하는지 논리 알고리즘을 통해 확인하는 방법이다. 시스템에서 검증 대상을 선택하고 검증 대상에 대해 유한 모델을 생성한다. 생성 된 유한 모델에서 시스템이 만족해야 할 속성을 충족하는지 확인할 수 있다. 만약 유한 모델이 검증하고자 하는 속성을 만족하지 못하는 경우 반례를 보여준다. 개발자는 반례를 통해 시스템의 문제점을 빠르게 파악할 수 있다. 모델검증은 모든 과정이 자동화가 가능하여 사용이 편리한 장점이 있지만 상태가 많아지면 상태 폭발 문제로 인해 검증이 어려워진다[8-10].

모델검증은 시스템을 검증한다는 목적에서 테스트와 같지만 검증하는 방법이 다르다.

Fig. 3을 보면 알 수 있듯이 모델검증은 시스템을 검증하기 위해 검증대상의 수행행위를 모델링 한다. 이 과정에서 검증하고자 하는 속성에 따라 추상화가 가능하며 정형명세된 수행행위를 검증하고자 하는 속성으로 논리적으로 검증한다. 반면 테스트는 발생할 수 있는 경로에서 가질 수 있는 모든 경우의 수 중에서 샘플링을 통해 검증한다.

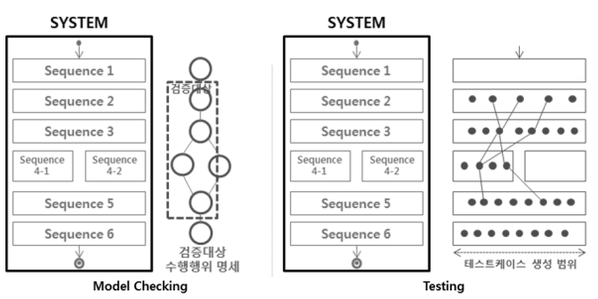

Fig. 3. Model Checking and Testing

즉 테스트로는 모든 경우의 수를 확인하기 어렵기 때문에 시스템이 오류가 없음을 증명하기 어렵다. 하지만 모델검증은 검증범위 내에서 상태에 대한 논리적 관계를 검증하기 때문에 검증범위에서 검증 속성에 대한 오류가 없음을 증명할 수 있다[6].

본 연구에서 정형 검증을 위해 사용한 UPPAAL은 실시간 시스템의 검증을 위한 모델검증 툴 중 하나이다. 현재 UPPAAL은 communication protocols 등의 미들웨어에서 Multimedia 등의 응용프로그램까지 다양한 분야의 시스템 검증을 위해 사용되고 있다[11-16]. UPPAAL은 timed automata 이론 기반의 모델 체커이며 UPPAAL의 모델링 언어는 integer variables와 urgency 등의 추가적 특징을 제공한다[18].

UPPAAL에서 시스템은 non-deterministic processes, 유한 제어 구조와 real-valued clocks으로 모델링되며 channels와 shared data structures를 통해 통신하게 된다. UPPAAL은 크게 GUI, 검증 서버, 명령라인 툴, 3가지로 구성된다. GUI는 모델링, 시뮬레이션, 검증을 위해 사용된다[19]. 서버는 시뮬레이션에서 모델링 된 범위의 성공적 상태들을 계산하기 위해 사용된다. 명령라인 툴의 경우에는 일괄검증 같은 독립형 검증기이다.

3. 시스템 요구사항

3.1 전자식 주차 브레이크 시스템

전자식 주차 브레이크 시스템은 자동차의 제동 및 운행과 관련된 시스템이기 때문에 높은 안전성을 요구하는 제품이다. 본 논문에서는 전자식 주차 브레이크 시스템 중 VDA 305-100의 내용을 준수하여 개발되는 Motor on Caliper Integrated Type 전자식 주차 브레이크 시스템 프로젝트 기준으로 기술한다[17]. Motor on Caliper Integrated Type는 전자식 주차 브레이크 시스템의 소프트웨어와 하드웨어를 안전 개념을 적용하여 2개의 제조사에서 개발하는 Type이다.

전자식 주차 브레이크 시스템은 자동차의 바퀴에 장착되어 운용되며 Fig. 4에서 알 수 있듯이 자동차 외부환경과 운전자의 상태 및 의지를 확인하여 Apply, Release, Diagnosis를 수행하는 시스템이다. 전자식 주차 브레이크 시스템의 Apply, Release, Diagnosis기능은 부수적으로 Re-clamp, Deceleration, Slip제어 등의 기능을 포함하고 있다.

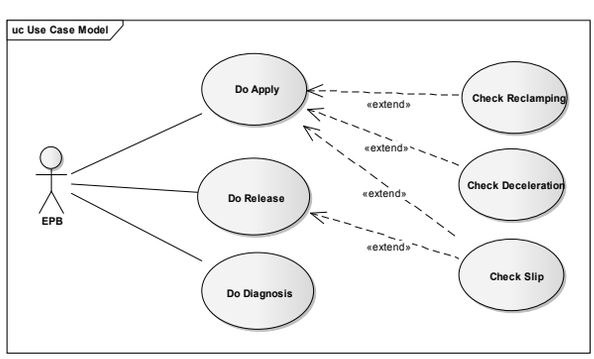

Fig. 4. Electric Parking Brake System Use Case Diagram

기본적으로 자동차가 운행 중에는 전자식 주차 브레이크 시스템이 제동력을 상황에 따라 적절하게 발생시켜 자동차의 주차를 지원할 수 있어야 하며, 주차 후에는 자동차가 제동

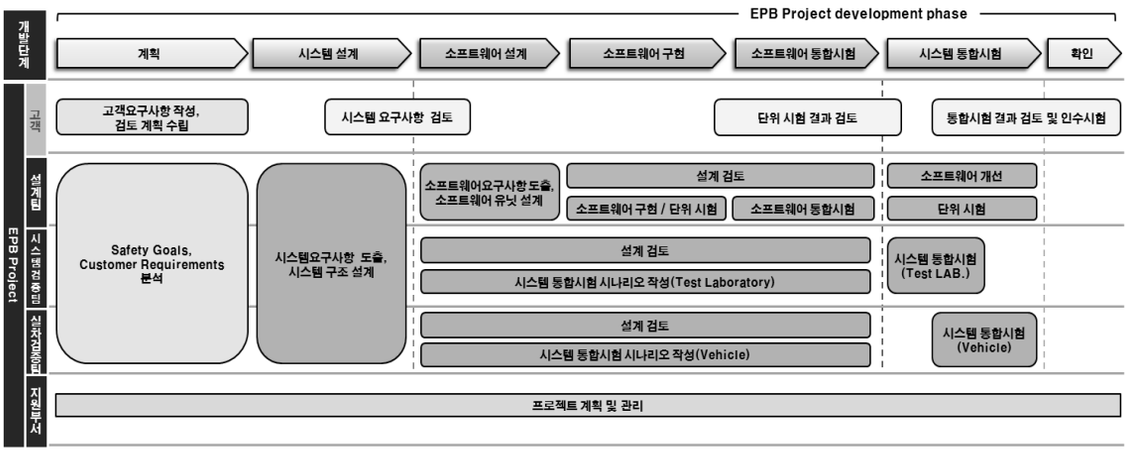

Fig. 5. Electric Parking Brake Software Development Process

력을 잃지 않도록 re-clamp기능을 통해 제어할 수 있어야 한다. 또한 자동차 주행 중 타 감속 시스템 등의 고장으로 감속이 필요하다면 전자식 주차 브레이크를 통해 자동차가 안정성을 유지하며 속도를 감속할 수 있도록 Deceleration 기능을 통해 제어할 수 있어야 한다. 추가적으로 간격 조정, 패드교환 등 추가적인 유지보수 기능을 가지고 있어야 한다.

3.2 전자식 주차 브레이크 시스템 개발 프로세스

전자식 주차 브레이크 시스템 개발은 Fig. 5와 같이 계획, 시스템 설계, 소프트웨어 설계, 소프트웨어 구현, 소프트웨어 통합시험, 시스템 통합시험, 확인 순으로 크게 7개의 절차로 개발되었다.

계획은 OEM으로부터 전자식 주차 브레이크 시스템 고객 요구사항을 제공받아 전자식 주차 브레이크 시스템에 대해 개념을 확인한다. 또한 안전 목표 및 기능 안전 요구사항을 충족시키기 위해 프로젝트가 완료되어야 하는 시점까지 어떻게 개발할 것인지 계획하는 단계이다.

시스템 설계 단계는 고객 요구사항을 분석한 내용을 바탕으로 실제 개발을 위한 기술적 안전 요구사항을 도출하고 하드웨어 및 소프트웨어 영역에 할당하는 단계이다. 시스템 설계는 요구사항 도출 및 아키텍쳐 설계와 함께 FMEA (Failure Mode Effect Analysis), FTA(Fault Tree Analysis) 등의 안전분석을 바탕으로 성숙도를 올린다. 설계내용에 대한 검증이 충분히 이루어지면 시스템 설계는 베이스라인이 설정된다[5, 6]. 시스템 설계의 베이스라인이 설정되면 하드웨어와 소프트웨어 개발 조직에서는 시스템 설계자료를 기반으로 하드웨어 및 소프트웨어 개발을 시작한다.

각각의 하드웨어 및 소프트웨어 설계자는 시스템 영역에서 도출된 시스템 요구사항과 기술적 안전 요구사항 중 해당 영역에 할당 된 요구사항에 대해 분석을 수행하고, 수행한 결과를 바탕으로 하드웨어 및 소프트웨어 요구사항 도출 및 설계를 진행한다. 하드웨어 및 소프트웨어 영역에 대해서도 FMEA, FTA등을 통해 성숙도를 올리며 검증이 충분히 이루어지면 하드웨어 및 소프트웨어를 구현한다[7, 10, 20].

소프트웨어 구현은 기 도출된 소프트웨어 요구사항 명세서, 소프트웨어 구조 설계서, 소프트웨어 단위 설계서를 바탕으로 코딩을 진행하며 동시에 단위시험을 수행한다. 소프트웨어의 정적 분석 및 동적 분석이 충분히 수행되면 소프트웨어 통합시험을 수행하고, 소프트웨어 통합시험이 완료되면 소프트웨어 요구사항 명세서를 바탕으로 소프트웨어 요구사항 테스트를 진행한다.

요구사항 테스트를 모두 만족하면 하드웨어와 소프트웨어를 통합하여 하드웨어 및 소프트웨어의 인터페이스 테스트를 수행하여 시스템 통합시험이 가능한지 확인한다. 하드웨어와 소프트웨어 간에 인터페이스가 확인이 되었으면 시스템 요구사항 명세서를 바탕으로 시스템 테스트를 진행하고 진행된 시스템 개발 절차에 대해 전반적으로 검토한다.

이후에는 OEM이 개발된 시스템에 대해 고객 요구사항을 바탕으로 테스트를 진행하고 시스템에 대한 확인이 완료되면 프로젝트는 종료된다.

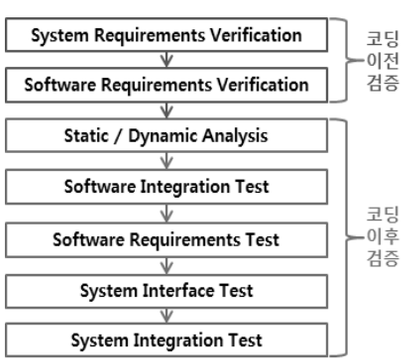

Fig. 6. System Verification Process

Fig. 6은 시스템을 검증하는 절차이다. V모델의 좌측에 해당하는 설계단계에서는 요구사항이 적절하게 작성되었는지 분석을 하는 단계이고, 소스코드를 작성하는 시점을 기준으로 우측은 설계대로 소프트웨어가 구현되었는지 검증단계로 정적 분석과 동적 분석을 필두로 테스트를 시작한다.

일반적으로 산업계에서는 시스템 요구사항과 소프트웨어 요구사항 분석을 위해 인스펙션을 수행한다. 정적 분석을 위해서 QAC, CodeSonar, PolySpace 등의 툴을 사용하며, 동적 분석, 소프트웨어 통합 시험, 소프트웨어 요구사항 시험을 위해 VectorCast, LDRA 등의 툴을 사용한다. 소프트웨어 요구사항 시험이 완료되면 시뮬레이터 환경에서 CANoe, vTestStudio 등의 툴을 사용하여 시스템 테스트를 수행하고 시뮬레이터 환경에서 시스템 테스트가 완료되면 실차 테스트를 진행한다.

본 연구에서는 모델검증을 System Requirements Verification 시점에 적용하였다. 모델검증으로 인스펙션의 부족한 점을 보완하면서 추후 테스트로 인해 발생되는 시스템 설계내용의 변경을 최소화 하였다.

3.3 UPPAAL 적용

1) 검증모델 생성

가장 먼저 UPPAAL을 사용하여 요구사항을 검증하는 시점을 고려하였다. 시스템 설계 문서가 베이스라인이 설정되고 시스템 설계 문서를 바탕으로 소프트웨어 설계를 시작하기 전 모델검증을 적용하였다. 개발초기에 결함이 발견되도록 하여 소프트웨어 설계 변경이 최소화 될 수 있도록 하였다. 전자식 주차 브레이크 시스템의 개발 프로세스 중 시스템 설계문서가 완성되고 설계문서에 대한 인스펙션이 완료된 시점에 UPPAAL을 이용하여 시스템 설계문서 중 시스템 요구사항 명세서에 기술된 시스템 요구사항을 검증하였다.

검증 하고자 하는 대상은 전자식 주차 브레이크 시스템에서 외부 정보를 제공받아 제공받은 정보를 바탕으로 현재의 Actuator의 상태를 판단하는 Actuator Status 판단부분을 검증 대상으로 선정했다. 그 이유는 실제 Actuator에 명령을 전달할 부분이며 Actuator Status 판단부분에서 결함이 발생할 경우 하드웨어의 파손은 물론 최악의 경우 안전 사고가 일어날 수 있기 때문이다.

검증 대상을 명확히 하고 나서는 검증 대상에 대한 모델링을 시작한다. 전자식 주차 브레이크 시스템을 모델링 하기 위해 사용될 UPPAAL의 기능은 System Description기능의 Editor 탭, Symbolic Simulator 기능의 Simulator 탭, Verifier 기능의 Verifier 탭 총 3가지이다.

System Description은 Fig. 7과 같이 Declarations, Templates, System Definition 3가지 영역으로 구성되어 있다.

Actuator Status 판단부분 검증을 위해 Actuator Status 판단부분과 연관이 있는 총 17개의 검증대상 프로세스를 생성하였다. 본 논문에서는 17개의 검증대상 중 Host Command, Host Availability Left, Host Availability Right, Actuator Status Left 그리고 Actuator Status Right 총 5개의 프로세스에 대해 기술한다.

Host Command와 Host Availability는 외부에서 제공하는 정보를 받는 프로세스로 전자식 주차 브레이크 시스템 내부에서는 제공받은 Host Command 및 Host Availability 정보들을 판단하여 Actuator Status의 상태를 결정하고 Motor를 제어한다.

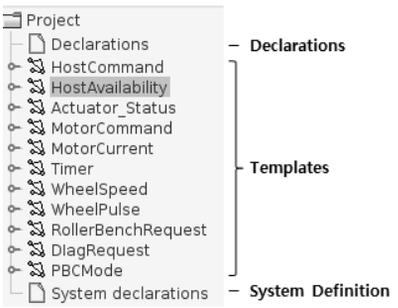

Fig. 7. UPPAAL Editor Tab

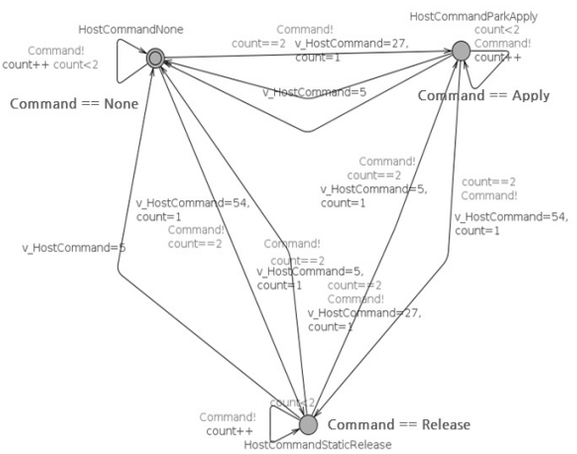

Fig. 8. Host Command Process

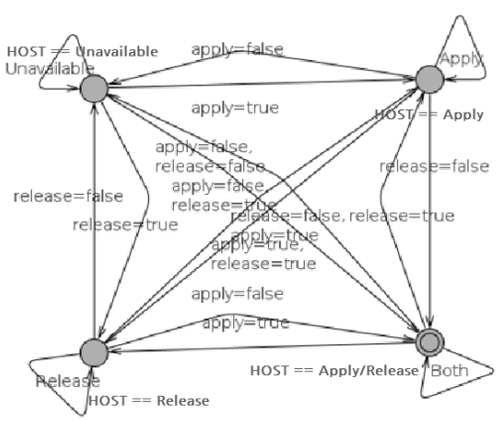

Fig. 9. Host Availability Process

Fig. 8은 전자식 주차 브레이크 스위치 또는 외부 상황을 판단하여 Park Apply, Hold Apply, Release, Dynamic Apply, Roller bench Apply, Auto Adjustment 요청을 받는 프로세스이다. 본 논문에서는 Apply 요청, Release 요청 2가지 location automata로 추상화시켜 표현한다.

Fig. 9는 전자식 주차 브레이크 시스템이 Apply 또는 Release가 가능한지, Apply와 Release 모두 가능한지 또는 모두 불가능한지에 대한 정보를 받는 프로세스이다.

Fig. 10은 검증모델의 중심이 되는 부분이다. Actuator Status Process는 타 프로세스의 정보를 바탕으로 모터의

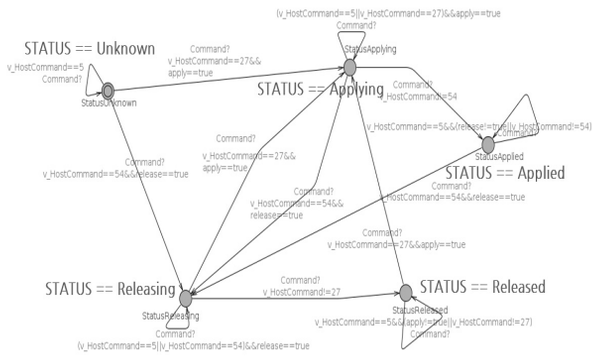

Fig. 10. Actuator Status Process

구동상태가 Park Applied, Hold Applied, Released, Applying, Releasing, Completely Released 또는 Unknown인지 판단하여 다음 Actuator 명령을 만들 수 있도록 상태를 결정하는 가장 중요한 프로세스이다. 본 논문에서는 Unknown, Applying, Applied, Releasing 그리고 Released 총 5개의 location automata로 추상화시켜 표현한다.

Host Command, Host Availability, Actuator Status 3개의 프로세스를 시스템 요구사항을 사용하여 모델링 하였다. 시스템 요구사항은 하나의 문장이 하나의 식별자로 만들어져 있으며 검증 대상의 시스템 요구사항은 총 807개의 요구사항으로 구성되었다. 그 중 본 논문에서 사용되는 Host Command, Host Availability, Actuator Status 3개의 프로세스를 모델링 하는데 사용된 관련 시스템 요구사항은 Table 1과 같이 시스템의 기능에 대해 기술하는 73개의 요구사항이다.

Table 1. Example of System Requirements

Function	System Requirements
Park Apply	PBC shall send the PbcOutHostCommand to 27 if PBC can check the actuator status.
Static Release	Static Release fuction to stop if the PbcInApplyReleaseRequest is 27 during operation.
Dynamic Brake	PBC shall send the PbcOutMotorCommandLeft signal to 54 after the EPB switch operation

Table 2. Parameters for System

Host Command Process		
Name	Value	Description
v_HostCommand	5	None
	27	Apply
	54	Release
Host Availability Process		
Name	Value	Description
Apply	True	Available
	False	Unavailable
Release	True	Available
	False	Unavailable

변수는 Table 2와 같다. 해당 변수 값을 Host Command 프로세스와 Host Availability 프로세스에서 업데이트 할 수 있도록 하였고 Actuator Status 프로세스에서는 업데이트 된 변수를 Guard로 활용하여 transition이 수행되도록 하였다.

2) 시뮬레이션

검증대상이 되는 시스템 검증모델이 완성된 뒤 Simulator Tab을 이용하여 Simulation을 수행하였다. Simulator는 시스템 모델의 가능한 동적 루트에 대한 검사를 수행할 수 있는 기능이다. Simulator를 통해 검증모델을 확인함으로써 모델 체커를 이용하여 속성을 검증하기 전 적은 비용으로 모델이 가지고 있는 결함을 발견할 수 있다. Simulator는 Fig. 11과 같이 simulation control, variables, processes, message sequence chart(MSC) 4개의 패널로 구성된다.

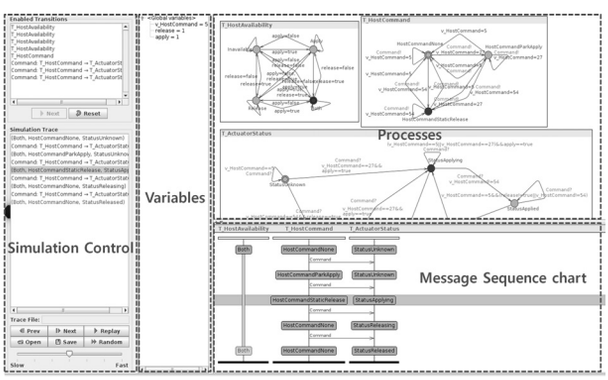

Fig. 11. UPPAAL Simulator Tab

Simulation을 통해 추상화된 모델이 예상되는 시나리오로 동작하는지 확인하여 추후 검증 시 제대로 된 결과를 도출할 수 있는지 검증하였다. 이 과정에서 검증대상에 대한 시스템 요구사항이 807에서 931개로 증가 되었으며 본 논문에서 설명되는 Host Command, Host Availability, Actuator Status 3개 프로세스에 관련된 시스템 요구사항은 73개에서 90개로 증가되었다.

추가된 요구사항은 설계자가 분석한 시스템 개념에는 있지만 요구사항으로 도출되지 않았던 요구사항 미정의 부분으로 볼 수 있다. 요구사항 미 정의로 인한 결함은 대부분 테스트 단계에서 발견되어 소프트웨어의 구조를 악화시키는 주범 중 하나로 소프트웨어 결함 원인 중 많은 비율을 차지하고 있다.

전자식 주차 브레이크 시스템 개발 프로젝트는 시스템 요구사항 명세서 및 시스템 구조 설계 문서가 작성되고 모든 문서에 대해 설계 관련 인원 9명의 인스펙션을 수행했으며, 뿐만 아니라 OEM과 함께 Formal meeting을 진행했음에도 불구하고 많은 부분의 시스템 요구사항이 누락되어 있었다. 가장 쉽게 발견할 수 있는 문제점은 고착상태를 확인할 수 있었다. 실시간 임베디드 시스템은 시스템이 시작되면 어떠한 이벤트 없이 지속적으로 서비스를 제공할 수 있어야 하며 검증대상인 전자식 주차 브레이크 시스템은 ASIL D의

안전성이 매우 중요한 시스템인 만큼 제어 시스템의 제어 불가 상태는 없어야 한다. 따라서 검증모델을 시뮬레이션 한 뒤 검증모델 교착상태가 없음을 확인할 수 있어야 한다.

전자식 주차 브레이크 시스템 개발에서는 교착상태를 확인하기 위해 Simulator Tab의 Random 기능을 주로 사용하였다. Random 기능은 Enabled Transitions을 자동으로 선택하여 시뮬레이션 되는 기능인데 Random 기능을 사용하면 시뮬레이션이 진행되다 Enabled Transitions이 없는 경우 'deadlock'을 반환하고 정지된다. 설계자는 'deadlock' 발생되면 Simulation Trace 파트의 진행경로를 확인하고 분석한 뒤 고착상태를 해결할 수 있는 시스템 요구사항을 추가하였다. 시뮬레이션 기능을 이용할 때 하나의 프로세스에 대한 영향을 시뮬레이션 하고 싶을 때는 모델링된 Locations에서 Transition을 수행할 Location들만 식별하여 Committed 옵션을 활성화하면 Committed된 Location으로만 제어된다. 시스템 요구사항, 구조 설계자료와 개념 등을 바탕으로 시뮬레이션을 통해 검증모델의 성숙도를 올린 뒤 검증모델이 검증되었다고 판단되면 검증모델을 통해 실제로 검증하고자 하는 속성을 Verifier Tab을 이용하여 검증한다.

3) 속성 검증

시스템 요구사항을 바탕으로 검증모델이 완성되면 UPPAAL Verifier Tab을 이용하여 기술적 안전요구사항 검증을 수행한다.

Fig. 12는 모델링 된 시스템에 대해 검증하고자 하는 속성을 작성할 수 있고 모델 체커를 이용하여 검증할 수 있다. Overview부분은 작성된 속성 값들을 List로 보여주며 Query 부분은 속성을 작성할 수 있다. 작성된 속성에 대해 검증을 수행하면 속성을 만족하는 경우 Overview에 표시되는 속성 오른쪽 원이 녹색으로 변경된다. 속성을 만족하지 못하는 경우에는 Overview에 표시되는 속성 오른쪽 원이 붉은색으로 변경된다.

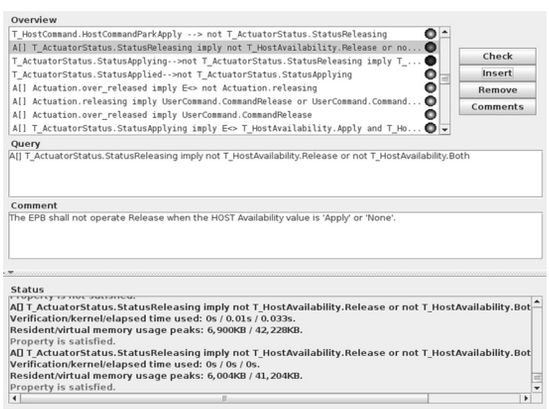

Fig. 12. UPPAAL Verifier Tab

또한 속성을 만족하는 경우에는 검증 경로 Trace를, 속성을 만족시키지 못하는 경우에는 반례를 Simulator Tab에 시현할 수 있다.

UPPAAL에서 지원하는 속성 타입은 Table3에서 확인할 수 있듯이 5종류이다.

기술적 안전 요구사항을 분석한 뒤 Verifier Tab의 Query를 이용하여 속성을 생성하였다. 전자식 주차 브레이크 시스템 개발 프로젝트에서는 기 도출된 137개의 기술적 안전 요구사항 중 점검 대상인 Actuator Status프로세스와 관련되는 79개의 기술적 안전 요구사항을 UPPAAL에서 제공하는 속성 타입으로 표현한 뒤 검증하였다.

Table 3. Property Equivalences

Name	Property	Equivalent to
Possibly	E<> p	
Invariantly	A[] p	Not E<> p
Potentially always	E[] p	
Eventually	A<> p	Not E[] not p
Leads to	p à q	A[] (p imply A<> q)

Table 4 내용의 일부를 예로 들어 살펴보면 "Electric Parking Brake System shall not operate RELEASE when the HOST Availability value is 'Apply' or 'None'" 기술적 안전 요구사항은 "A[] T_ActuatorStatus.Status.StatusReleasingimply not T_HostAvailability.Release or not T_HostAvailability.Both"로 UPPAAL 속성타입을 사용하여 표현하였다. 해당 속성의 의미는 Actuator 하드웨어가 Host availability 신호의 값이 'Apply' 또는 'None'인 경우 Host Command 프로세스의 Release 명령이 요청되더라도 Release기능을 수행해서는 안 된다는 의미이다.

Table 4. Example of Technical Safety Requirements

Technical Safety Requirements	Propoerty
Electric Parking Brake System shall not operate RELEASE when the HOST Availability value is 'Apply' or 'None'.	A[] T_ActuatorStatus.Status.StatusReleasing imply not T_HostAvailability.Release or not T_HostAvailability.Both
Electric Parking Brake System shall to be override the switch command(APPPLY->RELEASE, RELEASE->APPLY) while the motor is in operation	E<> T_ActuatorStatus.StatusReleasing(Start Location=StatusApplying)
The motor actuation shall continue to operate during normal mode.	A[] not deadlock
Electric Parking Brake System shall not operate APPLY after the parking force is generated to the maximum	T_ActuatorStatus.StatusAppliedànot T_ActuatorStatus.StatusApplying

해당 기술적 안전 요구사항을 검증하기 위해 Host Command 프로세스와 관계없이 Host Availability프로세스의 파라메터 값 중 Release파라메터의 값이 False인 경우 Actuator Status의 Location의 상태가 StatusReleasing이 될 수 없다는 속성

을 검증하였다.

이러한 방법으로 기술적 안전 요구사항에 대해 분석한 뒤 기술적 안전 요구사항이 요구하는 목적을 검증모델에 맞게 검증 속성으로 작성한 뒤 검증하였다.

4. 적용결과 및 평가

전자식 주차 브레이크 양산 프로젝트에 모델검증을 적용하면서 프로젝트 일정에 차질이 생기지 않도록 전체 시스템에서 주요부분을 식별하였고 해당부분만을 검증 모델로 작성하였다.

검증대상에 대한 모델링을 완성하는데 약 12일 정도의 시간이 소요되었다. 검증모델이 완성된 후 79개의 기술적 안전 요구사항을 검증하였다. 검증을 위한 UPPAAL 운용환경은 Intel i7-4810MQ의 CPU, 8GB의 RAM, 64bit 운영체제였으며 79개의 기술적 안전 요구사항을 검증하는데 소요된 시간은 총 42.8초가 소요되었다.

속성 검증을 통해 기 도출 된 79개의 기술적 안전 요구사항 중 8개의 속성을 만족하지 못하는 것을 확인하고 불만족한 8개의 속성을 만족시키기 위해 시스템 요구사항을 추가하였으며 그로 인해 시스템 요구사항은 931개에서 955개가 되었다. 불만족한 시스템 요구사항 중에는 “Electric Parking Brake System shall not operate APPLY after the parking force is generated to the maximum.”도 있었다. 전자식 주차 브레이크 시스템의 제동력이 최대 크기로 발생된 상태에서는 Apply 기능을 수행하면 안 된다는 의미이다. 해당 문제는 시스템 요구사항 중 “Motor command left / right can send APPLY message when actuator left / right status was APPLIED”의 잘못된 시스템 요구사항으로 발생하였고 기술적 안전 요구사항을 충족시키기 위해 잘못된 시스템 요구사항은 삭제되었다. 만약 기술적 안전 요구사항 검증이 시스템 테스트 단계에서 처음으로 수행됐다면 문제점은 뒤늦게 확인 되었을 것이고, 시스템 설계문서의 변경뿐만 아니라 소프트웨어 설계까지 변경을 해야 했다. 최악의 경우에 시스템 테스트에서 전자식 주차 브레이크의 제동력이 이미 최대 크기로 발생되어 있는 상태에서 제동력을 추가 발생시키는 동작을 수행 함으로써 테스트용 하드웨어를 손상시켜 프로젝트 비용을 증가시킬 수도 있었다. 해당사례와 같이 모델검증을 통해 시스템 요구사항의 성숙도를 증가시킬 뿐 아니라 속성 검증을 통해 잘못된 시스템 요구사항까지 발견할 수 있었다.

본 연구에서는 검증대상을 축소시켜 진행했음에도 불구하고 많은 결함을 찾을 수 있었다. 특히 반례를 통한 원인 분석은 시스템을 매우 효율적으로 분석 할 수 있었으며 빠르게 부족한 부분을 찾을 수 있게 하였다.

Fig. 13은 “Electric Parking Brake System shall not operate APPLY when the HOST Availability value is ‘Release’ or ‘None’”의 기술적 안전 요구사항을 검증한 그림이다.

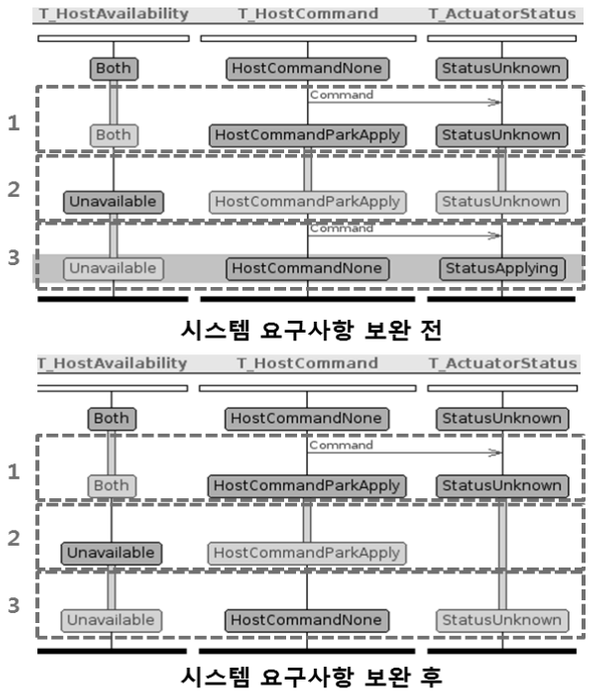

Fig. 13. Simulation Trace MSC

속성을 충족 못하는 시스템 요구사항 보완 전 그림에서는 1번 구간에서 Apply 명령이 수행된 뒤 2번 구간에서 Host Availability가 Unavailable로 바뀌더라도 3번 구간에서 Actuator Status는 Applying상태로 변경되었다. 즉 요구사항을 도출할 때 Actuator의 Apply기능이 수행된 뒤 Host Availability의 변경 영향성에 대해서는 생각하지 못했던 것이다. 반례를 통해 원인을 빠르게 분석할 수 있었고 적절한 요구사항을 추가 정의함으로써 현재는 Apply 명령이 수행된 뒤 Host Availability의 상태가 변경되더라도 Actuator Status는 Apply를 수행하지 않는다는 것을 보장한다

반례를 통한 원인분석은 인스펙션과 비교했을 때 매우 효율적이었다 만약 모델 체커를 통해 검증하지 못했다면 Actuator Status에 대해 Host Command와 Host Availability의 영향성을 분석하기 어려웠을 것이다. 더욱이 Actuator Status 프로세스가Left, Right 2개가 되며 Current, Wheel speed, Diag Request 등의 추가적인 프로세스들의 영향성까지 고려한다면 검증 툴의 도움 없이는 검증이 불가능 했다.

기 기술된 것과 같이 ISO 26262 표준에 따라 개발되는 전자식 주차 브레이크 시스템은 9명이 참석한 인스펙션을 수행하였다. 잘 정의된 요구사항을 검증하기 위해 2시간의 사전회의와 40시간의 검토시간이 소비 되었고 추가로 8 시간의 인스펙션 회의시간이 투입되었다. 또한 요구사항을 검증하기 위한 98개의 체크리스트 항목 작성시간도 추가되었다. 하지만 인스펙션의 결과는 16개의 Major Issue와 오타 등 31개의 Minor Issue만이 발행되었다.

Table 5에서 요구사항의 검증에 있어 인스펙션보다 모델검증의 효율이 좋다는 것을 알 수 있다. 인스펙션으로 인해 791개에서 807개로 증가한 요구사항은 모델검증을 사용함으

로써 955개의 요구사항으로 늘어났다. 그만큼 시스템은 정교하게 기술될 수 있었다. 기존의 요구사항 개수대비 증가한 비율만 보더라도 인스펙션은 약 2.0%에 그친 반면 모델검증은 약 18.2%의 요구사항 개수가 증가하였다. 모델검증 검증으로 삭제 된 불필요한 시스템 요구사항까지 고려하면 효율성은 더욱 차이 난다.

Table 5. The Ratio of Inspection and Model Checking

The number of found major faults	
Inspection	Model Checking
16	148

투입된 공수 및 추후 시스템 유지보수 시 재활용될 검증 모델까지 비교하지 않더라도 모델검증 툴 사용의 효율이 좋다는 것을 알 수 있다. 실제로 인스펙션 수행 결과를 확인해보면 요구사항 검증에 있어 문법적 오류나 오타, 3개 이하의 적은 수의 요구사항이 같은 페이지에 기술된 경우에는 검증하는데 좋은 효율을 보였지만 관계 된 요구사항들이 다른 페이지에 기술되거나 관계가 복잡해질수록 효율이 급격히 떨어짐을 알 수 있었다.

Table 6은 인스펙션과 모델검증을 통해 발견된 결함들의 유형이다. 결함 유형을 보면 인스펙션은 대부분이 개별 요구사항의 부적절함을 발견하였으며, 모델검증의 경우 개별 요구사항의 적절함보다는 다수 요구사항 간의 논리적 오류를 발견하였다.

Table 6. Types of Defects

	Types
Inspection	Grammatical error, Wrong scope of requirements, Type errors, Traceability of requirements
Model Checking	Logical error of requirements, Undefined requirements

5. 결론 및 향후 계획

본 연구를 통해 요구사항 검증의 중요성을 알 수 있었다. 인스펙션을 수행하더라도 설계된 요구사항에는 많은 결함이 있을 수 있고 이는 시스템의 정확성과 안전성을 떨어뜨린다는 것을 알 수 있었다.

또한 연구를 통해 인스펙션과 모델검증간에 발견할 수 있는 결함의 유형도 다르다는 것을 알 수 있었다. 인스펙션을 통해 요구사항 하나하나의 완성도를 올릴 수 있으며 모델검증을 통해 요구사항 간의 논리적 결함을 발견할 수 있다는 것을 알게 되었다.

시스템을 설계하다 보면 개발자의 입장에서는 분명 중요하게 생각해야 할 부분이 생길 것이고 우선적으로 검증해야 될 부분을 생각하게 된다. 만약 소스코드 작성단계 이후에 해당부분을 검증하고자 한다면 코드가 복잡할 수록, 다른 부분과 상호작용이 많아 질수록 검증하기는 어려워진다. 이 경우 모델검증을 이용하여 적은 비용으로 원하는 부분을 검증할 수 있다.

본 연구에서는 시스템 개발 프로세스에서 시스템 요구사항 분석 영역에서만 모델검증을 사용하였다.

시스템 설계단계에서 모델검증으로 많은 결함을 발견한 것과 같이 소프트웨어 설계단계에서도 좋은 효율을 낼 수 있을 것으로 기대되며 소프트웨어 설계단계에서도 소프트웨어 요구사항 검증을 위해 모델검증을 활용 할 예정이다.

또한 모델검증과 비슷한 개념으로 구현단계에서 Simulink나 Rhapsody 프로그램을 이용하여 시뮬레이션 또는 검증을 할 수 있다. 하지만 실제 구현 단계에서 사용할 수 있기 때문에 시스템이 많이 복잡해져 검증하는데 어려움을 겪게 된다. 무엇보다 모델검증 전용 툴보다 추상화가 어려워 시뮬레이션이나 검증을 수행할 때 리소스 부족에 자주 접하게 되어 시스템이 큰 경우에는 해당 기능을 사용하기가 어렵다. 추후 전자식 주차 브레이크 시스템 개발 프로젝트에서는 이러한 부분까지 모델검증을 접목하여 효율적으로 검증을 해보고자 한다. 모델검증을 이용하여 ISO 26262에서 요구하는 요구사항 검증의 효율을 증가 시킴으로써 시스템의 완성도를 올리기 위한 연구를 진행할 예정이다.

References

[1] ISO 26262 Functional Safety 2011, “Part8: Supporting Process,” The International Organization for Standardization, 2011.

[2] IEC 61508 Functional safety of electrical/electronic/ programmable electronic safety-related systems “Part 6: Guidelines on the application,” International Electronical Committee, 2010.

[3] RS Pressman, “Software engineering: a practitioner's approach,” Mc Graw Hill, 6th ed. 2005.

[4] W. Richards Adrion, “Validation, Verification, and Testing of Computer Software,” *Journal ACM Computing Surveys,* Vol.14, Issues 2, pp.159-192, June, 1982.

[5] ISO 26262 Functional Safety 2011, “Part1: Vocabulary,” The International Organization for Standardization, 2011.

[6] Ron Patton, “Software Testing (2nd Edition),” Sams Indianapolis, IN, USA, 2005.

[7] T. Maier, “FMEA and FTA to Support Safe Design of Embedded Software in Safety-Critical Systems,” Springer Safety and reliability of software based systems, pp.351-356, 1997.

[8] Magnus Lindahl, Paul Pettersson, and Wang Yi, “Formal design and analysis of a gearbox controller,” Springer International Journal of Software Tools for Technology Tansfer (STTT), 3: 353-368, 2001.

[9] A. Cimatti, E. Clarke, E. Giunchiglia, and F. Gjunchiglla, “An Opensource tool for symbolic model checking,” *14th International Conference, Computer Aided Verification*, pp.359-364, July, 2002.

[10] R. Alur, T.A. Henzinger, and M.Y. Vardi, “Theory in practice for system design and verification,” *ACM SIGLOG News,* Vol.2, Issue 1, pp.46-51, January, 2015.

[11] Pedro. R. D'Argenio, Joost-Pieter. Katoen, Theo C. Ruys, and Jan Tretmans, “The bounded retransmission protocol must be on time,” in *Proceedings of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, volume 1217 of LNCS*, pp.416-431. Springer-Verlag, April, 1997.

[12] Klaus Havelund, Kim G. Larsen, and Kristian Lund, “Formal verification of a power controller using the real-time model checker UPPAAL,” *5th International AMAST Workshop on Real-Time and Probabilistic Systems*, 1999.

[13] M. Hendriks, N.J.M. van den Nieuwelaar, and F.W. Vaandrager, “Adding symmetry reduction to uppaal,” in *Proceedings First International Workshop on Formal Modeling and Analysis of Timed Systems*, volume 2791 of Lecture Notes in Computer Science, 2003.

[14] J. Lahtinen, J. Valkonen, K.Bjorkman, J. Frits, I. Niemela, and K. Heljanko, “Model checking of safety-critical software in the nuclear engineering domain,” *Reliability Engineering & System Safety*, Vol.105, pp.104-113, September, 2012,

[15] Thomas Hune, Kim G. Larsen, and Paul Pettersson, “Guided synthesis of control programs using UPPAAL,” *IEEE ICDCS International Workshop on Distributed Systems Verification and Validation*, pp.15-22, IEEE Computer Society Press, April, 2000.

[16] Alexandre David and Wang Yi, “Modelling and analysis of a commercial field bus protocol,” *Proceedings of the 12th Euromicro Conference on Real Time Systems*, pp.165-172, IEEE Computer Society, 2000.

[17] VDA, “VDA Recommendation 305-100, Version August 2014,” Verband der Automobilindustrie, 2015.

[18] Hohn E. Hopcroft and Jeffrey D. Ullman, “Introduction of Automata Theory, Languages, and Computation,” Addison Wesley, 2001.

[19] Gerd Behrmann, Alexandre David, and Kim G. Larsen, “A Tutorial on UPPAAL 4.0.” www.uppaal.com, November, 2006.

[20] W. E. Vesely. “Fault Tree Handbook,” Technical Report NUREG-0492, US Nuclear Regulatory Commission, 1981.

최 준 열

e-mail : trustcjy@gmail.com

2012년 성균관대학교 컴퓨터공학과(석사)

2012년~2015년 한국항공 연구원

2013년~현 재 경북대학교 컴퓨터학부 박사과정

2015년~현 재 만도 CBS센터 설계1팀 연구원

관심분야 : 소프트웨어 안전성 분석, 결함 허용, 제어 소프트웨어

조 준 형

e-mail : Joonhyung.cho@halla.com

1996년 경북대학교 전자공학과(석사)

2010년~2015년 만도 CBS 2실 설계1팀 팀장

2016년~현 재 만도 CBS센터 설계1팀 팀장

관심분야 : 제어 소프트웨어

최 윤 자

e-mail : yuchoi76@knu.ac.kr

2003년 미네소타대학 전산과(박사)

2003년~2006년 프라운호퍼연구소 연구원

2016년~현 재 경북대학교 컴퓨터학부 교수

관심분야 : 소프트웨어 안전성 분석, 정형 검증, 모델기반 개발방법론