AUTOSAR와 모델기반 기법을 적용한 차량 임베디드 시스템 소프트웨어의 개발 및 검증 기법

Automotive Embedded System Software Development and Validation with AUTOSAR and Model-based Approach

금 대 현, 손 장 경, 손 준 우*, 김 명 진
(Daehyun Kum, Jangkyung Son, Joonwoo Son, and Myungjin Kim)

Abstract : This paper presents a new approach to automotive embedded systems development and validation. Recently automotive embedded systems become even more complex and the product life cycle is getting reduced. To overcome these problems AUTOSAR, a standardized software platform and component based approach, was introduced. Model-based approach has been widely applied in the development of embedded systems and has strong benefits such as early validation and automated testing. In this paper cooperative development and validation of AUTOSAR and model-based approach are introduced and automated testing techniques are proposed. With the proposed techniques we can improve complexity management through increased reuse and exchangeability of software module and automated testing is realized.

Keywords : AUTOSAR, model-based, automated testing, virtual prototyping

I. 서론

오늘날의 자동차는 안전하고 편리함을 추구하는 소비자의 요구, 기술 혁신, 경쟁, 제품의 차별화, 규제 등의 다양한 요인에 따라 차량 내부의 전기/전자적인 구조가 점점 복잡해지고 있다. 또한 하이브리드 자동차, 지능형 자동차 등을 비롯한 미래형 자동차 분야와 관련한 연구 개발과 맞물려 차량 임베디드 시스템은 더욱 복잡해 질 것으로 예측하고 있다. 이런 차량의 전자화 경향에 따라, 주요 제어기능을 수행하는 차량 임베디드 소프트웨어의 중요성이 높아지고, 소프트웨어 오류가 차량 고장에 상당한 원인을 차지하고 있는 추세이며, 이에 따른 품질보증 비용 또한 늘어나고 있다. 그리고 치열한 경쟁에서 살아남기 위하여 제품 개발 주기가 짧아지고, 단가 인하 압력이 가속화 되고 있다. 그러나 전문적인 소프트웨어 설계자와 관리자가 부족한 자동차 분야에서 복잡한 소프트웨어를 빠르고 신뢰성 있게 개발하기에는 어려운 실정으로 새로운 소프트웨어를 개발 방법의 필요성이 대두되기 시작하였다.

해외 선진 자동차 업계에서는 이러한 문제점을 극복하기 위하여 표준화 및 컴포넌트 기반 기법을 도입하고 있다. AUTOSAR(Automotive Open System Architecture)는 표준화에 대한 대표적인 사례로 하드웨어와 소프트웨어의 분리를 통하여, 소프트웨어의 재사용성, 확장성 등을 향상시켜, 복잡한 소프트웨어를 빠르고, 신뢰성 있게 개발 가능하게 된다[1].

AUTOSAR는 2003년 6월 자동차의 전기/전자 아키텍처에 대한 공개 표준을 제정하는 것을 목표로 완성차와 부품업체를 중심으로 설립되었으며, 최근 AUTOSAR 소프트웨어 적용 사례에 대한 연구[2] 및 AUTOSAR 지원 툴 개발에 대한 연구가 활발히 진행 중이다[3].

AUTOSAR 표준이 발표되기 이전에도 GSA(Generic Software Architecture) 형태의 소프트웨어 아키텍처를 사용하고 있었으나 업체들 또는 차량 플랫폼 사이의 표준화가 되어 있지 않았기 때문에 동일한 시스템을 여러 차량에 대하여 중복으로 개발해야 하는 문제점들이 지적되어 오고 있었다[4]. AUTOSAR 표준을 적용함으로써, 이러한 문제점을 극복하고 확장성, 재사용성, 이식성의 장점을 가질 수 있다.

AUTOSAR는 응용 소프트웨어에 대한 개발이나 검증 방법에 대해서는 언급하고 있지 않으며, 응용 소프트웨어의 개발 및 검증의 효율성을 위하여 모델기반 기법을 적용할 필요가 있다[5-7]. 모델기반 기법은 개발 초기 단계에 하드웨어의 도움 없이 가상 프로토타이핑(virtual prototyping)과 래피드 프로토타이핑(rapid prototyping)을 가능하게 하며, 정형화 기법을 활용한 테스트 케이스 생성 및 시뮬레이션 자동화의 장점이 있다[8].

본 논문에서는 AUTOSAR와 모델기반 기법을 접목한 개발 및 검증 프로세스를 제안하고자 한다. 그리고 모델 및 요구사항으로부터 생성된 테스트 케이스를 공유 할 수 있는 테스트 데이터베이스(data base) 구축 및 AUTOSAR와 모델기반 기법을 활용한 가상 프로토타이핑과 래피드 프로토타이핑 자동화 방법을 제안한다. 그리고 Window lift system의 사례연구를 통해 제안한 방법의 타당성을 확인하였다.

II. AUTOSAR 와 모델기반 기법

1. AUTOSAR 소프트웨어 아키텍처

AUTOSAR 소프트웨어의 구조는 크게 AUTOSAR SWC (Software Component), RTE (Run Time Environment), BSW (Basic Software)의 3 계층으로 나누어지며, 기본 설계 개념은 RTE의 개념을 도입하여 응용 소프트웨어인 SWC와 하드웨어 관련 소프트웨어인 BSW를 분리함으로써, 응용 소프트웨어를 하

* 책임저자(Corresponding Author)
논문접수 : 2007. 9. 29., 채택확정 : 2007. 10. 26.
금대현, 손장경, 손준우, 김명진 : 대구경북과학기술연구원 지능형자동차연구팀
(kumdh@dgist.ac.kr/sonjk@dgist.ac.kr/json@dgist.ac.kr/mjkim@dgist.ac.kr)
※ 본 연구는 과학기술부 기관고유사업의 연구비 지원으로 수행되었음.

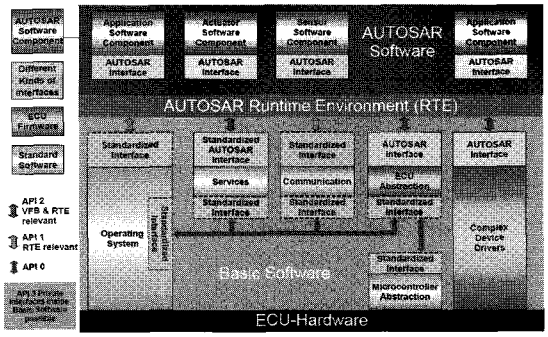


그림 1. AUTOSAR 소프트웨어 아키텍처.
Fig. 1. AUTOSAR software architecture.

드웨어에 독립적으로 개발할 수 있다. 그림 1은 AUTOSAR 소프트웨어 아키텍처를 나타낸다.

각 AUTOSAR SWC는 응용 소프트웨어 기능의 일부를 구현하며, ECU에 맵핑되는 기본 단위이다. AUTOSAR SWC는 AUTOSAR interface를 통하여 상호 데이터를 교환을 한다. sensor/actuator SWC는 AUTOSAR SWC의 한 종류로써 ECU의 센서 및 액추에이터의 구현을 위한 SWC이다. RTE는 각 SWC 사이 및 SWC와 BSW 사이의 정보 교환을 위한 중추적인 역할을 하며 소프트웨어와 하드웨어를 분리시키는 AUTOSAR의 핵심이다. BSW는 표준화된 레이어 계층으로 service layer, EAL(ECU abstraction layer), MCAL(Micro-controller Abstraction Layer) 그리고 CDD(Complex Device Driver)로 구성된다. Service는 OS(Operating System), 네트워크 서비스, 메모리 서비스, diagnostic, ECU state management 등의 기능을 수행한다. EAL은 ECU 내부의 장치들과의 인터페이스를 제공하며, 상위계층의 설계를 ECU에 독립적으로 할 수 있게 해준다. MCAL은 상위 계층에서 마이크로 콘트롤러의 레지스터를 직접 조작하는 것을 피하게 해주며, digital I/O, ADC(Analog Digital Converter, PWM, SPI 등으로 구성된다.

2. AUTOSAR 개발 방법론

AUTOSAR 소프트웨어 개발은 시스템 단계와 ECU 단계로 나누어진다. 그림 2는 AUTOSAR 방법론에 대한 개략도를 보여준다. 시스템 단계에서는 SWC의 데이터 타입, 인터페이스와 연결 상태 등을 기술하는 SWC description, 각 ECU의 하드웨어 구성을 기술하는 ECU resource description, 그리고 버스 시그널, 토폴로지 등의 system constraint description을 작성한다. 각 SWC 내부에는 응용 소프트웨어 구현을 위한 runnable정의 및 트리거 조건을 정의한다. 다음은 SWC를 각 ECU에 맵핑하고 네트워크 설계를 하여 system configuration description을 기술한다. 작성된 파일은 XML 형식의 템플릿을 사용하고 있으며, 데이터의 공유 및 전달을 표준화 할 수 있다.

다음 단계는 system configuration description으로부터 각 ECU 정보를 추출하여 ECU 설계를 하며, 태스크 정의 및 할당, RTE 생성, BSW 등을 설계하여 ECU configuration description을 기술한다. 다음은 응용 소프트웨어와 함께 RTE, OS, communication 등의 AUTOSAR Software 모듈의 코드를 생성하고, 컴파일, 링크를 거쳐 실행 파일을 만들어서 ECU를 구현한다.

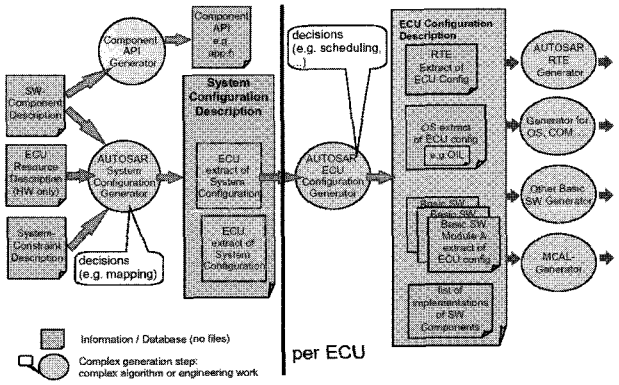


그림 2. AUTOSAR 방법론 개략도.
Fig. 2. Overview of the AUTOSAR methodology.

3. AUTOSAR와 모델기반 기법의 접목

모델 기반 기법을 도입하여 AUTOSAR의 장점을 더욱 부각 시킬 수 있다. 표준화된 AUTOSAR 소프트웨어 아키텍처와 XML을 이용한 정보교환 및 하드웨어와 소프트웨어의 설계를 분리하는 개발방법론은 재사용성, 확장성, 플랫폼간의 이식성을 최대화 시켜주며, 모델기반 기법은 시스템의 개발 초기 단계에 하드웨어의 도움 없이 검증을 용이하게 한다.

모델기반 기법은 시스템의 요구사항으로부터 상태 다이어그램 (state diagram), 활동 다이어그램 (activity diagram), 블록 다이어그램 등을 활용하여 모델을 설계하며, 모델로부터 코드를 자동으로 생성 가능하다. 또한 가상 프로토타이핑 및 래피드 프로토타이핑 환경을 제공해 주며, 테스트 케이스 생성 및 실행의 자동화로 오류 검출 율을 향상시켜 품질 향상 및 개발 기간을 단축 시킬 수 있다[9].

그림 3은 모델을 활용한 AUTOSAR SWC 코드 생성, AUTOSAR 시스템 아키텍처 생성 및 테스트 데이터베이스 생성 개략도를 나타낸다. 모델로부터 AUTOSAR 시스템 아키텍처에 관한 XML 파일을 추출하고, runnable 코드와 RTE-API를 생성하여 AUTOSAR 시스템 단계의 SWC 구현 코드로 사용한다. 그리고 모델로부터 자동으로 추출된 테스트 케이스는 테스트 데이터베이스로 입력되어, AUTOSAR 시스템 단계의 가상 프로토타이핑 및 래피드 프로토타이핑의 테스트 자동화 모듈을 생성하여 테스트 효율을 획기적으로 높일 수 있다.

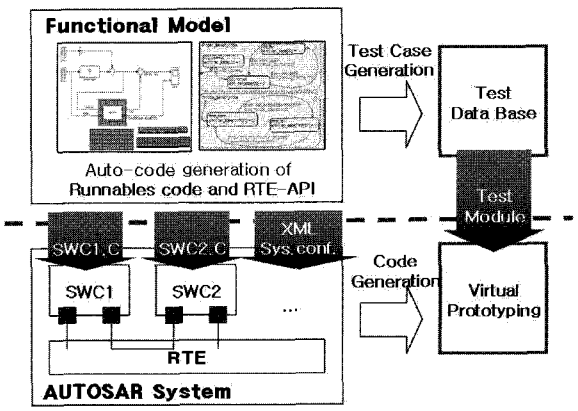


그림 3. 모델을 활용한 SWC 코드 및 테스트 DB 생성.
Fig. 3. SWC code and the test DB from a model.

III. 차량 임베디드 시스템의 개발 및 검증

1. 개발 프로세스

개발 프로세스는 크게 로직 단계, 시스템 단계, ECU 단계로 나눌 수 있으며, 로직 단계에서는 모델기반 기법에 따라 시스템 아키텍처 설계 및 응용 소프트웨어를 개발하며, 시스템 단계와 ECU 단계는 AUTOSAR 방법론과 소프트웨어 아키텍처를 적용한다. 그림 4는 AUTOSAR와 모델기반 기법을 적용한 V-프로세스를 나타낸다. 로직 단계로부터 AUTOSAR 시스템 아키텍처 정보의 추출 및 시스템 단계로부터 ECU 정보 추출은 AUTOSAR의 표준화된 XML 템플릿을 이용한다.

로직 단계: 임베디드 시스템의 개발은 요구사항으로부터 시작한다. 시스템의 요구사항에 따라 Silmulink/Stateflow, statemate 등과 같은 모델링 툴을 활용하여 기능적 아키텍처를 설계하고, SWC 내부 기능의 상세 구현을 위하여 상태 다이어그램, 블록 다이어그램 등을 이용하여 모델링 한다. 모델의 설계 단계부터 AUTOSAR의 SWC, runnable entity, port, interface, data type, runnable trigger 등 AUTOSAR 아키텍처를 고려하여 설계해야 하며, 만약 기존의 모델을 재사용 한다면 AUTOSAR에 맞게 모델을 수정해야 한다.

모델로부터 정보를 추출하여 기능적 아키텍처와 동일한 AUTOSAR 시스템 아키텍처를 생성한다. SWC 코드는 runnable과 RTE-API로 구성되며, RTE-API는 sender/receiver port, server/client port, runnable, RTE event, exclusive area 등이 포함된다. RTE-API는 매크로, 모델내부 변수의 API 설정 또는 툴박스의 개발 등의 방법으로 생성할 수 있다[10].

모델로부터 SWC runnable의 C 코드 및 RTE API를 생성하여 시스템 단계에 SWC.C 파일을 삽입하고, 모델의 기능적 아키텍처로부터 XML 파일을 생성하여 AUTOSAR 시스템 단계로 전달한다[11].

시스템 단계: 로직 단계로부터 생성된 시스템 아키텍처 정보와 SWC 코드를 입력 받아서 AUTOSAR 시스템 단계의 설계를 시작한다. 응용 소프트웨어는 여러 SWC의 조합으로 구성되며, 각 SWC는 응용 소프트웨어 기능의 일부를 담당한다. 그리고 SWC는 하나 이상의 runnable로 구성되며, runnable은 실행의 최소 단위이며 태스크 할당의 단위이다.

시스템 단계에서는 ECU 위치 및 개수, 네트워크 버스, 센서 및 액추에이터 등의 개략적인 하드웨어 설계를 하고, sensor/actuator SWC를 실제 ECU 하드웨어 신호에 맞게 구현한다. SWC를 ECU에 맵핑하고, ECU 사이의 데이터 교환에 대해서 네트워크 설계를 하고, ECU 내부 SWC 사이의 데이터 교환은 RTE를 통해서 이루어 진다.

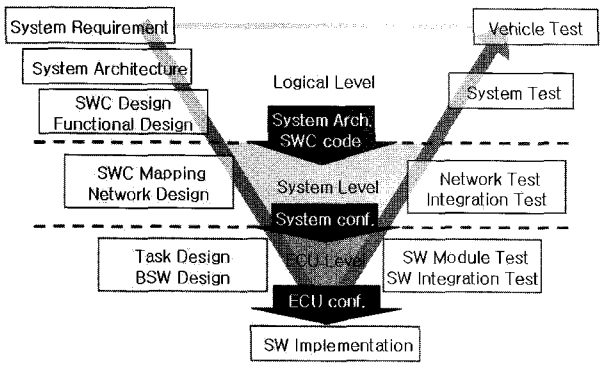


그림 4. AUTOSAR와 모델기반 기법을 적용한 V-프로세스.
Fig. 4. V-process of AUTOSAR and model-based approach.

그림 2에서 보는 바와 같이 시스템 단계의 결과물은 SWC description, ECU resource description, system constraint description 그리고 SWC 맵핑 정보를 포함하는 system configuration description 이다. XML 형태로 저장된 정보는 ECU 단계로 전달된다.

ECU 단계: 시스템 configuration으로부터 각 ECU 정보를 추출한다. 이 단계에서는 ECU configuration을 통해 하드웨어에서 동작 가능한 코드를 생성하고, 상세한 하드웨어 설계를 한다. ECU 단계는 기능 별로 분류하면 시스템 서비스, 메모리, 통신, I/O 하드웨어로 구성되며, 태스크, 알람, 이벤트, 인터럽터, 통신, SPI 핸들러, DIO, ECU manager, NVM, ADC, ICU, PWM, watch-dog 등 어플리케이션에서 사용하는 BSW를 구성한다. BSW는 표준화된 API를 통해서 설계하며, ECU configuration description 정보를 바탕으로 실행 가능한 코드를 생성하여 ECU에 구현한다.

2. 검증 프로세스

테스트는 테스트 케이스 생성, 테스트 실행, 결과 분석의 3단계로 나눌 수 있으며, 개발 단계에 따라서 로직 단계는 MILS(Model In the Loop Simulation), 시스템 단계는 SILS (Software In the Loop Simulation), ECU 단계는 HILS(Hardware In the Loop Simulation) 기법을 사용한다[12].

테스트 데이터베이스를 활용하여 모든 검증 단계의 테스트 케이스를 공유하고, 자동화 환경을 생성한다. 테스트 케이스는 기능 모델 및 시스템 요구사항으로부터 생성한다.

차량 임베디드 시스템이 복잡해지고 소프트웨어의 크기가 기하급수적으로 늘어남에 따라서 테스트 케이스의 생성 및 수행을 엔지니어가 직접 하나씩 하는 것은 매우 비효율적이고 또 다른 에러를 발생 시킬 가능성이 높다. 그러므로 테스트의 자동화는 필수라고 할 수 있다.

테스트 케이스: 테스트 데이터베이스를 활용하여 테스트 케이스의 공유화 및 테스트 수행의 자동화를 실현할 수 있다. 테스트는 각 개발 단계 마다 지속적으로 수행되며, 설계 변경 또는 오류 수정이 발생한 경우 반복 수행해야 한다. 그러므로 많은 테스트 케이스를 각 개발 단계 마다 여러 번 반복하게 된다.

테스트 데이터베이스를 활용하여 테스트의 효율을 높이기 위하여 로직 단계에서 모델 및 시스템 요구사항으로부터 생성된 테스트 케이스를 생성하고, 각 검증 단계마다 공유한다. 그리고, MILS, SILS, HILS를 자동으로 수행할 수 있는 테스트 모듈을 생성한다.

테스트 자동화를 위하여 테스트 케이스는 가장 간단하고, 측정 가능하게 만들어야 한다. 복잡한 테스트 케이스는 테스트 자동화를 방해할 뿐만 아니라, 테스트 과정에서 오류를 발생 시킬 수 있다. 테스트 케이스는 정형화 기법을 도입함으로써 상태 다이어그램, 시퀀스 다이어그램 등의 모델로부터 자동으로 추출 가능하며, 또한 시스템의 요구사항으로부터 직접 테스트 케이스를 생성 해야 한다. 요구사항 기반 테스트 케이스를 활용하여 시스템의 설계가 요구사항을 충분히 반영하여 설계가 되었는지를 확인하는 기본적인 테스트

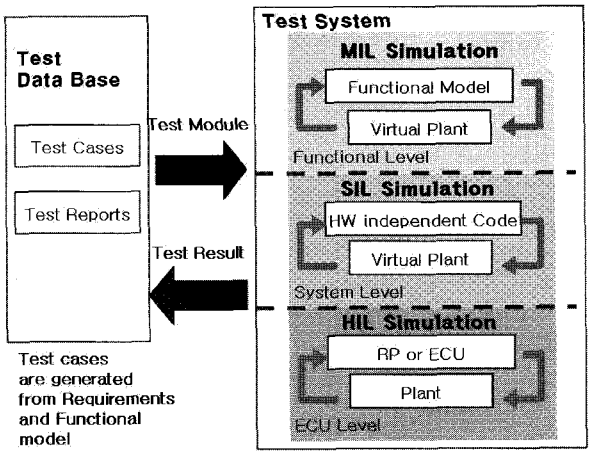


그림 5. 테스트 DB의 개념도.
Fig. 5. Concept of the test DB.

를 수행하며, 모델로부터 정형화 기법을 통해 추출한 테스트 케이스는 테스트 커버리지를 최대한 높여준다. 그림 5는 테스트 데이터베이스의 개념도를 보여준다.

MILS: 로직 단계의 모델이 요구사항에 적합하게 설계되었는지를 검증하기 위하여 MILS 시뮬레이션을 한다. 이 단계에서는 AUTOSAR 소프트웨어 및 하드웨어에 독립적인 모델로써 오로지 기능상의 오류만을 검증 한다.

대부분의 모델링 툴은 MIL 시뮬레이션을 할 수 있는 환경을 제공해주며, Matlab/Simulink의 T-VEC tester, statemate의 ATG 등과 같은 툴은 정형화 기법을 이용하여 모델로부터 테스트 케이스를 자동으로 추출해준다. 그리고 시뮬레이션 환경을 위한 센서, 액추에이터, 차량, 운전자 등의 플랜트 모델과 함께 시뮬레이션 한다.

SILS: 하드웨어에 관련된 코드를 제외한 SWC, RTE, OS, communication 등의 코드를 생성하여 SIL 시뮬레이션을 한다. PC상의 가상 프로토타이핑 또는 래피드 프로토타이핑 환경을 사용할 수 있으며, CAN, LIN, FlexRay 등의 가상 또는 실제 차량 네트워크 환경에서 시뮬레이션 한다. 테스트 데이터베이스로부터 SIL 시뮬레이션에 필요한 테스트 모듈을 자동 생성 시켜서 테스트 수행을 자동화 하고, 테스트 효율을 높인다.

HILS: I/O 드라이버, ECU 서비스, 메모리 등의 하드웨어 관련된 BSW를 추가 생성하여 ECU 또는 래피드 프로토타이핑 환경을 활용하고, 실제 센서 및 액추에이터와 연계하여 HIL 시뮬레이션을 한다. HIL 시뮬레이션 또한 테스트 데이터베이스로부터 테스트 모듈을 생성 한다.

IV. Case study-Window lift system

1. Window lift system

본 논문에서 제안한 AUTOSAR에 모델기반 기법을 접목한 개발 및 검증 프로세스, 테스트 데이터베이스를 활용한 MILS 및 SILS 테스트 자동화의 구현을 Window lift system에 적용하여 개발하였다. Window lift system은 BCM(Body Control Module), DDM(Driver Door Module), PDM(Passenger Door Module), RLDM(Rear Left Door Module), RRDM(Rear Right Door Module)의 5개 ECU로 구성되어 있으며, CAN 네트워크 통신

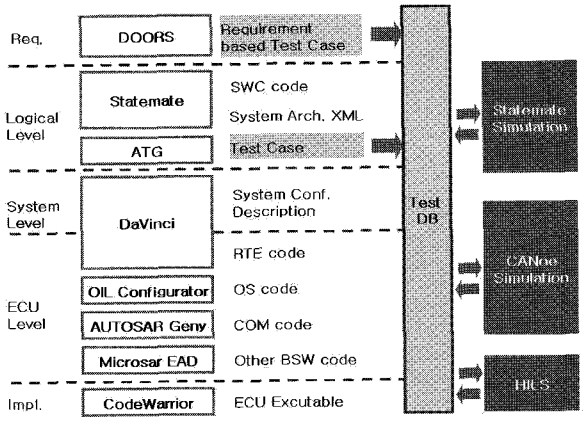


그림 6. AUTOSAR 개발 및 검증을 위한 툴 체인.
Fig. 6. Tool chain for AUTOSAR development and validation.

을 사용한다.

AUTOSAR를 적용한 개발 및 검증을 위한 툴 체인은 그림 6과 같다. Telelogic사의 DOORS를 사용하여 요구사항의 분석 및 관리를 하였으며, 기능 모델 설계는 Telelogic사의 Statemate를, 테스트 케이스 자동 생성을 위하여 Statemate의 플러그인 툴인 ATG를 사용하였다[13]. AUTOSAR 시스템 설계 및 RTE, OS, COM, 기타 BSW 코드 생성을 위하여 각각 Vector사의 DaVinci, OIL configurator, AUTOSAR geny, Microsar EAD를 사용하였다. 네트워크 시뮬레이션을 위하여 Vector사의 CANoe를 사용하였으며, 테스트 케이스의 공유 및 자동화를 위하여 테스트 데이터베이스를 활용하였다[14]. 생성된 코드의 컴파일, 링크, 빌드는 Metrowerks사의 Codewarrior를 사용하였다.

2. 로직 단계 개발 및 검증

그림 7은 Window Lift System의 기능적 아키텍처로부터 AUTOSAR 시스템 아키텍처를 위한 XML 파일을 추출하는 것을 보여준다. Statemate모델의 XML 변환은 직접 구현하였

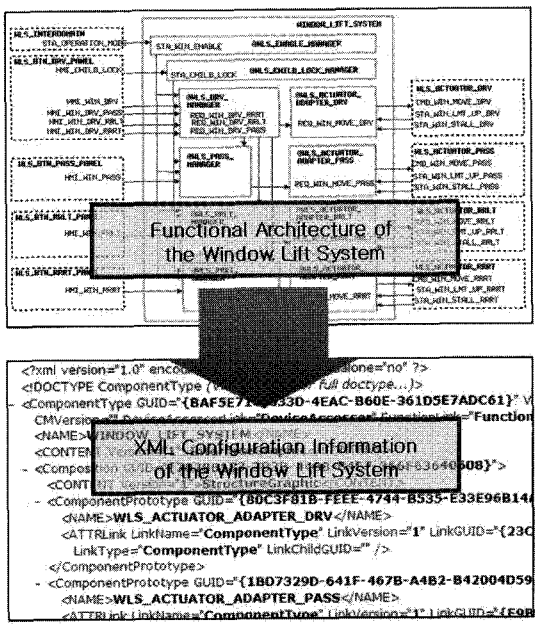


그림 7. 모델로부터 생성한 시스템 아키텍처의 XML.
Fig. 7. XML of system architecture from the model.

```
/*****************************************************************
 * Runnable Entities
 *****************************************************************

void WLS_DRV_MANAGER_Init(void)
{
  /* insert C-Code here for runnable entity */
  // Initializing CAN communication for Tx
  Rte_Call_ComM_RequestDataElement(RTE_DATAREQ_REQ_WIN_DRV_PASS_Btn);
  Rte_Call_ComM_RequestDataElement(RTE_DATAREQ_REQ_WIN_DRV_RRLT_Btn);
  Rte_Call_ComM_RequestDataElement(RTE_DATAREQ_REQ_WIN_DRV_RRRT_Btn);

}


/*****************************************************************
 * Runnable Entity Name: WLS_DRV_MANAGER_RE
 *****************************************************************

void WLS_DRV_MANAGER_RE(void)
{
  /* insert C-Code here for runnable entity */

  // Read from RTE
     Rte_Read_HMI_WIN_DRV_Btn(&HMI_WIN_DRV);
     Rte_Read_STA_WIN_ENABLE_Enable(&STA_WIN_ENABLE);

  // Runnable Entity Function
     cgActivity_DRV_MANAGER();


  // Write to RTE
     Rte_Write_REQ_WIN_MOVE_DRV_Move(REQ_WIN_MOVE_DRV);
}
```

그림 8. Functional 모델로부터 생성한 SWC 코드.
Fig. 8. A SWC code from the functional model.

다. 중앙의 10개의 activity chart는 AUTOSAR 시스템 단계에서 Atomic SWC에 해당하며, 양측 9개의 점선으로 된 external activity chart는 sensor/actuator SWC에 해당한다. 그리고 activity chart 사이의 데이터 흐름은 AUTOSAR port와 interface에 해당한다.

Runnable의 상세 구현을 위하여 state chart로 모델링 하였으며, RTE-API를 생성하기 위하여 모델의 변수에 API를 지정하였다. 그림 8은 statemate로부터 생성한 SWC 코드이며, runnable 코드와 RTE-API를 포함하고 있다.

MIL 시뮬레이션을 위해서는 테스트 케이스가 필요하다. 테스트는 요구사항에 적합하게 설계 되었는지 검증하기 위하여 요구사항 기반 테스트를 수행한 후, 테스트의 커버리지를 향상 시키기 위하여 statemate 플러그인 툴인 ATG로부터 정형화 기법으로 테스트 케이스를 자동으로 생성하여 테스트 하였다.

ATG로부터 추출된 테스트 케이스 및 요구사항 기반 테스트 케이스를 테스트 데이터베이스로 불러들이고, statemate 시뮬레이션을 자동으로 수행할 수 있는 테스트 모듈을 테스트 데이터베이스로부터 생성하여 자동으로 시뮬레이션을 수행하고 결과 분석을 하였다. 시뮬레이션을 정확하게 수행하기 위해서 Window lift system의 모터를 묘사할 수 있는 플랜트 모델을 포함하여 시뮬레이션 하였다.

3. 시스템 단계 개발 및 검증

모델로부터 생성한 XML 파일을 DaVinci에서 읽어 들어서, SWC, port, interface, data type, runnable로 구성된 시스템 아키텍처를 생성한다. 그리고 모델에서 설계하고 검증된 SWC 코드를 삽입한다. 그림 9는 Window lift system의 AUTOSAR 시스템 아키텍처를 나타낸다.

DaVinci에서 Window lift system의 BCM, DDM, PDM, RLDM, RRDM의 5개의 ECU와, CAN 네트워크 버스를 정의하고, 센서 및 액추에이터를 설정하여서 개략적인 하드웨어 설계를 하였다.

Sensor/Actuator SWC는 ECU의 하드웨어를 고려하여 구현하였다. SWC들을 5개의 ECU에 맵핑하고, 맵핑한 이후 생성

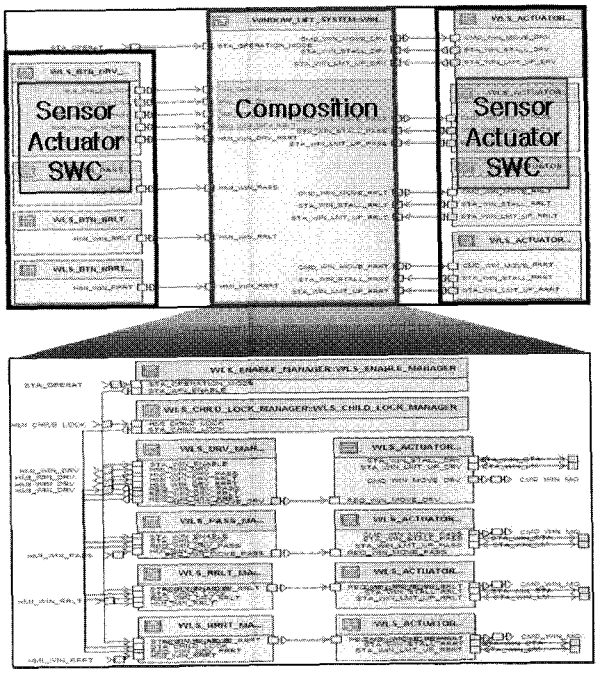


그림 9. Window lift system의 AUTOSAR 시스템 아키텍처.
Fig. 9. AUTOSAR system architecture of the window lift system.

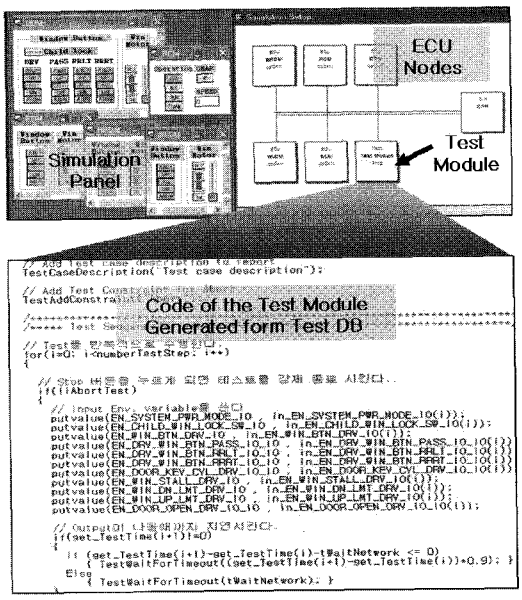


그림 10. CANoe 시뮬레이션 및 테스트 모듈.
Fig. 10. CANoe simulation and test module.

된 네트워크 신호 정의, 메시지 생성, 송신 타입 등 네트워크 설계를 하였다.

시스템 단계의 검증을 위하여 DaVinci에서 생성한 코드와 플랜트 모델의 코드를 DLL 파일로 생성하여, Vector사의 CANoe에 삽입하여 SIL 네트워크 시뮬레이션을 하였다. 이 단계에서 생성한 코드는 하드웨어에 독립적인 코드로써, SWC, RTE, communication, OS 관련 코드를 일부 포함한다. 테스트 DB로부터 CANoe 시뮬레이션을 위한 테스트 모듈을 생성하여 자동으로 시뮬레이션 하였다. CANoe의 테스트 모듈은 CAPL(CANoe Application Programming Language) 형태로

생성된다. 그림 10은 Window lift system의 CANoe네트워크 시뮬레이션 환경 및 테스트 모듈을 나타낸다.

4. ECU 단계 개발

EUC 단계에서는 상세한 하드웨어 설계를 통해 하드웨어에서 동작이 가능한 코드를 생성한다. Window lift system에서 사용하는 BSW는 communication manager, NVM, SPI handler, Watch-Dog을 사용하였다. Watch-dog 설정, MCU 하드웨어 초기 설정, 메모리 설정, SPI 채널 및 핸들러 설정, port 초기화, DIO 설정 등의 BSW configuration 정보는 XML 파일로 저장되고, XML 파일 정보를 바탕으로 BSW 코드를 생성하였다.

실제 ECU환경에서 각 ECU의 테스트 및 HILS환경에서 시스템 integration 테스트를 수행하였다.

5. 결과

AUTOSAR의 sensor/actuator SWC와 RTE를 적용함으로써 임베디드 소프트웨어의 하드웨어와 관련된 부분을 분리시킬 수 있었으며, 하드웨어의 변경 및 수정 발생 시 소프트웨어의 수정 및 테스트를 간단하게 할 수 있었다. 또한 응용 소프트웨어를 SWC 단위로 모듈화함으로써 기능의 확장 및 변경을 쉽게 적용할 수 있었다. 모듈화 및 표준화된 BSW를 활용하여 하드웨어 기능을 쉽게 구현할 수 있었다. 하지만 방대한 AUTOSAR 표준을 이해하고 제대로 적용하기는 쉽지 않은 문제점도 있었다.

모델 기반 기법 및 테스트 데이터베이스를 적용함으로써 응용 소프트웨어에 대한 테스트 기간 단축, 테스트 효율 및 신뢰성을 향상 시킬 수 있었다. Window lift system의 테스트 케이스는 약 2000 경우 이며, 각 경우에 대해서 하나씩 테스트를 할 경우 하루 정도의 시간이 소요되지만 자동화를 함으로써 1시간 이내에 수행할 수 있었다. 테스트는 소프트웨어의 오류 수정이나 변경으로 인해 수 차례 반복 수행되었지만, 자동화 및 테스트 케이스의 공유로 상당한 시간을 단축할 수 있었다. 그리고 개발 초기 단계에 가상 프로토타이핑 환경을 활용하여 대부분의 오류를 검출할 수 있었으며, 하드웨어가 개발된 이후 실제 환경에서 테스트한 결과 기능상의 오류는 거의 발견되지 않았으며, 하드웨어와 연계한 성능상의 오류 몇 가지만이 발견되었다. 그리고 테스트 케이스 생성을 자동화하여 테스트 커버리지를 높임으로써 소프트웨어의 신뢰도를 향상시킬 수 있었다.

V. 결론 및 고찰

본 연구에서는 AUTOSAR에 모델기반 기법을 접목한 개발 프로세스 및 검증 기법을 제안하였으며, Window Lift System 개발을 통하여 제안한 방법의 타당성을 확인하였다.

AUTOSAR 개발 단계는 로직단계, 시스템 단계, ECU 단계로 세분화였으며, 로직 단계는 응용 소프트웨어의 설계뿐만 아니라, AUTOSAR시스템 아키텍처 XML 및 SWC 코드를 생성하여서 AUTOSAR 툴과 연계 할 수 있었다. 특히 응용 소프트웨어 설계 시 모델기반 기법을 적용하여 초기 단계의 오류 검출 율을 높여 소프트웨어의 신뢰성을 향상 시켰다. 그리고 각 단계별 테스트 데이터베이스를 이용한 검증 자동화로 테스트 효율을 개선 할 수 있었다.

RTE를 이용하여 응용 소프트웨어와 하드웨어 분리하여 개발 할 수 있음을 확인하였지만 재사용성, 확장성 등의 AUTOSAR 장점을 부각하기 위해서는 다양한 응용 시스템의 표준화 작업과, 여러 하드웨어에 호환 가능한 BSW 개발도 함께 진행되어야 할 것으로 판단된다. 그리고 현재 시작단계로 여러 가지 결함이 내제되어있는 소프트웨어 컴포넌트 및 설계의 안정화 역시 병행하여야 할 것이다.

참고문헌

[1] www.AUTOSAR.org.

[2] K. Nishikawa and K. Kajio, "TOYOTA Electronic Architecture and AUTOSAR Pilot," *SAE 2007 World Congress*, 2007-01-1614, April 2007.

[3] D. Stichling, O. Niggenmann, R. Otterbach and K. Hoffmeister, "Effective Cooperation of System Level and ECU Centric Tools within the AUTOSAR Tool Chain," *SAE 2007 World Congress*, 2007-01-1279, April 2007.

[4] J. Son, I Wilson, W. Lee and S. Lee, "Software Architecture for Model Based Automotive System Development and Its Application," *Proc. of the 13th International Pacific Conference on Automotive Engineering*, PP520-525, Aug. 2005.

[5] J. Son, I. Wilson, W. Lee and S. Lee, "Model Based Embedded System Development for In-Vehicle Network Systems," *SAE 2006 World Congress*, 2006-01-0862, April 2006.

[6] Y. Dong, M. Li and R. Josey, "Model Based Software Development for Automotive Electronic Control Units," *SAE 2004*, 2004-21-0038, Oct. 2004.

[7] M. Muts, M. Huhn and U. Goltz, "Model Based System Development in Automotive," *SAE 2003 World Congress*, 2003-01-1017, March 2003.

[8] R. Probert, H. Ural and A. Williams, "Rapid Generation of Functional Tests using MSCs, SDL and TTCN," *Computer Communications*, Vol 24, No 3, pp. 374-393, Feb. 2001.

[9] D. Kum, J. Son, S. Lee, I. Wilson and W. Lee, "Model-Based Automated Validation Techniques for Automotive Embedded Systems," *SAE 2007 World Congress*, 2007-01-0503, April 2007.

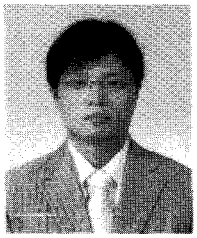
[10] A. Mjeda, G Leen and E. Walsh, "The AUTOSAR Standard-The Experience of Applying Simulink According to its Requirements," *SAE 2007 World Congress*, 2007-01-0509, April 2007.

[11] O. Niggemann U. Eisemann, M. Beine and U. Kiffmeier, "Behavior Modeling Tools in an Architecture-Driven Development Process – From Function Models to AUTOSAR," *SAE 2007 World Congress*, 2007-01-0507, April 2007.

[12] D. Kaleita and N. Hartmann, "Test Development Challenges for volving Automotive Electronic Technologies," *SAE 2004*, 2004-21-0015, Oct. 2004.

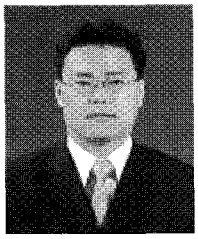
[13] www.telelogic.com

[14] www.vector-worldwide.com



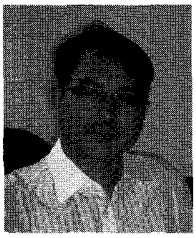
금 대 현

2001년 계명대학교 자동차공학과 졸업. 2003 계명대학교 자동차공학과 대학원 졸업(석사). 2003년~2005년 LG전자 연구원. 2005년~현재 대구경북과학기술연구원 연구원. 관심분야는 차량 임베디드 시스템, AUTOSAR.



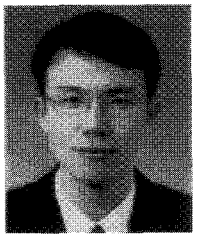
손 장 경

2003년 울산대학교 전기공학과 졸업. 2005년 울산대학교 전기전자정보시스템 공학과 졸업(석사). 2005년~현재 대구경북과학기술연구원 연구원. 관심분야는 차량 임베디드 시스템, AUTOSAR



손 준 우

1994년 부산대학교 생산기계공학과 졸업. 1996년 부산대학교 대학원 생산기계공학과 졸업(석사). 2006년 부산대학교 대학원 지능기계공학과 졸업(박사). 1996년~2005년 대우정밀㈜ 기술연구소 근무. 2005년~현재 대구경북과학기술연구원 근무중. 관심분야는 모델기반 자동차 임베디드 시스템 설계 및 검증 기술, 차량 네트워크 설계 기술, 고령운전자를 위한 Human Factor 연구, 정신적 부하를 고려한 Advanced HMI 기술.



김 명 진

1998년 포항공과대학원 박사 졸업. 2001년 삼성전자 자동화 연구소 근무. 2007년 NCB Networks 근무. 현재 대구경북과학기술연구원 근무. 관심분야는 자동차 지능화 시스템, 고신뢰성 영상처리 embedded s/w, 고속 영상 처리, 신뢰성 향상을 위한 program기법.