논문 2009-04-24

AUTOSAR 임베디드 소프트웨어의 모델기반 개발 및 테스트 방법 - 사례연구 : 운전자 위치제어 시스템

(Model-Based Development and Test Method for The AUTOSAR Embedded Software)

박 광 민, 금 대 현, 이 성 훈*
(Gwangmin Park, Daehyun Kum, Seonghun Lee)

Abstract : Automotive systems have tended to be equipped with many electronic contents to satisfy safety, comport, convenience, and entertainment services over the past years. As a result, the amount of vehicle embedded software in electrical/electronic(E/E) systems is steadily increasing to manage these requirements. This leads to the traditional, document-based software development in the vehicle embedded systems being increasingly displaced by a model-based development in order to reduce software development time and cost. Due to the application of model-based development, a great evolution is being realized in the aspect of efficiency, but the development is being made without sufficient testing. So, erroneous automotive embedded software may cause serious problems such as car accidents which relate to human safety. Therefore, efficient methods for model-based test and validation are needed to improve software reliability in the stage of embedded software development. This paper presents the model-based development and test method for AUTOSAR embedded software to improve its reliability and safety, and it is demonstrated based on the case study.

Keywords : AUTOSAR, Automotive software, SWC(Software Component), Model-based test, DPS

I. 서 론

오늘날 자동차는 단순한 이동수단으로부터 차량 탑승자의 안전을 제공하며 편리한 새로운 기능들이 추가되고 있다. 자동차 산업은 점차 통합 모듈 제어 방식으로 진화되고 있으며, 성능과 디자인뿐 아니라 편의성이나 안전 등의 다양한 서비스 기술의 개발까지 범위가 확대되고 있다. 앞으로도 미래지능형 자동차와 관련한 연구 개발과 맞물려 전자/전기 시스템은 더욱 복잡해 질 것으로 예측하고 있다. 이러한 변화들의 결과로 자동차 회사는 다양하고 복잡한 차량전자 장치들과 소프트웨어까지 개발해야 하는 상황에 이르게 되었다. 한 분석 자료에 따르면 2000년도 이후부터 자동차 소프트웨어의 양이 해마다 50%정도 증가한 것으로 나타났다[1]. 따라서 많은 양의 소프트웨어를 효율적으로 개발하기 위하여, 사람이 직접 작성하는 코드 기반의 개발에서 점차적으로 모델 기반의 개발 (Model-based Development)형태로 차량 임베디드 소프트웨어의 개발 프로세스가 진화하고 있다.

또한 AUTOSAR 표준플랫폼은 하드웨어와 소프트웨어의 분리를 통하여 소프트웨어의 재사용성과 이식성을 보장할 뿐만 아니라 모델 기반의 개발환경에 적합한 컴포넌트 구조의 인터페이스를 제공하기 때문에 모델 기반의 설계기법을 접목함으로써 장점을 최대화 할 수 있다. 즉, 모델기반 기법설계를 통해 보다 빠르고 정확한 설계를 할 수 있고 개발 초기 단계에 충분히 모델 검증을 할 수 있으며 결과적으로 하드웨어와 독립적으로 재사용 가능한

* 교신저자(Corresponding Author)
논문접수 : 2009. 10. 30., 수정일 : 2009. 11. 24.,
채택확정 : 2009. 12. 17.
박광민, 금대현, 이성훈 : 대구경북과학기술원
※ 본 연구는 교육과학기술부에서 지원하는 기관 고유사업비로 수행하였음.

모델을 개발할 수 있다. 하지만 모델 기반의 개발 프로세스로 인해 개발시간 단축, 소프트웨어 재사용성 증대 등 효율성 측면에서는 소프트웨어의 진화가 이루어졌지만, 차량 소프트웨어 품질 및 오류검사 등에 대한 테스트는 충분히 이루어지지 않고 개발이 이루어지고 있다. 결국 충분한 소프트웨어 신뢰성을 확보하지 못한 채 소프트웨어의 복잡도가 증가함에 따라 소프트웨어 오류로 인한 차량 고장이 해마다 늘어나고 있는 추세이다. 따라서 차량 내 임베디드 소프트웨어의 신뢰성과 안정성을 확보하기 위해서는 차량 소프트웨어 모델 설계 단계에서부터 정확한 테스트 방법과 검증 방법을 적용하여 소프트웨어 품질을 향상시키기 위한 노력이 필요하다. 본 논문에서는 AUTOSAR 플랫폼을 적용한 모델기반의 응용소프트웨어 개발 및 테스트 방법을 제시하고 사례연구를 통해 이를 검증한다[2].

II. AUTOSAR SWC 방법론

1. AUTOSAR SWC 개념

AUTOSAR는 기본적으로 컴포넌트 기반 소프트웨어 개발 방법론을 따르고 있으며 소프트웨어 개발 단계마다 해당 행위(Activity)를 지원해주는 도구 및 플랫폼을 사용하고 있다.

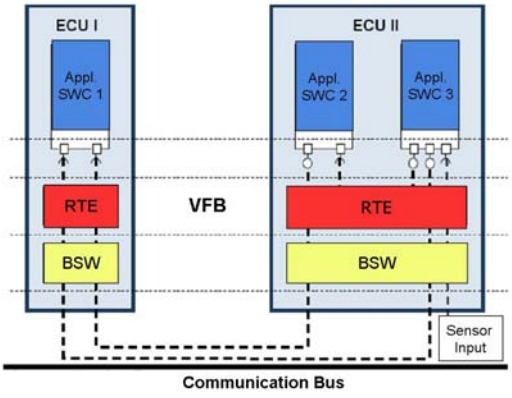

그림 1. AUTOSAR SWC 개념
Fig. 1. AUTOSAR SWC concept

AUTOSAR는 응용시스템을 구성하는 소프트웨어 컴포넌트들을 연결해 주는 VFB(Virtual Function Bus)라는 가상 버스 상에 컴포넌트를 배치하고 이를 적절한 ECU에 할당하여 응용시스템을 구성한다. 가상버스를 제공하는 RTE(Runtime Environment) 계층은 각 SWC 사이 및 SWC와 BSW(Basic SoftWare) 사이의 데이터 교환을 관장하는 통신센터의 역할을 하며, 동일한 인터페이스와 서비스를 제공한다. 따라서 응용 소프트웨어 컴포넌트 개발자가 하드웨어 종속적인 BSW 계층 및 네트워크 구성에 독립적으로 컴포넌트를 개발 할 수 있는 환경을 제공함으로써 재사용성을 극대화 하고 있다[3-4].

2. AUTOSAR 개발방법론

AUTOSAR 표준의 개발방법론은 크게 상위 설계에 해당하는 시스템 설계 단계와 ECU(Electronic Control Unit) 설계 단계로 나누어진다. 시스템 단계에서는 SWC의 데이터 타입, 인터페이스 등을 정의하고 응용 소프트웨어 구현을 위한 런어블(Runnable)과 트리거 조건을 설계한다. 그리고 기본설계를 마친 SWC를 각 ECU에 맵핑하고 네트워크 설계를 함으로써 SWC 설계를 마친다. ECU 설계 단계에서는 System Configuration Description으로부터 각 ECU정보를 추출하고 태스크 정의, RTE 및 BSW 등을 설계한다. 마지막으로 응용 소프트웨어, RTE, OS 등의 코드를 생성하고 컴파일, 링크를 수행하여 ECU Executable환경을 구성한다[5-6].

III. 임베디드 소프트웨어 테스트

소프트웨어 테스트 작업의 유형과 효율성은 소프트웨어 프로세스 모델과 밀접한 관계가 있다. 테스트 작업이 프로그램에 포함된 단순한 코딩의 오류만을 찾는 작업이 아니라 요구 분석에서의 오류, 설계 등 개발 단계의 작업들에 대한 테스트까지 모두 포함하기 때문에 개발 프로세스와 맵핑시킬 필요가 있다. 일반적으로 임베디드 소프트웨어는 V-프로세스 모델을 따라서 개발 및 테스트를 수행한다. V-모델의 해석은 구현부분에 대해서는 단위 테스트를, 디자인 부분에서는 통합 테스트를, 구현된 시스템과 사양간의 비교·분석부분에서는 시스템 테스트를, 요구사항 부분에서는 인수테스트를 각각 설계/수행한다는 것을 의미한다[7].

1. 정적 테스트(Static Test)

정적테스트는 소프트웨어가 개발완료되기 전에 요구사항 정의서, 설계서, 코드 등의 개발 중간 산

출물을 실행 없이 리뷰나 정적 분석과 같은 기법을 통해 테스팅 하는 것이다. 이것은 동적 테스트 등 다른 테스트보다 먼저 수행되기 때문에 개발 초기에 결함을 줄이고 개발기간을 단축하는데 기여할 수 있다. 이 단계에서는 개발중간에 코드 뿐 아니라 요구사항, 테스트 스크립트 등을 검토하고 테스트하는 리뷰나 개발표준검사, 코드 무결성 및 소프트웨어 모델의 구문 규칙(Syntax)을 테스트하는 코드 검증과 정적 분석을 수행한다[8]. 표준위반이나 요구사항 결함, 개발 설계 결함 등의 문제점을 조기에 발견하고 수정할 수 있기 때문에 코드와 설계의 유지보수성이 크게 향상될 수 있다. 자동화 도구 등의 도움을 받아 개발설계 결함 등을 사전에 진단하고 수정할 수 있기 때문에 코드와 설계의 유지보수성이 향상된다.

1.1 리뷰/정적 분석(Review/Static Analysis)

소프트웨어 개발 중간산출물을 검토하고 개발표준 위반이나 상위 레벨의 개발문서와의 불일치 등의 중요한 이슈들을 검증하고 테스트하기 위하여 반드시 수행되어야 할 테스트의 한 부분이다. 대부분의 경우 자동화 도구의 지원으로 모델과 코드 알고리즘의 결함을 발견할 수 있다.

1.2 코딩 표준 검사(Coding Standard Check)

C프로그램 언어는 대부분 자동차 산업에 응용되는 실시간 임베디드 시스템에 사용된다. 하지만 구현의 관점에서 프로그래밍 언어는 마지막 실행 코드가 프로그래머의 의도대로 정확히 동작할 지 여부는 보장할 수 없다[9]. 이러한 문제점을 해결하기 위하여, 차량 응용 소프트웨어를 개발함에 있어서 개발 가이드 라인이자 표준인 MISRA-C를 준수하여 개발을 수행함으로써 정적 에러, 컴파일러의 런타임 오류 등 많은 부분의 오류발생 가능성을 줄일 수 있고, 안정성을 확보할 수 있다. 코딩 규약과 표준을 검증하기 위해서, 보통 프로그래머 및 개발자가 직접 수동으로 체크하기보다 자동화 도구의 도움을 받아 테스트를 수행하게 된다.

1.3 코드 검증(Code Verification)

소프트웨어 개발 단계에, 임베디드 코드가 안전하고 버그가 없고 태스크 수행시 결함이 없음을 일차적으로 검증하기 위해서는 코드 확인검증을 통한 무결성 테스트가 필수적이다. 예를 들어 구문/문법 오류와 같은 특정 코드오류는 개발 초기에 발견이 쉽기 때문에 조기 수정하여 소프트웨어가 보다 신뢰성 있게 되고 잠재적인 오류가 줄어들게 된다[10]. 그림 2와 같이 코드검증 개발도구를 사용하여 코드오류를 체크하고 수정함으로써 코드 무결성을 확인할 수 있다.

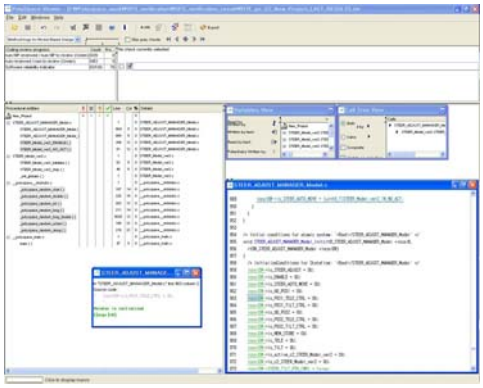

그림 2. 소프트웨어 코드 검증
Fig. 2. Software code verification

2. 동적 테스트(Dynamic Test)

정적 테스트만으로는 실시간 시스템의 기능 모델을 검증하기에는 충분하기 않기 때문에 임베디드 소프트웨어의 동적 특성들을 고려한 동적 테스트를 수행하여 충분히 검증할 필요가 있다. 이러한 방법은 화이트박스 테스트나 블랙박스 테스트와 같이 실시간 임베디드 시스템을 위한 동적 테스트를 근간으로 한다. 화이트 박스 테스트는 테스트 오브젝트의 내부 구조를 검증을 통해 시스템의 구현이 잘 되었는지를 검증하고 테스트 커버리지를 측정하는 것에 주로 초점을 맞춘다. 그리고 시스템의 명세를 기반으로 테스트 케이스와 테스트 시나리오를 결정하여 블랙박스 테스트를 수행함으로써 소프트웨어를 기능적으로 검증할 수 있다[11]. 테스터는 구성된 테스트 시나리오에 맞게 소프트웨어 모델에 특정 입력을 인가하고 해당 출력이 예상된 기대값에 부합하는 지 여부를 평가함으로써 테스트는 종료된다.

IV. 임베디드 SW의 모델기반 개발 및 테스트방법

1. AUTOSAR SWC의 모델기반 개발 방법

일반적으로 AUTOSAR 소프트웨어는 대부분 하

향식(Top-Down) 설계 방식으로 개발하고 있다. 하향식 설계 방법은 시스템 상위 단계의 기본 아키텍처 설계를 시작으로 단계적으로 하위 소프트웨어의 내부 구조와 모듈들을 상세하게 구현해 나가는 방식이다. 하지만 하향식 개발방식의 경우, 상위 시스템의 기본 설계가 완성될 때까지 하위 레벨의 코딩과 모델링을 개시할 수 없다. 그러므로 설계가 대부분 완료될 때까지 소프트웨어 내부의 주요한 기능의 테스트는 할 수 없기 때문에 AUTOSAR 소프트웨어의 개발과정에서 기능 검증이 효과적으로 이루어지기 어렵다. 반면, 상향식(Bottom-Up) 개발 방식은 시스템을 구성하는 세부 구성요소를 먼저 설계하고 모듈을 조합하여 시스템으로 점차 확장해 나가기 때문에 작은 모듈 단위의 설계가 완료한 시점부터 코딩과 그 테스트를 개시할 수 있다. 하지만 상향식 설계방식은 모듈간의 인터페이스가 명확화되어 있지 않으면 나중에 설계의 일부 또는 전체의 변경이 발생할 수 있다는 단점이 있다. 따라서 본 연구에서는 보다 효율적이고 신뢰성 있는 소프트웨어 컴포넌트를 설계하기 위해서 하향식 설계와 상향식 설계를 조합한 모델 기반의 개발 방법을 적용한다. 즉, 하향식 개발방식의 장점인 설계의 효율성, 수정의 용이성을 부각시키면서 테스트의 비효율적인 측면을 보완하기 위하여 개발 중간단계에서 상향식 개발방식을 접목하여 충분히 검증하는 과정을 거친다. 마지막으로 검증된 SW 모듈을 다시 상위 시스템 단계에서 SWC와 통합하고 AUTOSAR 아키텍처 설계를 완료함으로써 개발을 종료한다.

그림 3과 같이 먼저 Vector Informatik의 Davinci Tool Suite와 같은 아키텍처 설계 도구를 사용하여 소프트웨어 기본 구조 설계를 한다. 하위 모듈을 모델링하고 테스트하기 위한 기본 템플릿을 생성하기 위하여 AUTOSAR 상위 아키텍처 및 인터페이스의 기본 설계 정보를 SWC Description 데이터 포맷으로 이출(Export) 한다. 그리고 상위 설계정보를 다시 모델 기반 도구로 이입(Import)하여 AUTOSAR 인터페이스에 맞게 소프트웨어 기능을 먼저 구축한다. 그리고 세부기능 요구사항을 만족하도록 기능을 추가해 나가면서 동시에 상향식 방식으로 각 단계별 하위 모듈을 테스트하고 모듈을 벗어나 SWC 레벨, 시스템 레벨로 확장해 나가면서 상위 시스템 단계의 설계를 구체화할 수 있다. 마지막으로 AUTOSAR SW 태스크 및 런어블 설계, RTE 구현 등 상위 모델의 설계와 테스트를 수행하여 응용 소프트웨어 컴포넌트의 개발을 완료한다. 이와 같은 방법으로 개발함으로써 하향식 개발방식의 최대 단점인 낮은 테스트성(Low Testability)을 개선할 수 있고, 따라서 기능 모델 설계자는 AUTOSAR 응용소프트웨어 개발의 각 단계별로 충분한 모델검증을 수행할 수 있다.

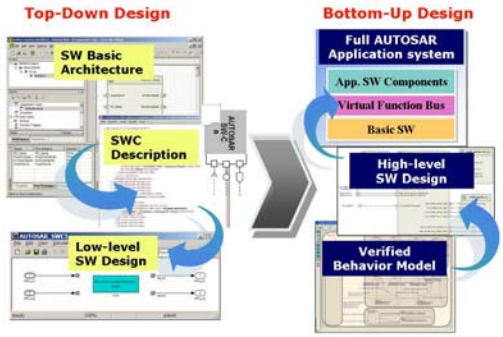

그림 3. AUTOSAR시스템의 모델기반 개발방법
Fig. 3. Model-based development for AUTOSAR

2. AUTOSAR SW의 모델기반 테스트 방법

AUTOSAR 임베디드 소프트웨어는 기존의 시스템에 없는 런어블, 태스크, Atomic/Composition SWC, AUTOSAR 표준 인터페이스, RTE 등의 개념이 있기 때문에 기존의 소프트웨어 테스트 방식 그대로 적용하여 테스트하기에는 많은 어려움이 따른다.

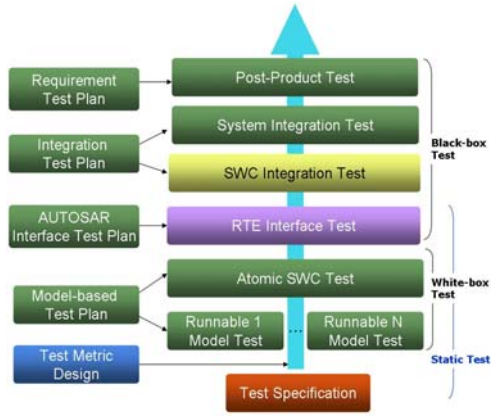

그림 4. AUTOSAR SW의 모델기반 테스트 프로세스
Fig. 4. Model-based test process for AUTOSAR

또한 하드웨어와 독립적으로 테스트 환경을 구축하고 테스트하기 위해서는 RTE와 AUTOSAR 인터페이스 테스트를 수행한 후에 전체적인 통합테스

트를 수행할 필요가 있다. 모델 기반의 기법을 도입하여 AUTOSAR 소프트웨어를 구현 및 검증하기 위해서는 설계뿐만 아니라 테스트 방법의 수정이 불가피하다.

그림 4는 본 연구에서 제안하는 AUTOSAR 임베디드 소프트웨어의 모델기반 테스트를 위해 수정된 테스트 프로세스를 나타낸다. 테스트 사양과 메트릭을 설계한 후 각 개발단계별 결과물에 맞는 테스트를 충분히 수행한다. 테스트 대상 모델을 적절한 방법으로 수정하고 평가하여 테스트 요구사항을 만족하도록 한다.

2.1 테스트 메트릭 설계(Test Metric Design)

테스트 계획 및 설계 명세 단계에서 응용 소프트웨어의 특성을 고려하여 테스트 메트릭을 설정하고 이에 따른 테스트 완료 조건을 정하여 단계별 테스트를 위한 테스트 사양을 확정한다. 먼저 임베디드 소프트웨어 평가 목표 모델을 설정하기 위하여 표 1과 같은 후보 테스트 메트릭을 설정한다. 본 논문에서는 SW 품질평가를 위한 표준지표로 가장 일반적으로 적용되는 국제 표준 위원회(ISO)의 ISO-9126에서 제시하는 메트릭을 기반으로 정적 테스트의 메트릭을 선정하였고, 그 외에 MISRA-C 표준에 근거한 표준적합성(Coding Violation rate)과 코드자체의 구문오류 발생비율(Coding Violation rate)까지 포함하여 정적테스트의 메트릭을 설계하였다[12].

표 1. SW 품질평가를 위한 테스트 메트릭
Table 1. Test metrics for SW quality evaluation

Static Test Metrics	Standard Conformance rate, Cyclomatic Complexibility, Testability, Clarity, Maintainability, Reliability, Functionality, Coding Violation rate.
Dynamic Test Metrics	Statement Coverage, Branch Coverage, MC/DC Coverage, Pass/Fail rate for Functional Test.

그리고 동적테스트에서 화이트박스 테스트는 SW의 내부구조와 테스트 커버리지를 측정하는 것이 주된 목표이기 때문에 임베디드 소프트웨어 테스트에서 가장 적합한 구문 커버리지(Statement Coverage), 분기 커버리지(Branch Coverage), 그리고 변경조건/결정 커버리지 (Modified Condition /Decision Coverage)를 테스트 메트릭으로 설정하였다. 마지막으로 기능 테스트를 위한 블랙박스 테스트의 메트릭은 요구사항 기반의 테스트 통과/실격(Pass/Fail) 비율으로 설계하였다.

임베디드 응용 소프트웨어와 시스템 타겟(Target)의 특성에 따라 테스트 조건과 테스트 요구사항이 달라지기 때문에 응용 시스템 특성을 충분히 고려하여 테스트 메트릭과 테스트 완료조건은 일부 수정 또는 변경되어야 한다. 테스트 계획과 설계명세가 수립되면 테스트 시나리오대로 각 단계별 테스트를 수행한다.

2.2 테스트 절차(Test Procedure)

그림 4에서 제시한 테스트 절차를 따라서 먼저 설계된 테스트 메트릭과 테스트 사양에 기반하여 기능단위의 최소 구성요소인 런어블의 테스트를 수행하고 ECU에 맵핑될 수 있는 단위인 Atomic SWC의 테스트 수행한다. 테스트 완료조건과 부합하는 향상된 정적 테스트 결과와 모델 커버리지가 얻어지게 되면 마지막 단계인 블랙박스 테스트로 넘어가게 된다. AUTOSAR에서는 컴포넌트 간의 정보를 주고받기 위해서는 항상 미들웨어에 해당하는 RTE를 통과하기 때문에 컴포넌트 통합테스트에 앞서 RTE 검증과 AUTOSAR 인터페이스에 대한 테스트가 먼저 수행이 되어야 한다. 검증된 컴포넌트와 RTE를 ECU 단위에서 통합하여 사용자 요구사항 기반의 블랙박스 테스트를 수행하여 최종적인 동적기능테스트를 만족함으로써 모델기반의 프로토타입 개발이 완료된다.

2.3 테스트 모델 수정 및 평가

각 단계별로 수행된 테스트 대상 모델은 적절한 방법으로 수정하고 평가하는 과정을 거친다. 모델기반의 테스트를 수행하는 과정에서 기능 모델 및 생성된 SW 품질에 영향을 줄 수 있는 핵심 이슈들을 표 2와 같이 발견했다. 이것들은 모델 및 소스에 직접적으로 영향을 줄 수 있는 중요한 사항이기 때문에 이러한 이슈들을 해결함으로써 소프트웨어의 커버리지를 향상하고 나아가 소프트웨어 품질을 부분적으로 개선할 수 있을 것으로 예상된다.

표 2. SW 품질 개선을 위한 핵심 이슈
Table 2. Critical issues for quality improvement

Factor	Critical issues	Solutions
Reliability	MISRA-C Coding Rule	Code-verification / Modify syntax Error
Testability /Clarity	Priority of State-model	Clear state priority based on Requirement
	Inconsistent State	Modify redundancy and ambiguity of Condition
Complexibility	Interface Consistency	Consistency check of Module vs. Component
	SW cyclomatic complexity	Reduce State Depth
Coverage	Dead State/ Low-Coverage State	Delete Unnecessary State / State Integration
Maintainability	Requirement violation	Link Model to requirements
	Reusable Block	Create Subsystem Block
Conformance	ISO 61508, DO178-B	Check pre-validated with Model Advisor

V. 사례 연구 - 운전자 위치제어 시스템

앞에서 설명한 바와 같이, 모델 기반의 설계 및 테스트를 AUTOSAR 플랫폼에 적용함으로써 응용 소프트웨어의 재사용성 및 신뢰성 등이 크게 향상된다. 본 논문에서 제시한 모델기반의 응용소프트웨어 개발 및 테스트 방법을 운전자 위치제어 시스템(DPS)에 적용하고 테스트 수행 결과를 분석하였다

1. 응용 시스템 모델 설계

DPS 시스템은 그림 5와 같이 운전자 제어 ECU, 조향 제어 ECU, 시트 제어 ECU로 구성되고 CAN 프로토콜을 사용하여 통신을 수행한다. 차량 속도, 기어의 현재 상태 등의 차량 기본 데이터는 바디제어 ECU에서 통신으로 제공한다. 운전자제어부에서는 조향과 시트의 위치를 수동으로 제어하는 모드와 조향과 시트의 현재 위치값을 미리 메모리에 저장시켰다가 필요시 자동으로 복원하는 모드로 나누어서 시스템의 동작방식을 결정한다. 그리고 조향과 시트제어 ECU에서는 중앙에 있는 운전자 제어 ECU에서의 원격입력을 받거나 자체 ECU에서의 입력을 받아서 직접 구동모터를 동작시켜서 위치제어를 하게 된다. 본 연구에서는 운전자 위치제어모델기반의 테스트를 고려한 AUTOSAR 시스템 개발을 위해 하향식 설계와 상향식 설계를 조합한 개발 방식을 적용하였다. 설계된 소프트웨어 기본 아키텍처를 바탕으로 모델링 도구에서 이입하여 기능 제어 모델을 개발하였고, Vector Informatik의 DaVinci Tool Suite를 사용하여 상세한 AUTOSAR

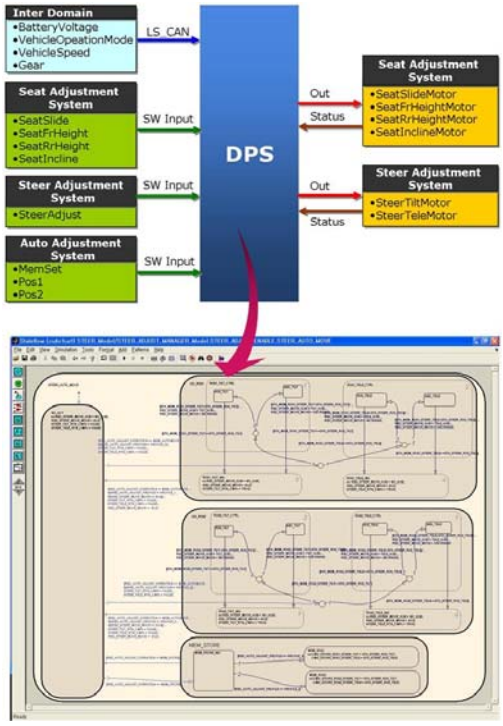

그림 5. DPS 시스템 설계 및 모델링
Fig. 5. DPS system design and modeling

기반 상위 시스템 설계를 하였다. DPS시스템은 차량 바디 응용 시스템으로써 복잡한 수학적 알고리즘이 아닌 상태기반으로 제어되는 시스템이기 때문에 그림 5와 같이 상태도(State-Diagram) 기반으로 대부분 모델링하였다.

2. 모델기반 테스트 요구사항

앞에서 제시한 테스트 메트릭을 본 응용시스템의 요구사항 및 특성을 고려하여 표 3과 같이 테스

트 요구사항을 설계하였다. 본 사례연구에서 설정한 메트릭의 목표값들은 관련 국제표준문서와 타 임베디드 시스템들과 타 도구에서의 데이터를 참조하여 가장 기본이 되는 목표값으로 설정하였다[12-15]. 특히 정적테스트 메트릭 기준값은 차량 바디 응용 소프트웨어에 일반적으로 적용이 되는 값을 채택하였다. 또한 응용 타겟이 차량 ECU에 탑재되는 시스템이기 때문에 소프트웨어의 안정성에 직접적으로 관련이 있는 코드 검증과 기능 테스트의 실패로 관련이 있는 코드 검증과 기능 테스트의 실패 확률을 0%로, 그리고 커버리지 역시 90% 이상으로 타 응용 시스템에 비해 상대적으로 높은 테스트 요구사항으로 설계하였다.

표 3. DPS 시스템을 위한 테스트 요구사항
Table 3. Test requirements for DPS system

Static Test Metrics	Complexibility	7 이하
	Testability	80% 이상
	Clarity	70% 이상
	Maintainability	80% 이상
	MISRA Violation rate	5% 이하
	Standard Conformance rate	95% 이상
	Code Verification	No Err.
Dynamic Test Metrics	Statement Coverage	100%
	Branch Coverage	90% 이상
	MC/DC Coverage	90% 이상
	Pass/Fail rate for Functional Test	0% Fail

3. 정적 테스트

개발 중간 단계의 소프트웨어 모델 및 코드는 개발 표준검증, 리뷰, 정적 분석 등의 과정을 거쳐 테스트 사양에 부합하는지 여부를 충분히 검증하고 수정되어야 한다. 먼저 앞에서 언급한 AUTOSAR의 모델기반 테스트 개념에 맞도록 런어블, Atomic SWC, RTE 각 개발 단계별로 정적 테스트를 수행하여 테스트 메트릭과 관련된 품질 지표들을 측정한다. 또한 앞에서 언급한 SW 모델의 품질 향상을 위한 핵심 이슈와 해결방법을 적용하여 모델 및 프로그램 코드를 수정하였다. 1차 수정된 모델은 그림 6과 같이 Mathwork사의 Matlab Model Verifier와 같은 모델 검증 도구를 사용하여 모델 테스트를 수행하였고, DO-178B, IEC61508 등의 개발표준에 적합하도록 검증하였다.

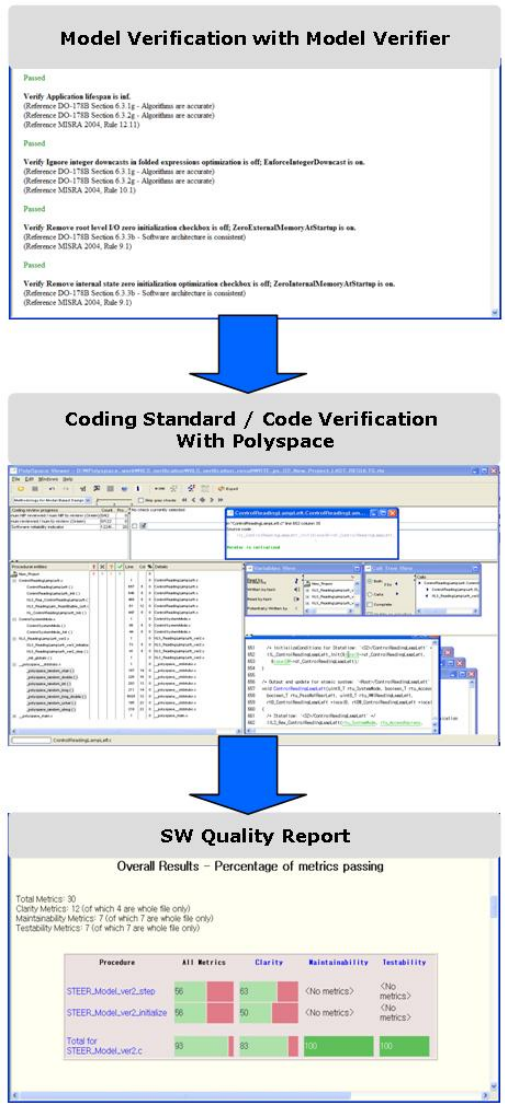

그림 6. SW 모델의 정적 테스트
Fig. 6. Static test for SW model

잠재적인 오류를 방지하기 위해 수행 중간에 발생되는 표준 위반, 모델링 규약 위반 등의 수정이 필요하다. 생성된 코드레벨에서는 MISRA-C 표준 적합성 검사와 코드 구문오류검사를 수행하기 위하여 코드 검증을 위한 대표적인 상용도구인 Polyspace를 이용하여 테스트를 수행하였다. 반드시 에러가 없음을 확인하고 테스트를 종료한다. 마지막으로 그림과 같이 SW 품질에 대한 결과 보고서를 검토하여 원하는 결과가 나오는 것을 확인하게 되면 정적 테스트를 종료한다.

4. 동적 테스트

4.1 화이트 박스 테스트(White-box Test)

화이트박스 테스트는 소프트웨어의 구조적 검증을 수행하여 테스트 커버리지를 산출하는 데 그 목적이 있다. 앞에서 설정한 테스트 메트릭 중 커버리지 목표값을 만족하기 위하여 SW품질 개선을 위한 핵심 이슈들을 적용하고 테스트를 반복 수행함으로써 모델의 질을 개선하고 커버리지를 향상하였다.

4.2 블랙 박스 테스트(Black-box Test)

화이트박스 테스트를 통과한 소프트웨어 컴포넌트는 소프트웨어 설계사양과 시스템 명세 기반의 기능적 검증을 위하여 통합테스트 및 시스템 레벨에서 RTE를 포함한 소프트웨어의 블랙박스 테스트를 수행하였다. DPS 시스템은 상태 다이어그램 기반으로 기능모델이 설계되어 있기 때문에 입출력 조건과 상태변화를 중점적으로 테스트하는 기법인 상태 천이 테스트 기법으로 요구사항에 근거하여 테스트케이스를 작성하였다.

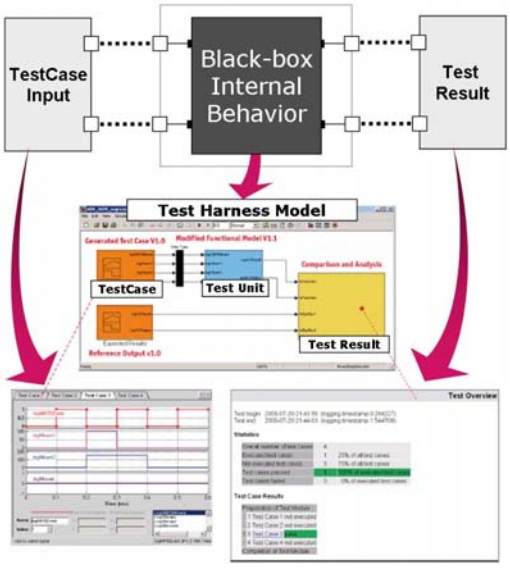

그림 7. 기능요구사항 검증을 위한 블랙박스 테스트
Fig. 7. Black-box test for functional requirement test

그림 7에서와 같이 테스트 할 대상인 테스트 오브젝트를 정하고 내부 기능모델에 대한 테스트 하네스(Harness) 모델을 생성한다. 요구사항에 기반하여 미리 설계된 테스트케이스 입력을 테스트 유닛(Unit)에 인가하여 나오는 테스트 결과를 기대값과 비교하여 일치할 경우 Pass로 처리하고 다른 결과값이 나오게 되면 Fail로 처리한다. 본 연구를 위해서 테스트 하네스 모델을 자동으로 생성하고 통과/실격 테스트 보고서 생성까지 수행하는 Simulink의 테스트 툴셋을 사용하였다. 실험 결과 100% 기능 요구사항을 만족하는 것을 확인하였다.

5. 테스트 수행 결과 (Quality Evaluation)

표 4는 모델 기반 테스트 방법을 DPS 시스템에 적용하여 테스트를 수행한 결과를 보여준다. AUTOSAR 기반 응용 소프트웨어의 품질을 평가하기 위하여 앞에서 설계한 각 테스트 메트릭 항목에 대하여 단계별 검증과 모델 테스트를 적용하기 전과 후로 나누어 측정하였다. 메트릭 측정을 위해서 소프트웨어 신뢰성 측정 도구인 LDRA와 Matlab의 V&V검증 도구를 사용하였다. 먼저 화이트박스 테스트의 결과물인 커버리지를 측정하기 위해서는 설계모델에 대해서 중복되지 않는 경우의 수를 모두 산출해서 Formal Method를 적용한 테스트케이스 입력을 인가하여 그에 따른 커버리지를 Matlab V&V도구를 이용하여 측정하였다. 단계별 모델 검증을 하지 않고 얻은 테스트 커버리지는 70%를 밑도는 결과가 얻어졌지만 단계별 테스트를 수행하고 앞에서 설명한 핵심이슈들을 모두 반영하여 수정한 결과 커버리지를 100%까지 개선시킬 수 있었다.

표 4. 전체 메트릭에 대한 테스트 결과
Table 4. Test result for whole metric

Metrics	Test Result		Improvement Rate(%)
	Before	After	
Statement Coverage	72%	100%	39
Branch Coverage	69%	100%	44.9
MC/DC Coverage	57%	100%	75.4
Complexibility	8	5	37.5
Testability	70%	100%	42.9
All Metrics	79%	93%	17.7
Maintainability	80%	100%	25
Clarity	60%	83%	38.3
MISRA Violation rate	13%	6%	53.8
Standard Conformance rate	55%	90%	63.6
Code Verification	6 Err.	0 Err.	-
Pass/Fail rate for Functional Test	4 Fail	0 Fail	-

그 외 기능 테스트를 제외한 정적 테스트 메트릭을

측정하기 위하여 모델에서 생성한 코드를 정적테스트 도구인 LDRA의 TestBed 에서 실행시켜 복잡도(Complexibility), 유지보수성(Maintainability) 등의 값을 얻을 수 있다. 마찬가지로 체계적인 모델 기반 테스트를 수행하기 전과 후에 대하여 각각 비교하여 측정하였다. 실험결과 평균적으로 메트릭의 측정값이 30% 이상으로 개선되는 것을 확인할 수 있다. 마지막으로 시스템 통합 단계에서 수행한 요구사항 기반의 기능테스트 결과에서도 기능 요구사항의 테스트케이스 실패 가능성을 줄일 수 있음을 확인하였다.

응용 소프트웨어의 신뢰성과 질을 향상하기 위하여 체계적으로 모델기반의 테스트를 수행하고 소프트웨어 모델에 직접적으로 영향을 줄 수 있는 각 항목에 대한 핵심 이슈들을 해결함으로써 궁극적으로 소프트웨어의 품질 및 테스트 커버리지 등이 개선되는 것을 확인할 수 있었다. 또한 테스트 계획에서 설정한 테스트 메트릭과 테스트 완료 조건을 만족하여 응용소프트웨어의 신뢰성을 확보할 수 있다. 본 사례연구에서 측정한 12가지의 메트릭은 대표적인 소프트웨어 품질측정을 위한 항목으로서 여러 가지 다양한 메트릭을 추가하여 보다 객관성을 확보할 수 있을 것으로 예상된다.

소프트웨어의 내부 품질 평가 외에 테스트의 가용성과 응용성을 고려해 볼 때 테스트를 수행하는 시간에 대한 평가도 중요하다. 그림 8은 일반적인 차량 임베디드 소프트웨어의 개발 및 테스트 수행시간과 본 연구에서 수행한 DPS 소프트웨어의 개발 및 테스트 시간을 비교한 도표이다[16].

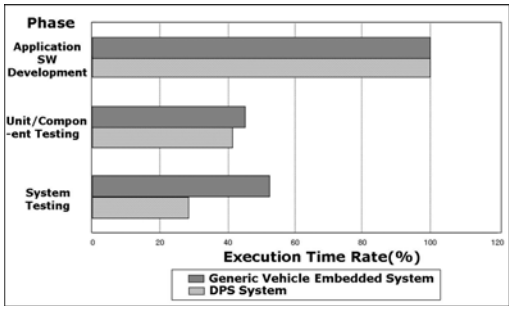

그림 8. 단계별 테스트 수행시간 비교
Fig. 8. Test execution time per phase

서로 다른 응용 시스템의 개발과 테스트 시간을 절대적으로 비교하는 것은 의미가 없기 때문에 SW 개발과 테스트에 소요되는 절대적인 시간을 측정하여 상대적인 비율로서 도식화하였다. 그래프의 그림은 응용SW개발 시간을 100%로 봤을 때 각 단계별 테스트에 소요되는 시간을 상대적인 비율로서 나타낸 결과이다. 그림에서 테스트에 소요되는 시간을 비교해 볼 때 일반적인 전장 소프트웨어 테스트에 비해 단계별 모델 검증 등을 거친 본 DPS 응용시스템의 테스트 수행시간이 전체적으로 단축되는 것을 확인할 수 있다. 특히 소프트웨어 모듈이나 컴포넌트 단위의 테스트보다 시스템 테스트에 소요되는 시간의 비율이 50%에서 30% 정도로 크게 감소하였다. 이는 개발과 병행하여 모델 기반 테스트를 충분히 수행함으로써 대부분의 결함을 조기에 발견하여 수정 및 개선하였기 때문에 상위 시스템 레벨의 테스트에 소요되는 시간 및 비용이 절감된다고 할 수 있다.

VI. 결론

본 연구에서는 AUTOSAR 표준 플랫폼 기반 소프트웨어의 개발과 병행하여 단계별로 모델기반의 테스트를 하는 방법을 제시하였다. 또한 SW 품질에 직접적으로 영향을 줄 수 있는 핵심 이슈들을 제시하고 적용함으로써 모델 테스트의 단점을 보완하고 품질을 향상하도록 하였다. AUTOSAR 아키텍처로 설계된 운전자 위치제어 시스템에 본 모델 테스트 방법을 적용한 결과, 소프트웨어의 신뢰성과 품질이 개선되고 테스트에 소요되는 시간 및 자원이 절감되는 것을 확인할 수 있었다. 본 연구에서 제시한 방법과 같이 차량 소프트웨어의 모델 설계 단계에서부터 정확한 테스트 방법과 검증 방법을 적용함으로써 차량 임베디드 소프트웨어의 품질과 신뢰성 향상에 기여할 수 있을 것이다.

참고문헌

[1] Automotive Electronization Technology and Design/Development Process, THN, 2008.

[2] D. Rai, T. Jestin, "Model-based development of AUTOSAR-compliant application", SAE World Congress, 2008.

[3] AUTOSAR Layered Software Architecture, 2007, AUTOSAR Specification R.3.0.

[4] AUTOSAR Interaction with Behavioral Models, 2007, AUTOSAR Specification R.3.0.

[5] AUTOSAR Methodology, 2007, AUTOSAR Specification R.3.0.

[6] J. Son, D. Kum, "Development of automotive system based AUTOSAR", KSAE, 2007.

[7] W. Oh, J. Shin, "Model based development process for the embedded system in vehicle", SAE World Congress, 2008.

[8] W. Kwon, "Practical software testing foundation", pp. 80-96, STA, 2008.

[9] MISRA-C: Development Guidelines For Vehicle Based Software, http://www.misra.org.uk.

[10] Mathworks: PolySpace for C, Technical Contents.

[11] G. Park, D. Kum, S. Lee, "Test methods of the AUTOSAR application software components", ICCAS-SICE, 2009.

[12] ISO/IEC 9126: Quality Model and Process for Evaluating Quality, 1991.

[13] DO178B: Software Considerations in Airborne Systems and Equipment Certification, 1999.

[14] Myers, J. Glenford, "The art of software testing", John Wiley & Sons Inc., USA, pp. 148-155, 2004.

[15] J. Herrmann, "Guideline for validation & verification real-time embedded software systems", ITEA, pp. 61-64, 2001.

[16] 홍상균, "임베디드 SW 테스팅 안전과 신뢰의 시작", 한국소프트웨어진흥원, 11-16쪽, 2008.

저 자 소 개

박 광 민

2005년 한국항공대 항공기계학과 학사.
2007년 광주과학기술원 기계전자과 석사.
현재, 대구경북과학기술원 연구원.

관심분야: 임베디드 시스템, SW테스트.
Email: ggangmin@dgist.ac.kr

금 대 현

2001년 계명대학교 자동차공학과 학사.
2003년 계명대학교 자동차공학과 석사.
현재, 경북대학교 전자전기컴퓨터학과 박사과정.

2003~2005 LG전자 연구원.
2005~현재 대구경북과학기술원 연구원.
관심분야: 임베디드소프트웨어, 테스팅자동화.
Email: kumdh@dgist.ac.kr

이 성 훈

1996 경북대학교 전자공학과 학사.
1998 경북대학교 전자공학과 석사.
2007 경북대학교 전자공학과 박사.

1999~2002 대우 정밀 기술연구소.
2002~2005 국방과학연구소.
2005~현재 대구경북과학기술원 선임연구원.
관심분야: 차량 임베디드 시스템.
Email: shunlee@dgist.ac.kr