< 기술 논문 >

AUTOSAR 대응 경량화 소프트웨어 아키텍처를 이용한
어플리케이션 소프트웨어 모델링 및 통합 방법

박 인 석¹⁾ · 이 우 택²⁾ · 선 우 명 호^{*3)}

한양대학교 대학원 자동차공학과¹⁾ · 창원대학교 제어계측공학과²⁾ · 한양대학교 미래자동차공학과³⁾

Application Software Modeling and Integration Methodology
using AUTOSAR-ready Light Software Architecture

Inseok Park¹⁾ · Wootaik Lee²⁾ · Myoungho Sunwoo^{*3)}

¹⁾Department of Automotive Engineering, Graduate School, Hanyang University, Seoul 133-791, Korea

²⁾Department of Control & Instrumentation Engineering, Changwon National University, Gyeongnam 641-773, Korea

³⁾Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea

(Received 13 February 2012 / Revised 16 May 2012 / Accepted 14 June 2012)

Abstract : This paper describes a model-based software development methodology for AUTOSAR-ready light software architecture(AUTOSAR-Lite). The proposed methodology briefly represents an application software modeling technique using Matlab/Simulink. Using the proposed technique, application software architecture elements (e.g. software components, runnables, and interfaces) and functional behaviors can be designed in a single modeling environment. From the designed model, the codes of application software is automatically generated using Real-Time Workshop Embedded Coder. The generated application software is easily integrated with hand-coded basic software using the proposed method. In order to evaluate the proposed methodology, a diesel engine management system for a passenger car was employed as a case study. Based on the methodology, 8 atomic software components and 52 runnables are successfully developed, and they are evaluated by engine experiments. From this case study, AUTOSAR compatible model-based application software was successfully developed, and the effectiveness of the proposed methodology was evaluated.

Key words : AUTOSAR(오토사), Model-based software design(모델기반 소프트웨어 설계), Matlab/Simulink(매틀랩/시뮬링크), Code generation(코드생성), Integration methodology(통합 방법론), Diesel engine management system(디젤엔진제어시스템)

1. 서 론

소비자들의 편의를 위한 요구사항 증가 및 각종 규제 강화로 인하여 차량에 탑재되는 내장형 제어 시스템들의 규모는 1990년도 부터 비약적으로 증가하고 있다. 현재 고급차량의 경우 약 70 여 개의 전자제어시스템들이 탑재되며, 약 2500여 개의 정보들이 차량 네트워크를 통해 공유된다.¹⁾ 급격하게 증

가한 내장형 제어시스템의 규모 만큼 기능 복잡도 역시 기하 급수적으로 증가하고 있어, 내장형 제어 시스템의 소프트웨어 개발 비용 및 시간을 줄일 수 있는 효율적인 개발 방법에 대한 연구들이 활발하게 진행되어 왔다.²⁾

내장형 제어시스템 소프트웨어의 효율적인 개발을 위한 대표적인 연구로는 요구사항 도출단계 부터 소프트웨어 구현까지 그래픽 기반의 설계 도구를 활용하는 모델기반 소프트웨어 개발 방법이 있

*Corresponding author, E-mail: msunwoo@hanyang.ac.kr

다.^{3,4)} 기존의 텍스트 기반 소프트웨어 개발 방법에서는 텍스트의 모호성, 개발 단계 별 의사소통 문제 등으로 인하여 많은 개발 비용 및 시간이 소요되었다. 반면에 모델기반 소프트웨어 개발 방법은 전체 개발과정에서 일관된 형상을 갖는 모델을 이용함으로써 의사소통 문제로 인한 오류를 줄이고, 시뮬레이션 기능 활용을 통하여 개발 비용 및 시간을 크게 줄일 수 있는 장점이 있다. 자동차 산업에서는 어플리케이션에 따라 주로 사용되는 모델링 도구가 조금씩 다르긴 하지만, Matlab/Simulink⁵⁾와 ASCET⁶⁾이 널리 이용되고 있으며, 최근에는 양산을 대상으로 한 제품에도 모델 기반 소프트웨어 개발 기법이 적용될 정도로 널리 활용되고 있다.

이와 같이 기술적으로 성숙 단계에 이른 것으로 평가 받는 모델기반 소프트웨어 개발 방법은 최근 자동차 산업에서 소프트웨어 표준화⁷⁻⁹⁾가 중요하게 인식됨에 따라, AUTOSAR와 호환 가능한 형태로 발전하기를 요구받고 있다.¹⁰⁾

지금까지 제시된 AUTOSAR 아키텍처를 위한 모델기반 소프트웨어 개발 방법들은 아키텍처와 functional behavior 모델링을 각각 독립적인 도구를 이용하여 설계하는 형태로 구성되어 있다.¹¹⁻¹³⁾ 예를 들어 아키텍처 모델링은 Vector 사의 Davinci¹⁴⁾나 dSPACE 사의 SystemDesk¹⁵⁾ 같은 도구를 이용하고, functional behavior 모델링은 ASCET이나 Simulink를 이용하는 방법들이 개발되어 있다. 이와 같이 아키텍처와 functional behavior 모델링이 이원화된 개발 환경은 불필요하게 개발 과정을 복잡하게 만들며, 모델링 도구 간의 호환성 문제를 일으키기 쉽다.

또한 고가의 설계 도구들에 대한 의존성이 높기 때문에 AUTOSAR 표준 도입에 적지 않은 부담이 되고 있다.

이 연구에서는 기존에 제시되었던 AUTOSAR 아키텍처를 위한 모델기반 소프트웨어 개발 방법들의 단점들을 극복하기 위하여, 소프트웨어 아키텍처들의 구성요소와 functional behavior 모두를 Matlab/Simulink를 이용하여 모델링, 코드생성 및 통합하는 기법들에 대해서 제시한다. 이 논문의 구조는 다음과 같다. 2장에서는 이 연구에서 사용한 소프트웨어 아키텍처인 AUTOSAR 대응 경량화 아키텍처(AUTOSAR-Lite)에 대해 소개한다. 이를 바탕으로 어플리케이션 소프트웨어를 모델링하는 기법과 통합 방법을 각각 3, 4장에서 제시한다. 마지막으로 5장에서는 이 연구에서 제시한 방법론을 검증하기 위하여 승용디젤엔진 제어시스템 개발에 적용한 결과들을 설명하고 6장에서는 그 결과들에 대해 요약하고 결론을 맺는다.

2. AUTOSAR 대응 경량화 소프트웨어 아키텍처(AUTOSAR-Lite)

이 연구에서는 AUTOSAR 표준 사양에서 필수적인 요소들만 반영하여 경량화 시킨 AUTOSAR-Lite 소프트웨어 아키텍처를 기초로 하고 있다.¹⁶⁾

AUTOSAR-Lite는 AUTOSAR와 마찬가지로 크게 세 가지 계층인 basic software (BSW), runtime environment (RTE), application software (ASW)로 구성되어 있으며, Fig. 1에서 세부 구성 요소들과 인터페이스들을 도식화 하고 있다.

Fig. 1 The structural overview of AUTOSAR-Lite architecture

2.1 Basic software

Analog-to-digital converter (ADC), timer, digital input/output, controller area network (CAN) 등과 같이 하드웨어 의존적인 입출력 기능들을 추상화 시켜 상위 계층에 인터페이스를 제공하는 hardware abstraction layer (HAL) 모듈, 서비스, 스케줄러들로 구성되어 있다.

2.2 Application software

ASW 계층은 단위 기능을 갖는 가장 작은 스케줄링 대상인 runnable, 그리고 runnable 들의 조합으로 이루어져 독립적으로 동작 가능한 모듈인 software component (SwC)들로 구성되어 있다. SwC들은 미리 정의된 인터페이스들을 통해 정보교환이 가능하다.

2.3 Runtime environment

RTE는 BSW와 ASW의 통합을 담당하는 계층으로 RteHal, RteApplication, RteScheduler 모듈들로 구성되어 있다. RteHal 모듈은 BSW와 ASW 간의 정보교환을 server/client 방식의 RteHal 인터페이스를 제공하고 있으며, RteApplication 모듈은 ASW 간의 정보교환을 sender/receiver 방식의 RteApp 인터페이스를 제공한다. RteScheduler의 경우, ASW에서 설계된 runnable들의 스케줄링 정보를 관리한다.

3. Simulink를 이용한 어플리케이션 소프트웨어 모델링 기법

이 장에서는 AUTOSAR-Lite 소프트웨어 아키텍처를 기반으로 하는 ASW의 모델링 기법들에 대해서 소개한다.

모델링에 사용한 도구는 제어알고리즘 설계에 널리 이용하는 Simulink를 사용하였다. ASW의 구성 요소는 composite SwC, atomic SwC, runnable, 인터페이스 들로 구분할 수 있으며, 각각의 구성 요소들은 Fig. 2에서 도시하고 있는 바와 같은 계층 구조를 가지고 있다.¹⁶⁾

3.1 Runnable 모델링

Runnable은 AUTOSAR-Lite 아키텍처에서 기능을 갖는 가장 작은 스케줄링의 단위로 ASW의 계층

Fig. 2 ASW model hierarchy

Fig. 3 Example of runnable modeling

구조에서 가장 하부에 위치한다. Runnable은 재사용이 가능하고, 스케줄링의 단위로서 활용이 되기 때문에 함수의 형태로 코드생성이 이루어져야 한다. 이를 위해 Simulink에서는 Fig. 3의 예제에서 보여주듯이, runnable의 functional behavior 모델을 Simulink의 'atomic subsystem' 블록을 이용하여 그룹 지어주고, 코드생성 옵션을 함수형태로 지정하여 모델링 하였다. Runnable의 기능적인 요소들은 Simulink에서 제공되는 라이브러리들을 이용해서 모델링이 가능하다. 개발이 완료된 runnable들은 Simulink의 라이브러리 파일형태로 변환하여 유지보수성 및 재사용성을 확보한다.

3.2 Composite and atomic SwC 모델링

ASW의 SwC는 두 단계로 구분하여 모델링 한다. 첫 번째는 runnable들만의 조합으로 이루어진 atomic SwC이다. Atomic SwC는 Simulink의 'atomic sub-

Table 1 Summary of AUTOSAR-Lite architecture modeling

Architecture element	Simulink model	Implementation
Runnable	Atomic subsystem block	Function
Atomic SwC	Atomic subsystem block	Module with data object
Composite SwC	Simulink model file (.MDL)	Not implemented
RteHal interface	RteHal Interface block	Server/Client
RteApp Type1 interface (Inner-composite SwC)	Signal object	Sender/Receiver
RteApp Type2 interface (Inter-composite SwC)	RteApp Interface block	
Inter-runnable interface	Signal object	Static variable

Fig. 4 Example of SwC modeling

system' 블록을 이용하여 runnable 들을 묶어 줌으로써 모델링이 가능하다. 이렇게 모델링된 atomic SwC 는 함수 형태로 구현된 runnable들을 포함하고 있는 모듈의 형태로 구현된다. 두 번째 단계는 composite SwC의 모델링으로써, atomic SwC 들의 조합으로 이루어지며, 이 연구에서는 Simulink의 model 파일(.MDL)로 맵핑하였다. Composite SwC 는 ASW 모델의 코드생성 단위로 이용되며, 실제 코드로는 구현되지 않는 가상의 개념이다.

Fig. 4는 SwC 모델링 예제를 간략하게 도시하고 있는 것으로써, 세 가지 종류의 atomic SwC (SwC_FISys, SwC_InjCtl, SwC_CrSys) 들로 구성되어 있는 composite SwC (Sys_InjSys.MDL) 를 도시하고 있다.

3.3 인터페이스 모델링

AUTOSAR-Lite의 ASW의 인터페이스들은 Fig. 1에서 도시하고 있는 바와 같이 RteHal, RteApp, Inter-runnable variable (IRV) 인터페이스들로 구성되어 있다.¹⁶⁾

3.3.1 RteHal 인터페이스

RteHal 인터페이스는 Fig. 1에서 도시하고 있는

Fig. 5 Example of interface modeling using custom block

바와 같이 ASW 계층의 SwC와 RTE 계층의 RteHal 모듈사이의 정보교환을 구현한다. 정보교환의 대상이 ASW 계층에 존재하지 않기 때문에 이 연구에서 개발한 custom 블록(RteHal interface block)을 이용하여 모델링 한다. Fig. 5에서 도시하고 있는 바와 같이 인터페이스 이름을 블록의 설정 파라미터로 입력하여 코드 생성 시 RteHal 모듈에 미리 정의된 인터페이스 함수를 호출하도록 구현하였다.

3.3.2 RteApp 인터페이스

RteApp 인터페이스는 ASW 계층의 SwC 사이에서 발생하는 정보교환을 구현하는 것으로 대상이 composite SwC 내부, 즉 Simulink 모델 파일 내부에 존재하는 경우와 그렇지 않은 경우를 나누어서 모델링하며, 자세한 것은 아래에서 설명한다.

1) Inner-composite SwC case (RteApp Type1)

Fig. 1의 RteApp Type1 과 같이 composite SwC 내부에서 atomic SwC 간의 정보교환이 이루어지는 경우를 Inner-composite SwC case 라고 정의한다. 정보 교환하는 대상들이 모두 모델 내부에 명시적으로

형상화 되어 있기 때문에 Simulink의 signal object를 이용하여 모델링이 가능하다. 코드 생성 시, Rte-Application 모듈의 인터페이스 함수 호출은 signal object의 storage class 설정을 통해 구현한다.

2) Inter-composite SwC case (RteApp Type2)

Fig. 1에서 RteApp Type2라고 도시하고 있는 것과 같이 composite SwC 사이에 존재하는 인터페이스를 Inter-composite SwC case라고 정의한다. Inner-composite SwC case와는 달리 정보 교환하는 대상이 명시적으로 모델에 드러나지 않기 때문에 RteHal 인터페이스와 마찬가지로 custom 블록(RteApp interface block)를 개발하여 모델링 할 수 있다. Fig. 5에서 도시하고 있는 바와 같이 인터페이스 이름을 블록의 설정 파라미터로 입력하여 코드 생성 시 RteApplication 모듈에 미리 정의된 인터페이스 함수를 호출하도록 구현하였다.

3.3.3 Inter-runnable variable 인터페이스

Atomic SwC 내부에서 runnable 간의 정보교환 인터페이스를 Inter-runnable variable(IRV)이라고 부르며, Simulink의 signal object를 이용하여 모델링이 가능하다. 코드 생성 시, IRV를 포함하고 있는 atomic SwC의 모듈의 정적 변수로 구현이 되어야 하기 때문에 signal object의 storage class 설정을 통해 구현한다.

4. 소프트웨어 통합 방법

이 연구에서 소개한 모델링 기법을 이용하여 개발한 ASW의 통합 과정들을 Fig. 6에서 간략하게 도시하고 있다. Runnable 라이브러리들을 조합하여 모델링된 atomic SwC들은 composite SwC로 맵핑되는 Simulink 모델 파일들 안에 위치하게 된다. 각각의 모델 파일들은 MathWorks사의 Real-Time Workshop Embedded Coder(RTW-EC)를 통하여 atomic SwC 모듈들의 소스코드로 생성되며, 이렇게 자동으로 생성된 ASW의 코드들은 RTE 계층의 RteHal, RteApplication, RteScheduler 모듈 구현을 통해 미리 준비된 BSW와 통합되고, 빌드과정을 거쳐 ECU에 내장된다.¹⁶⁾

Fig. 6 Integration process

이 장에서는 RTE 계층을 이용하여 어떻게 통합하는지에 대하여 인터페이스 통합과 스케줄링 통합으로 나누어 설명한다.

4.1 인터페이스 통합

ASW와 BSW 간의 인터페이스 통합은 RTE 계층의 RteHal 모듈을 통해 이루어진다. AUTOSAR-Lite에서는 RteHal의 인터페이스 방식을 server/client 방식으로 제한하였기 때문에 Fig. 7의 예제에서 도식화 하여 보여주고 있는 것과 같이 함수 매핑을 통하여 구현한다.

ASW의 SwC 간의 인터페이스 통합은 RteApplication 모듈을 통하여 구현되며, Fig. 7에서 도시하고 있는 예제처럼 sender/receiver 방식의 정보 교환

Fig. 7 Example of RteHal, RteApplication module¹⁶⁾

방식을 이용한다. RteApplication 모듈 내부에 존재하는 정적 변수를 이용하여 SwC 간 비동기 정보 교환을 구현한다.

4.2 스케줄링 통합

BSW 계층에 위치하는 스케줄러를 이용하여, 설계자가 요구하는 타이밍에 runnable들을 실행하기 위해서는 스케줄링 정보에 대한 통합 과정이 필요하다. AUTOSAR-Lite 에서는 RTE 계층에 위치하는 RteScheduler 모듈을 통해 스케줄링 통합 작업을 구현하며, Fig. 8에 도시하고 있는 바와 같이 atomic SwC 모듈 내에 함수 형태로 생성된 runnable을 task에서 호출하는 형태로 구현한다.

Fig. 8 Example of RteScheduler module¹⁶⁾

4.3 빌드

앞서 설명한 RTE 계층의 구현을 통한 인터페이스 및 스케줄링 통합 과정이 끝나면 각각의 모듈들에 대해 컴파일과 링킹 과정을 통해 대상 프로세서에서 실행 가능한 이진 코드를 얻는 것으로 통합과정을 마무리 마친다. 이 연구에서는 프리스케일사에서 제공하는 32-bit 마이크로컨트롤러인 MPC5554와 컴파일러 및 링커를 이용하였으며, 통합 개발환경(Integrated development environment, IDE)으로는 이클립스 CDT¹⁷⁾를 사용하였다.

5. 사례연구: 승용디젤엔진 제어시스템 개발

앞서 제시한 개발 방법의 실효성을 검증하기 위

하여, 승용디젤엔진 제어시스템(Engine management system, EMS)의 ASW 개발에 적용하였다.

5.1 대상시스템

승용디젤엔진 제어시스템은 크게 연료 시스템과 흡배기 시스템으로 구성 되어 있다. 연료 시스템은 Fig. 9에서 도시하고 있는 것과 같이 연료탱크로부터 연료를 공급하는 저압 연료 펌프, 최대 1800 bar의 압력까지 형성하는 고압 연료펌프 및 커먼레일, 그리고 연료를 분사하는 인젝터들로 구성되어 있으며 커먼레일의 압력과 인젝터의 연료분사를 제어한다. 흡배기 시스템은 터보차저와 배기가스 재순환 시스템들로 구성되어 있으며, VGT의 베인과 EGR 밸브 리프트를 전자제어 함으로써 엔진의 과급 압력과 흡기 유량을 제어한다.

Fig. 9 Diesel engine control system

이 사례연구에서는 위에서 설명한 연료 시스템과 흡배기 시스템들의 제어를 위한 소프트웨어 개발을 목적으로 하고 있다.

5.2 승용디젤엔진 제어 소프트웨어 개발

승용디젤엔진 제어를 위한 ASW는 센서신호 처리, 액추에이터 구동, 운전모드 판단, 연료시스템 제어, 흡배기 시스템제어와 같은 기능들을 52개의 runnable들로 구현하였다. ASW의 복잡도가 높은 관계로 전체 구조를 구체적으로 도시하기 어려워 Fig. 10의 상단부에서 도시하고 있는 것과 같이 composite SwC 모델(Sys_Sensor, Sys_ModeCtl, Sys_InjSys, Sys_AirSys, Sys_Actuator) 들로 표현한 상위계층의 구조

Fig. 10 Development results of the case study

로 간략화 시켜 표현하였다.

각각의 composite SwC 모델 파일들은 RTW-EC를 이용하여 Fig. 10에서 도시하고 있는 것처럼 8 개의 atomic SwC 모듈(Sensor, ModeCtl, FlSys, CrSys, InjCtl, EgrCtl, BstCtl, Actuator)의 소스 코드들로 생성되었으며, 총 52개의 runnable 들이 함수 형태로 SwC 모듈에 포함되어 있다. 각각의 runnable들은 off-line simulation 과 hardware-in-the loop simulation 과정을 통해 기능성 검증을 수행하였다. 4장에서 소개한 방법을 기초로 구현한 RTE 계층을 이용하여 미리 준비한 BSW 모듈들을 ASW의 SwC들과 통합하였으며, 그 결과물을 EMS의 프로세서에 탑재하였다.

5.3 엔진실험 검증

개발한 ASW의 실험 검증은 Photo 1에서 도시하고 있는 엔진동력계 실험환경에서 수행하였다. Fig. 11은 실험환경 구성을 도식화 하여 보여주고 있는 것으로 대상 엔진의 회전속도와 부하와 같은 시험 조건들을 제어할 수 있는 동력계 제어시스템과 EMS 내부의 연산결과들을 실시간으로 측정하고 보정하기 위한 표준 보정 프로토콜 기반 인터페이스 장비와 소프트웨어들로 구성되어 있다.

Photo. 1 Experiment environment

Fig. 11 Experimental setup

Table 2 Summary of the evaluation list

Subject	Evaluation list
BSW	Sensor signal processing function
	actuator driving function
	Communication function
	Engine position identification function
	Injector command signal generation function
ASW	Mode control function
	Common-rail pressure control performance
	Injection control performance
	Starting control performance
	Idle-speed control performance
	Boost pressure control performance
	Mass air flow rate control performance
	Diagnosis function

개발한 EMS의 SW들을 평가하기 위하여, 다양한 평가항목 및 기준들을 설정하여 엔진 실험 검증을 수행하였으며, Table 2에서는 평가항목 및 기준들에 대해 간략히 요약하였다. 대표로 연료시스템과 흡배기 시스템의 제어 성능들을 확인하기 위하여 Fig. 12에서 도시하고 있는 New European Driving Cycle (NEDC)을 기준으로 엔진동력계 시험을 수행하였다. Fig. 13은 연료시스템의 커먼레일 압력제어 실험 결과를

Fig. 12 NEDC Test cycle

Fig. 13 Experiment results of common-rail pressure control

목표/측정 값과 제어 오차로 구분하여 도시하고 있다. 시동, 아이들, 도심 및 고속 주행과 같은 다양한 운전조건들이 포함되어 있는 NEDC의 엔진회전속도와 부하조건을 Fig. 11에서 도시한 엔진동력계 시스템을 이용하여 모사 하였을 때, 커먼레일 압력 목표 값의 추종 성능을 확인할 수 있었다. 같은 방식으로 흡배기 시스템의 과급압력과 흡입공기유량 실험 결과들을 Fig. 14와 Fig. 15를 통해 도시하였다. 각각의 결과들에서 확인할 수 있듯이 전체 주행 사이클 동안 목표 값들을 정상적으로 추종하는 제어 기능성들을 확인할 수 있었으며, 이를 통해 이 연구에서 제안한 ASW 모델링 및 소프트웨어 통합 방법론이 차량용 전자제어시스템 개발에 효과적으로 적용될 수 있음을 확인할 수 있었다. 향후 제어성능 개선을 위하여 이 사례연구에서 개발한 시스템을 효과적으로 사용할 수 있을 것으로 평가된다.

Fig. 14 Experiment results of boost pressure control

Fig. 15 Experiment results of mass air flow control

6. 결 론

이 연구에서는 AUTOSAR와 같은 표준 소프트웨어 아키텍처를 대상으로 하는 모델기반 차량용 임베디드 소프트웨어 개발 방법론에 대하여 제시하였다. AUTOSAR의 주요 특징들을 반영한 경량화 소프트웨어 아키텍처인 AUTOSAR-Lite를 바탕으로 Simulink를 이용한 어플리케이션 소프트웨어모델링 및 통합 방법들을 제시하였다. 이 연구에서 제시한 방법들을 승용디젤엔진 제어시스템 개발에 적용하였으며, 다음과 같은 결론들을 얻을 수 있었다.

- 1) Matlab/Simulink로 일원화 된 어플리케이션 소프트웨어 모델링 및 통합기법 제시를 통하여, AUTOSAR 표준을 위한 모델기반 소프트웨어 개발의 효율성을 향상시켰다.

- 2) 이 연구에서 제시한 어플리케이션 소프트웨어 모델링 기법은 AUTOSAR-Lite 뿐만 아니라 AUTOSAR와도 쉽게 호환이 가능하기 때문에 상용 아키텍처 설계 도구와도 쉽게 통합이 가능한 장점을 지니고 있다.
- 3) AUTOSAR 표준과 호환 가능한 모델기반 소프트웨어 설계 기법 및 사례를 성공적으로 보여줌으로써, AUTOSAR 표준의 도입에 필요한 노력과 시간을 절감할 수 있을 것으로 기대한다.

후 기

이 연구는 지식경제부와 한국산업기술재단의 전략기술인력양성사업과 지식경제부 산업원천기술개발 사업의 일환으로 수행된 연구결과(No. 10039673)이며, 2011년도 교육과학기술부의 재원으로 한국연구재단의 지원을 받아 수행된 연구이다(No. 2011-0017495). 또한 지식 경제부 에너지자원기술개발사업의 일환(2006ETR11P091C)으로 수행되었다.

References

- 1) N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, "Trends in Automotive Communication Systems," Proceedings of the IEEE, Vol.93, pp.1204-1223, 2005.
- 2) M. Broy, I. Kruger, A. Pretschner and C. Salzmann, "Engineering Automotive Software," Proceedings of the IEEE, Vol.95, pp.356-373, 2007.
- 3) J. Ma, J. Youn, M. Shin, I. Hwang and M. Sunwoo, "Integrated Development Environment from Modeling to Implementation for Automotive Real-time Embedded Control System," Int. J. Automotive Technology, Vol.7, pp.335-341, 2006.
- 4) K. Lee, S. Hong, S. Oh and W. Lee, "A Study of Model Based Automotive Electronic System Development Process," Fall Conference Proceedings, KSAE, pp.1480-1485, 2006.
- 5) Matlab/Simulink, Available at <http://www.math-works.co.kr>, 2012.
- 6) ASCET, Available at <http://www.etas.com>, 2012.
- 7) W. Yoo, J. Park, J. Yoo, S. Kim and S. Hong, "AUTOSAR-based Software Architecture for Automotive Systems," Fall Conference Proceedings, KSAE, pp.60-65, 2006.
- 8) H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Maté, K. Nishikawa and T. Scharnhorst, "Automotive Open System ARchitecture - An Industry-Wide Initiative to Manage the Complexity of Emerging Automotive E/E-Architectures," SAE 2004-21-0042, 2004.
- 9) G. Park, S. Lee, D. Kum, C. Choi, W. Lee and W. Won, "Development of Software Component for EPS System Based on AUTOSAR," Annual Conference Proceedings, KSAE, pp.1867-1873, 2009.
- 10) R. Yerushalmi and R. A. Felice, "Implementing AUTOSAR Atomic Software Components Using UML/SYSML in C," SAE 2010-01-026, 2010.
- 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 2007-01-0507, 2007.
- 12) U. Freund, V. Jaikamal and J. Löchner, "Multi-level System Integration of Automotive ECUs based on AUTOSAR," SAE 2009-01-0918, 2009.
- 13) G. Sandmann and R. Thompson, "Development of AUTOSAR Software Components within Model-based Design," SAE 2008-01-0383, 2008.
- 14) Davinci Developer, Available at <http://www.vector.com/>, 2012.
- 15) SystemDesk, Available at <http://www.dspace.com/>, 2012.
- 16) K. Lee, I. Park, M. Sunwoo and W. Lee, "AUTOSAR-ready Light Software Architecture for Automotive Embedded Control Systems," Transactions of KSAE, To be published.
- 17) Eclipse C/C++ Development Tool, Available at <http://www.eclipse.org/cdt/>, 2012.