한국정밀공학회지 제 23 권 제 9 호 (2006 년 9 월)
Journal of the Korean Society for Precision Engineering, Vol. 23, No. 9, September 2006

◆특집◆ 자동차 제어 및 전자화 기술

모델기반의 자동차 전장 시스템 설계 및 검증 기술

손준우*, 금대현*, 이선봉*, 이석**

Model-Based Design and Validation Method for Body Electronic Systems

Joonwoo Son*, Daehyun Kum*, Seonbong Lee* and Suk Lee**

Key Words : Model-Based Development Process (모델기반 개발절차), Body Electronic System (자동차 전장 시스템), Electronic Control Unit(전자 제어 장치), In-Vehicle Network System (차량 내부 네트워크)

1. 서론

최근 자동차 전자제어 시스템이 점점 더 복잡해지고 차량에서 차지하는 가격 비중이 높아짐에 따라 전자 및 소프트웨어 오류로 인한 품질비용 발생 비중이 크게 증가되고 있다. 실제로 국내의 한 경제연구소 조사에 따르면 북미에서 자동차업계가 지출하는 품질보증 비용 중 30~40%가 전자 및 소프트웨어 문제에서 발생하고 있으며, 2004 년 독일 내에서 발생한 리콜 중 9%가 전자적인 문제로 보고되고 있다.[1] 특히, 전자부품이나 기기 자체보다 네트워크를 통한 정보교환이나 제어를 담당하는 소프트웨어에서 문제가 빈발한다는 점에서 신뢰성있는 소프트웨어 설계 및 검증 방법이 더욱 절실하다고 볼 수 있다.

따라서, 높은 품질 수준을 유지하면서 개발 기간 단축과 비용 절감을 동시에 만족시킬 수 있는 설계 및 검증 프로세스의 필요성이 대두되고 있으며, Fig. 1 과 같이 각 단계별로 모델을 이용하여 설계 및 검증을 동시에 수행할 수 있는 모델기반 개발 프로세스 (MBDP, Model-Based Development Process)가 이러한 자동차 산업의 요구를 만족시킬 수 있는 효과적인 개발 방법론임을 인정받고 급속히 확산되고 있다.

모델기반 개발 프로세스는 자동차 제어에 필요한 제어 알고리즘 또는 기능의 작동 방식을 상태도(Statechart), 블록선도(Block Diagram), 순서도(Flowchart) 등을 이용하여 기능 모델을 만들고 가상의 환경에서 시뮬레이션을 통하여 기능을 미리 검증하고자 하는 것이다.[2]

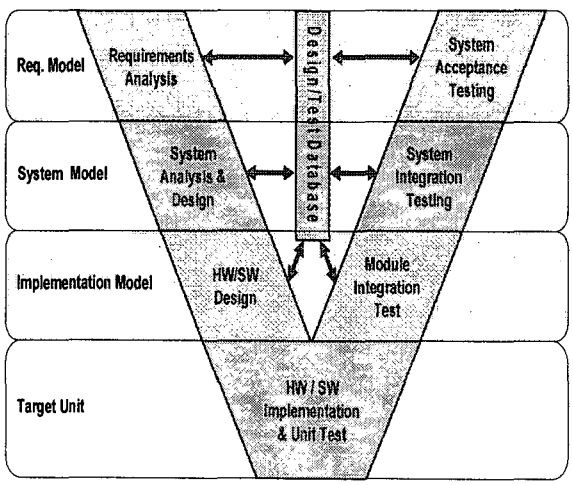

Fig. 1 Model-based development process and model phases

* 대구경북과학기술연구원
Tel. 053-430-8453, Fax. 053-430-8479
Email json@dgist.ac.kr
자동차 전장시스템 특히, 모델기반 시스템 설계 및 검증기술, 차량 내부 네트워크 설계 기술 분야에 관심을 두고 연구활동을 하고 있다.
** 부산대학교 기계공학부

이와 같이 검증이 완료된 모델을 이용하여 자동차 전자제어장치(ECU, Electronic Control Unit) 내에서 수행될 수 있는 프로그램 코드를 자동 생성할 경우, 개발 기간 단축은 물론 소프트웨어 엔지니어에 의하여 야기될 수 있는 오류를 최소화시킴으로써 소프트웨어 신뢰성을 향상시킬 수 있다.[3]

본 논문에서는 네트워크 기반의 자동차 전장 시스템을 보다 효율적으로 설계 및 검증할 수 있는 방법에 대하여 소개하고자 한다. 먼저 2 장에서 모델 기반 전장 시스템 및 네트워크 설계 기법을 소개하고, 3 장에서 모델기반의 자동 테스트 생성 및 수행 기법을 설명한 후, 4 장에서 간단한 사례 연구를 통하여 모델기반 설계 및 검증 절차의 유효성을 확인하고자 한다.

2. 모델기반 바디 네트워크 시스템 설계

2.1 바디 네트워크 시스템 설계 동향

자동차에 차량 내부 네트워크의 적용이 본격화 되면서, 전장품의 설계 방식이 기존의 부품 중심의 설계 최적화 방식에서 시스템 중심으로 변화되고 있다.[4] 이러한 변화는 전장 시스템의 설계에 있어서 보다 체계적인 접근 방법을 요구한다. 즉, 여러 개의 ECU 로 구성된 전장시스템을 설계할 경우 기존의 ECU 설계를 토대로 기능을 수정하던 방식을 탈피하여 ECU, 센서, 액츄에이터의 위치와 정보 교환 빈도 등을 고려하여 최적의 ECU 와 네트워크를 설계할 수 있는 방법이 요구되고 있다.

CAN 네트워크를 이용한 바디 네트워크 구현 분야에서 일어나고 있는 또 하나의 중요한 변화는 메시지 중심의 소프트웨어 설계에서 시그널 중심의 설계 방식으로의 전환이다. 이는 오늘 날 자동차용 운영체제 및 통신의 사실상 표준인 OSEK/VDX 표준이 전장 분야에서 적용되기 시작하면서 Fig. 2 와 같은 계층구조를 이용한 통신 소프트웨어 설계가 보편화되었기 때문이다.

모델기반 개발 프로세스는 이러한 시스템 중심의 설계 방식 및 시그널 중심의 설계 방식과 접목되어 보다 효율적인 네트워크 설계 및 검증 절차를 제공하여 준다.

2.2 모델기반 바디 네트워크 시스템 설계 프로세스

차량 내부 네트워크 개발을 위한 모델기반 개발 프로세스는 Fig. 3 에서 제시된 바와 같이 기능 모델을 네트워크 구조에 할당시키는 Allocation 단계가 중요한 역할을 수행한다.[2] 따라서 이번 절에서는 Allocation 기법을 이용한 개발 프로세스에 대하여 보다 상세히 설명하고자 한다.

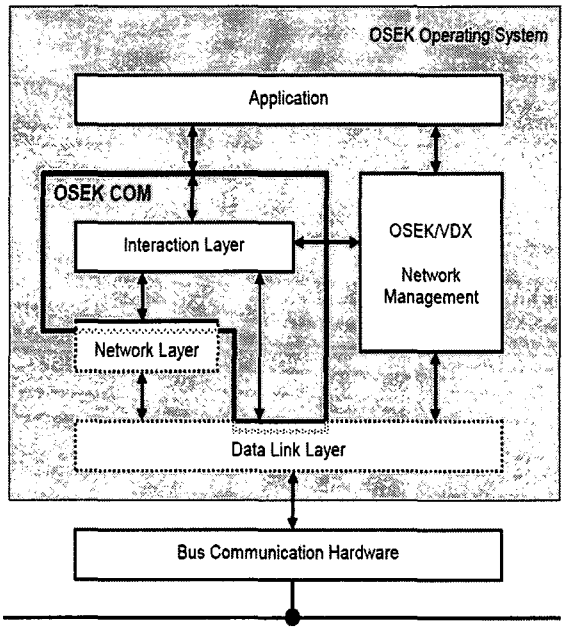

Fig. 2 Layered architecture of OSEK/VDX

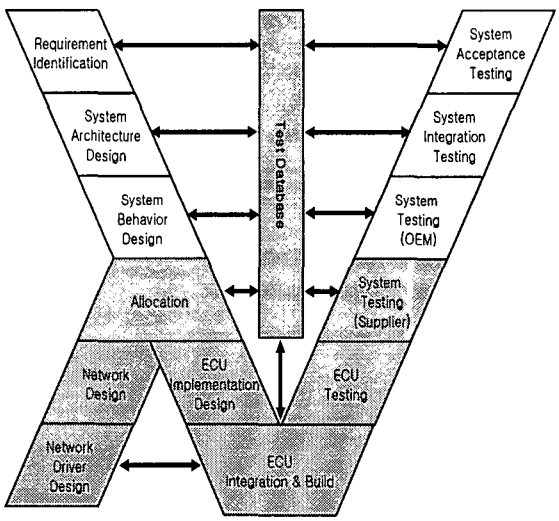

Fig. 3 Model-based automotive network system development process

2.2.1 기능 모델의 설계

Fig.3 에 도식화된 것과 같이 요구사항작성 및 분석을 완료한 후 기능모델을 설계하고 시뮬레이션을 이용하여 기능 검증을 수행한다. 전장시스템의 기능모델 설계는 Fig. 4 와 같이 요구사항을 몇 개의 기능그룹으로 분할하여 설계함으로써 시스템 모델의 복잡성을 줄이고 재사용성을 증대시킬 수 있다.[5] 전장시스템의 기능그룹은 일반적으로 계층적 기능 전개 Level 1 에서 전동 시트 제어, 전동 윈도우 제어, 램프 제어, 운전자세 기억 제어 등으로 구분하고, Level 2 에서 버튼입력제어, 구동모터 제어, 제어알고리즘 등의 세부 기능단위로 나눈다.

바디 전장시스템의 기능전개가 완료되면 상태도(Statechart)나 진리표(Truth Table) 등을 이용하여 기능모델 설계를 하고 시뮬레이션을 통하여 요구사항을 만족하는지 검증작업을 수행한다. Fig.5 는 ON/OFF 스위치에 의한 램프의 점등을 제어하는 메커니즘을 표현한 상태도의 예시이다.

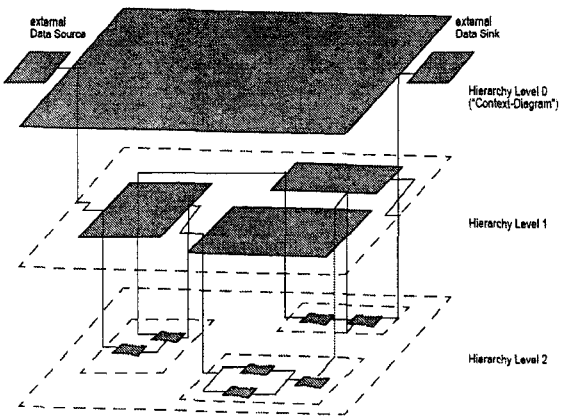

Fig. 4 Hierarchical functional decomposition

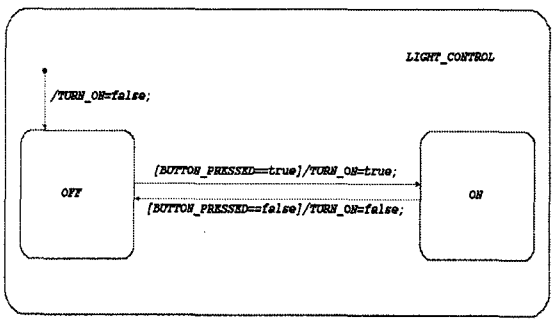

Fig. 5 Statechart to describe a simple control mechanism

2.2.2 바디 네트워크 구조 설계

바디 네트워크 설계 방법은 아직 표준화된 절차를 갖추고 있지 못하고 있으나 본 논문에서 제안하는 Allocation 기법을 이용할 경우, 전장시스템의 기능을 고려하지 않고 최적의 ECU 개수와 센서, 액츄에이터 등의 위치와 같은 물리적 특성만을 고려하여 네트워크 구조를 설계할 수 있다. 즉, 기존의 부품 중심 최적화 설계에서 발생할 수 있는 근접한 위치에 존재하는 다수의 ECU 들을 통합하고, 배선의 길이가 최소화될 수 있도록 ECU 와 센서 및 액츄에이터를 연결함으로써 네트워크 설계의 가격경쟁력을 향상시킬 수 있다.

이때, 각 ECU 에서 담당할 기능을 고려하지 않아도 된다는 점은 설계 유연성 향상에 크게 기여하게 된다.

2.2.3 ECU 의 담당 기능 할당

기능 모델과 네트워크 구조 설계가 완료되면 Allocation 단계에서 기능 모델의 세부 기능 단위를 네트워크 상의 ECU 에 할당하는 과정을 수행하게 된다. Fig.6 은 기능 모델과 네트워크 구조 간의 맵핑에 대한 개념도를 나타내며 수작업 또는 일부 모델링 툴에서 제공하는 기능을 이용하여 손쉽게 기능을 할당할 수 있다.

기능 모델의 할당에 따라 각 ECU 의 주요 사양 및 네트워크 성능에 많은 영향을 미치므로, Allocation 과정 중에 시뮬레이션을 이용하여 성능을 확인해 볼 필요가 있다.

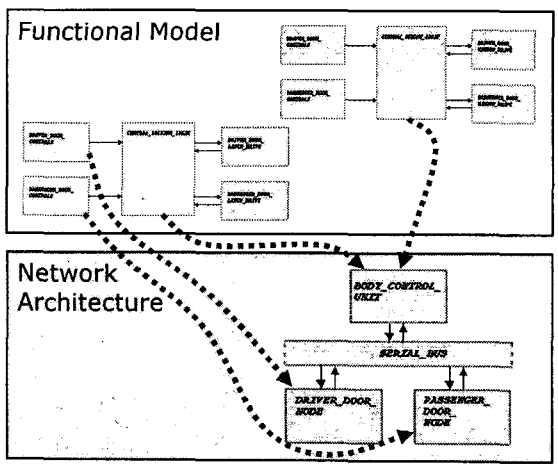

Fig. 6 Allocation concept for network system design

기능모델의 ECU 할당이 완료되면 ECU 구현 모델을 설계하게 되며, 다른 한편에서는 네트워크 구현을 위한 작업이 시작된다.

2.2.4 ECU 구현 모델의 설계

ECU 구현 모델은 ECU 별 기능 할당이 완료된 ECU 단위의 기능모델에 수행방식과 입출력 함수 정의 등을 추가하는 것이다.[3] 즉, 버튼제어, 모터 제어 등과 같은 세부 기능 단위를 주기적인 태스크로 수행할 것이지 인터럽트 서비스로 수행할 것인지를 정의하고, 모델에서 외부로 전달되는 변수와 하드웨어 구동 함수 간의 관계를 정의하여 자동 코드 생성에 필요한 정보를 완성시켜주는 것이다.

2.2.5 네트워크 구현

전송프로토콜(Transport Protocol), 네트워크 관리(Network Management) 등의 계층이 차량 네트워크에서도 활용되기 시작하면서 네트워크 구현 소프트웨어 설계가 점점 복잡해고, 보다 효율적인 네트워크 설계를 위하여 시뮬레이션을 이용한 설계 검증이 활발히 이루어 지고 있다.

네트워크 시뮬레이션 모델은 통신 모델뿐만 아니라 기능 모델도 포함하고 있어야 하는데, 기존의 개발 방식에서는 시뮬레이션 툴이 제공하는 스크립트 언어를 이용하여 기능을 개략적으로 묘사하는 방식으로 기능 모델을 대신하였다. 그러나 모델기반 개발 프로세스를 이용할 경우 Fig. 7 과 같이 검증된 기능 모델을 네트워크 시뮬레이션 모델에 삽입함으로써 보다 정교한 시뮬레이션 모델을 구성할 수 있다. 또한, 개발 초기에 시뮬레이션을 이용한 네트워크 설계 및 검증과정에서 기능 모델의 오류를 발견할 수 있다는 장점도 갖는다.

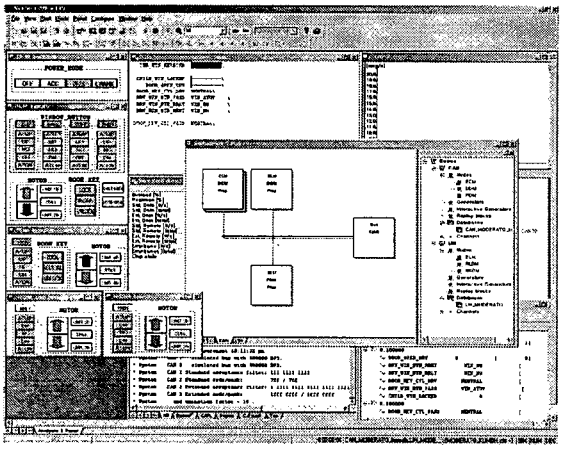

Fig. 7 Function and network co-simulation environment

3. 모델기반 자동 검증 기법

3.1 테스트 프로세스 개요

모델기반 개발 프로세스를 이용하여 전장시스템을 개발할 경우 모델기반 자동 검증 기법을 이용함으로써 보다 효과적인 모델기반 설계 및 검증 절차를 완성할 수 있다.[6] Fig.8 은 모델기반 개발 프로세스에서 단계별로 요구되는 테스트 프로세스를 요약한 것이다.

우선, 본격적인 테스트에 앞서 시스템에 대한 완벽한 이해와 분석이 있어야 한다. 즉, 시스템의 전체적인 구조 분석, 하드웨어와의 인터페이스 및 제약조건 파악, 소프트웨어의 기능 요구사항 등을 이해함으로써 테스트 되어야 할 정확한 입출력을 정의하고 테스트 사양을 정의한다. 입출력에 대한 정확한 정의가 바탕이 되어야만 정확한 테스트 결과를 얻을 수 있으며, 효율적인 자동화 환경을 구축할 수 있다.

다음 단계로 테스트 요구사항에 대한 테스트 시나리오 및 케이스를 작성한다. 테스트 케이스는 심플하고 정형적인 방법으로 표현되어야 하고 순차적으로 실행할 수 있도록 만들어 져야 한다. 복잡한 테스트 케이스나 시나리오로 인하여 또 다른 오류가 발생하지 않도록 해야 한다.

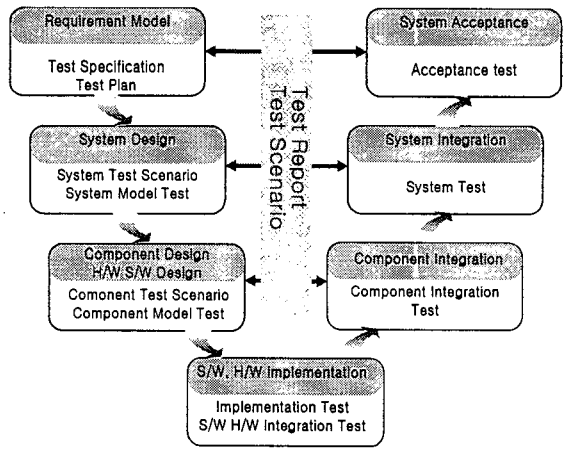

Fig. 8 Model-based development and test process

테스트 시나리오를 바탕으로 모델기반 프로세스에 따라 각 단계에 맞는 테스트 장비 및 시스템을 구축하여 테스트를 수행한다. 그리고 테스트의 전 과정에 대한 보고서를 작성하고, 테스트 결과 및 관련 정보들을 문서화하여야 한다.

모델기반 개발 프로세스의 첫 단계인 요구사항 단계에서부터 자동 검증 기법을 도입하기 위하여 요구사항을 정형언어(Formal Language)로 표현하기도 하지만 자동차 분야에서는 정형언어에 익숙하지 않은 조직과의 정보교환이 빈번하므로 문서기반으로 요구사항을 관리하는 것이 효과적이다 .[7] 시스템 설계 단계부터 시뮬레이션을 이용하여 모델을 검증할 수 있으며 이러한 반복적 검증 과정을 자동화함으로써 개발 프로세스의 효율과 신뢰성을 향상시킬 수 있다.

3.2 테스트 자동화 개요

테스트 자동화는 모델로부터 자동 생성된 테스트 케이스를 기반으로 테스트 실행을 자동으로 수행하고 결과분석 보고서를 자동 생성하는 일련의 과정을 의미한다. 따라서, 테스트 자동화는 테스트 케이스 생성, 테스트 실행, 그리고 테스트 결과 분석 및 문서화라는 3 단계로 구성된다.

테스트 케이스 자동 생성 기법은 오래 전부터 많은 연구가 진행되어 상용화 단계에 이르렀으나 자동으로 생성된 방대하고 복잡한 테스트 케이스를 자동으로 실행하고 결과를 분석하는 기술과 같은 실제 자동차 산업에서 많이 필요로 하는 기술에 대한 연구나 지원 툴 개발은 상대적으로 미흡한 실정이다. 이에 본 연구에서는 테스트 자동 실행과 분석 보고서 자동 생성에 초점을 맞추었다.

3.3 모델기반 자동 테스트 실행 및 분석

테스트 자동 수행 환경을 구축하기 위한 개발 단계별 테스트 기법을 Table 1 에 정리하였다.

Table 1 Test phase and test technology

Test Phase	Test Technique
Requirement testing	Peer Review
Functional model testing	Model-in-the-loop
Implementation testing	Software-in-the-loop
Integration testing	Hardware-in-the-loop

기능 모델의 테스트 자동화는 대부분의 상용 툴에서 제공하는 MILS (model-in-the-loop simulation) 환경을 이용하여 손쉽게 구현할 수 있다. 그리고, ECU 구현 모델의 테스트는 자동 생성된 코드를 실제 ECU 에 다운로드하여 테스트를 수행할 수 있으며, ECU 상에서 SILS (software-in-the loop-simulation) 환경을 이용한다. 마지막으로 시스템 통합테스트는 HILS (Hardware in the loop simulation) 환경을 이용하여 네트워크 테스트 및 유닛 테스트를 실시한다.

MILS 에서 사용된 시뮬레이션 자동 실행 스크립트를 확장하여 Rapid Prototype ECU 나 Prototype ECU 상에서 테스트 자동 실행 환경을 개발할 수 있으며, Fig. 9 는 본 연구를 위하여 DsysD 사와 공동 개발한 SILS 기반 자동 테스트 실행을 위한 하드웨어와 소프트웨어이다.[8] 자동 생성된 테스트 시나리오를 시뮬레이션 소프트웨어에서 실행시키면서 입출력값을 CAN 또는 RS-232 직렬통신을 통하여 하드웨어에 전달하는 방식으로 자동 실행 및 분석 환경을 구축하였다.

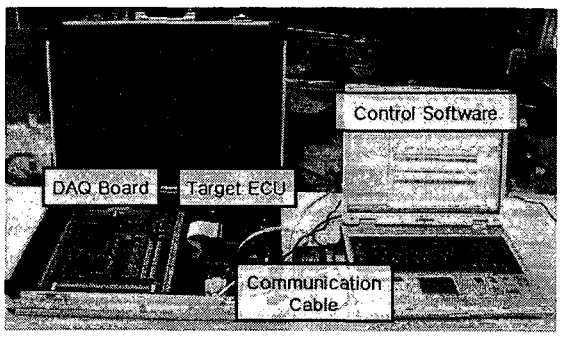

Fig. 9(a) Automated test execution system hardware

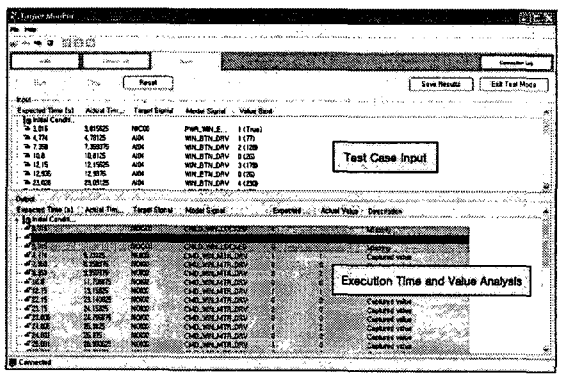

Fig. 9(b) Automated test execution system software

4. Windows Lift System 설계 및 검증 사례

4.1 Windows Lift System 의 개요

Windows Lift System 은 Fig. 10 과 같이 4 개의 도어 모듈, 즉 DDM (Driver Door Module), PDM (Passenger Door Module), RLDM (Rear Left Door Module), RRDM (Rear Right Door Module)과 1 개의 BCM (Body Control Module)로 구성되었으며, 각 모듈들은 CAN 과 LIN 을 통하여 연결되어 있고, BCM 이 CAN/LIN 간의 Gateway 역할을 수행한다. Windows Lift System 의 주요 기능은 네 창문의 전동 모터 제어, 어린이 보호 기능, 손낌 방지 기능, 조작 가능 조건 제어 등이다.

4.2 Windows Lift System 설계 및 검증

Fig.3 에 기술된 개발 프로세스를 중심으로 설계 및 검증에 대한 설계 사례를 소개하고자 한다. 먼저, 요구사항을 주요 기능그룹 단위로 모호한 표현이 없이 명확한 문장으로 작성한 후 다수의 관련자에 의하여 충분한 검토를 마친 후 모델 설계에 착수하였다.

4.2.1 Windows Lift System 기능 모델

Windows Lift System 의 기능모델은 Fig.11 과 같이 7 개의 기능그룹으로 구분하고 외부 입출력 변수를 정의한 후, 각 기능그룹별로 상세 기능을 상태도를 이용하여 설계하였다. 단위 기능 및 전체 시스템 기능을 검증하기 위하여 정적 오류가 있는지 체크한 후 시뮬레이션을 이용하여 동적 오류 검증을 수행하였으며, 이때 테스트 시나리오는 요구사항을 기반으로 수기로 작성하였다.

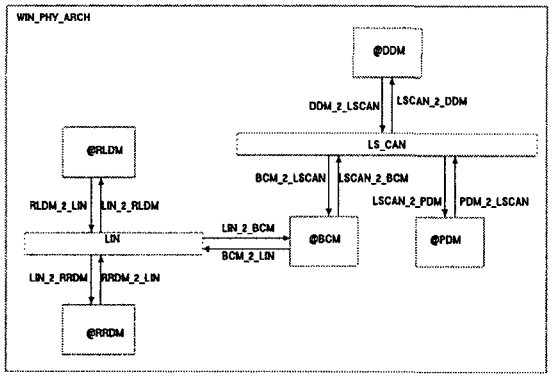

Fig. 10 Network architecture of windows lift system

기능 검증이 완료된 모델로부터 대부분의 상태(State)와 전이(Transition)를 커버하는 테스트 케이스를 자동 생성하고, 자동 생성된 테스트 케이스를 이용하여 시뮬레이션을 자동 수행한 결과를 분석하는 과정을 반복하면서 모델의 신뢰성을 향상시킬 수 있었다. 또한 오류 수정 과정 중 필연적으로 수반되는 단순 반복적이지만 정확성을 요하는 작업들을 자동화함으로써 테스트의 정확성과 일관성을 향상시키고 소요 시간 및 노력도 절감할 수 있었다. Fig.12 는 시뮬레이션 결과와 예상 결과가 상이한 경우를 검출하여 신속한 결과 분석이 가능하도록 자체 개발한 Test Analyzer 의 분석결과를 나타낸다.

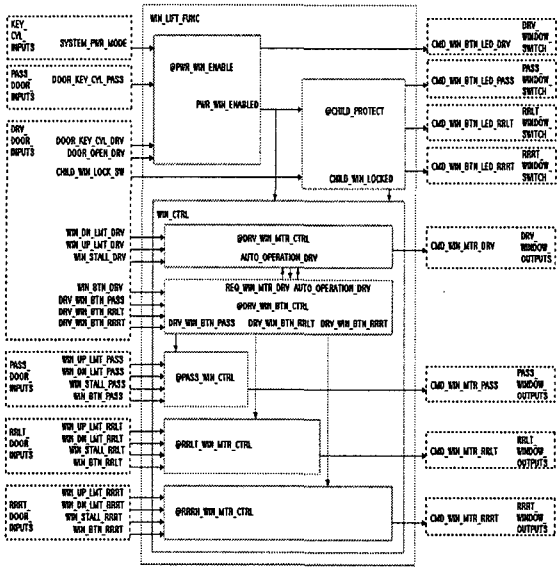

Fig. 11 Functional top chart for windows lift system

		OUTPUT-VARIABLES (PASS FAIL)					
Time	Step	DOOR_KEY_CYL_DRV	CMD_WIN_MTR_DRV	DRV_WIN_BTN_PASS	DRV_WIN_BTN_RRLT	DRV_WIN_BTN_RRRT	DOOR_OPEI DRV
0	4	0	0	0	0	0	0
1	8	2	0	2	2	2	1
2	11	2	0	4	4	4	1
3	15	1	0	4	4	4	1
4	20	1	1	Detected Error		4	1
5	23	1	1			4	1
6	27	1	1	4	4	4	1
7	30	1	1	4	4	4	1
8	33	1	1	4	4	4	1
9	37	1	1	4	4	4	1
10	40	1	1	4	4	4	1
11	43	1	1	4	4	4	1

Fig. 12 Test report generated by Test Analyzer

4.2.2 Windows Lift System 의 기능 할당

Windows Lift System 이라는 단순화된 전장 시스템의 네트워크 구조는 4.1 절의 Fig. 10 과 같이 가정하였다. 그리고, 기능모델과 네트워크 구조 간의 맵핑은 모델링 툴에서 제공하는 기능을 사용하였으며, 자동 생성된 ECU 모델을 이용하여 ECU 구현 모델을 설계하였다. Fig.13 은 자동 생성된 DDM 의 ECU 모델로서 미들웨어(Middleware) 계층을 거쳐 정보를 교환하는 구조를 가지며, 어린이 보호 기능과 윈도우 버튼 제어 등을 담당한다.

기능할당이 완료된 후에도 시스템의 기능이 정상적으로 수행되는지 검증한 후 네트워크 설계에 착수한다. 네트워크 설계는 메시지 정의 및 우선 순위 선정, 전송방식 및 주기 등을 결정하여 CAN 데이터베이스를 생성하는 작업이며, 2.2.5 절에서 설명된 바와 같이 ECU 모델과 연계하여 기능모델과 네트워크 모델의 통합 시뮬레이션 환경을 만들 수 있다.

4.2.3 Windows Lift System 의 ECU 구현 설계

ECU 구현 모델은 기능모델에 자동 코드 생성에 필요한 정보들을 추가한 모델을 의미하는데, 추가 정보에는 태스크 수행 방식 정의, 입출력 함수 정의 등이 포함된다.[3] ECU 구현 모델의 검증은 어플리케이션 코드를 자동 생성하여 3.3 절에서 소개된 SILS 기반 자동 테스트 장치 상에서 수행하였으며, 모델 설계 단계에 기능뿐만 아니라 성능까지 테스트할 수 있었다.

Fig.14 는 SILS 기반 자동 테스트 실행 및 검증 장치를 이용하여 실시한 ECU 구현 모델의 자동 테스트 수행 및 분석 결과를 나타낸다.

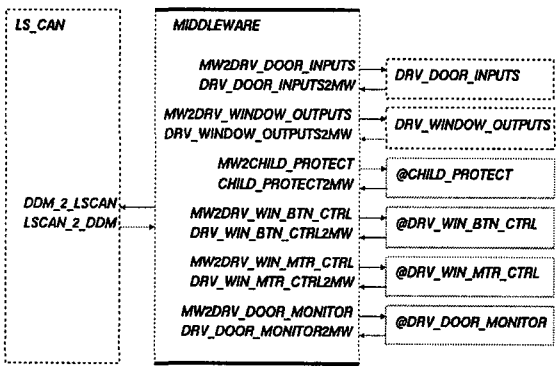

Fig. 13 ECU model created by automatic allocation tool

4.2.4 Windows Lift System 의 소프트웨어 통합

소프트웨어 통합을 위해서는 사전에 소프트웨어의 구조와 기능을 명확히 정의하여야 할 필요가 있는데, 본 사례 연구에서는 Fig.15 와 같은 구조를 갖는 Generic Software Architecture 를 이용하였다 .[3] 이러한 구조에서 어플리케이션 코드는 모델로부터 자동 생성하고, I/O 드라이버는 C 언어를 이용하여 직접 작성하였으며, 네트워크 관련 코드는 독일 Vector 사 [9] 의 자동 코드 생성 툴을 이용하였다.

소프트웨어 통합 과정이 완료된 실행코드는 프로토타입 ECU 를 이용하여 구축한 실부하 시뮬레이터 상에서 단위 ECU 들과 네트워크 시스템에 대한 최종 검증과정을 완료하였다.

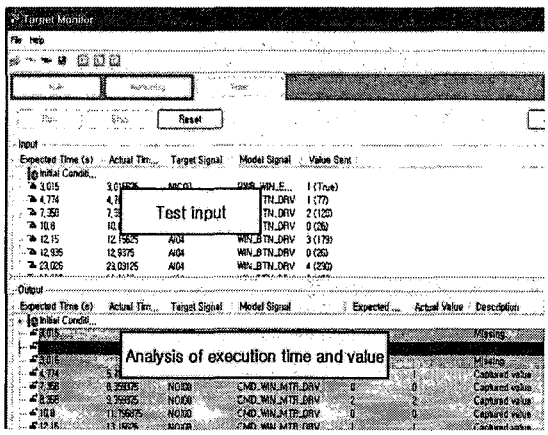

Fig. 14 Test Results from SILS based automatic test

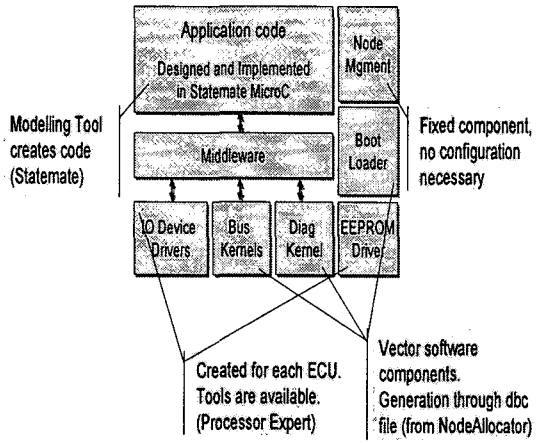

Fig. 15 Generic software architecture and generation techniques

5. 결론

최근 자동차 전장 시스템이 점점 더 복잡해지고 있음에도 불구하고, 품질 수준은 높이고 개발 기간과 비용은 줄여야 하는 어려운 당면 과제를 안고 있는 자동차 산업에서 모델기반 개발 프로세스를 도입하여 이러한 요구를 만족시키고자 노력하고 있다. 이에 본 논문에서는 네트워크 기반의 자동차 전장 시스템 설계 및 검증에 적합한 모델기반 개발 프로세스를 제안하고 사례연구를 통하여 그 유효성을 확인하였다.

복잡한 네트워크 설계를 단순화하기 위하여 기능 모델 설계와 네트워크 구조 설계를 독립적으로 수행한 후 Allocation 단계에 기능을 할당하는 접근방식을 활용하였으며, 설계가 진행되는 동안 수없이 반복하여야 하는 검증 절차를 자동화하여 검증의 효율성을 한층 향상시켰다.

이러한 효율적 개발 프로세스는 제품의 신뢰성을 향상시켜줄 뿐만 아니라 개발 기간 단축과 비용 절감이라는 자동차 산업의 궁극적인 기대효과를 달성할 수 있는 수단이 될 것으로 예상된다.

후 기

본 연구는 과학기술부의 "대구경북과학기술연구원" 기관고유연구 사업비지원으로 수행되었습니다.

참고문헌

1. Choi, B., "Current Status and Prospects of Convergence of Information Technologies into Cars," SERI Economic Focus, No. 65, pp.1-23, 2005.
2. Son, J., Wilson, I., Lee, W. and Lee, S., "Model Based Embedded System Development for In-Vehicle Networks Systems," SAE 2006 World Congress, 2006-01-0862, 2006.
3. Son, J., Wilson, I., Lee, W. and Lee, S., "Software Architecture for Model Based Automotive System Development and Its Application," The 13th International Pacific Conference on Automotive Engineering, pp. 520-524, Aug. 2005.
4. Hadeler, R. and Mathony, H., "Design of Intelligent Body Networks," SAE 2000 World Congress, 2000-01-0152, 2000
5. Hoffmann, H., "From Concept to Code – The Automotive Approach to using Statemate MAGNUM and Rhapsody MicroC," I-Logix Inc., 2000
6. Kum, D., Son, J. and Lee, S., "Automated Testing for an Automotive Embedded System," KSAE 2006 Spring Conference Proceedings, Vol. 3, pp. 1380-1385, 2006
7. Kaleita, D. L. and Hartmann, N., "Test Development Challenges for Evolving Automotive Electronic Technologies," SAE 2004 World Congress, 2004-21-0015, 2004.
8. DsysD Ltd. website. [Online]. Available: http://www.dsysd.com , 2006.
9. The Vector-Informatik GmbH website. [Online]. Available: http://www.vector-informatik.com, 2006.