Esterel 기반 임베디드 소프트웨어의 신뢰성 향상을 위한 개발 기법

양진석° 심재환 김진현 강인혜* 최진영
고려대학교 컴퓨터학과 정형기법 연구실
*서울시립대 공과대학 기계정보 공학과
{jsyang°, jhsim, jhkim, choi}@formal.korea.ac.kr
*inhye@uos.ac.kr

A Development Methodology for Reliability Improvement of ESTEREL based Embedded Software

Jin-Seok YANG° Jae-Hwan SIM Jin-Hyun KIM In-Hye KANG* Jin-Young CHOI
The Computer Theory and Formal Methods Lab, Dept. Computer Science, Korea University
*Dept. of Mechanical and Information Engineering, University of Seoul

요 약

본 논문은 정형 명세 언어인 Esterel이 가지는 취약점을 보안하기 위해 Safety-Critical Aided Development Environment를 추가로 활용하여 신뢰성 있고 안전적인 임베디드 제어 소프트웨어 개발을 위한 기법을 제시하고 있다. 그 뿐만 아니라 제시한 기법을 이용하여 개발된 간단한 자동 감속 제어 소프트웨어를 인터페이스와 기능 부분에 대해서 각각 확인 및 검증을 수행한 후 임베디드 시스템인 레고 마인드 스톰으로 제작된 차량 로봇에 탑재한 후 실험을 하고, 실험을 통해 기존 개발 기법과의 차이점을 분석한다.

1. 서 론

오늘날 임베디드 시스템은 일상생활 속에 사용되는 가전제품에서부터 고안전성(Safety-Critical)을 요구하는 비행기나 원자력 발전소 계측장비까지 다양한 분야에서 사용이 되고 있다. 그리고 좀 더 좋은 품질과 안전성을 가진 임베디드 시스템 소프트웨어를 개발하기 위해 다양한 연구과 개발 도구와 기법들이 제시되고 있다. 이런 도구와 기법들은 시스템에 따라 달라지는 개발환경에 따라 길어지는 소프트웨어의 개발주기(Time-to-Market) 및 개발된 소프트웨어의 확인(Validation)과 같은 문제점들을 해결하기 위해 시뮬레이션 도구나 자동 코드 생성과 같은 기법을 제공하고 있다.

이 논문에서 다루는 Esterel 역시 임베디드 소프트웨어를 개발을 위해 1980년대부터 많이 사용되어온 기법 가운데 하나이며, 특히 제어 소프트웨어에 대해서 직관적이면서도 지배적인 측면에서의 명세를 할 수 있다는 장점을 가지고 있다. 또한 명세된 모델에 대한 검증 및 확인 작업을 위해 XES와 XEVE같은 도구를 지원해 주며, 모델로부터 자동으로 C소스 코드를 생성할 수 있다는 장점 역시 가지고 있다.[1] 하지만 이런 장점에도 불구하고 오늘날 임베디드 소프트웨어 개발에 가장 많이 사용되는 C언어에 비해 표현력이 부족하여 복잡한 제어 소프트웨어의 행위(Behavior)를 모델링하는 작업에 어려움이 있다. 이런 약점을 보완하기 위해 Esterel에서는 외부 C함수 호출을 허용하고 있지만, Esterel이 지원하는 XES나 XEVE와 같은 도구들이 외부 C함수에 대한 검증 및 확인 작업을 지원하지 못하는 관계로 안전과 신뢰성을 중시하는 임베디드 소프트웨어를 개발하기에 부족한 점이 있다. 본 논문에서는 이러한 Esterel의 약점을 보안하고 Esterel을 이용하여 신뢰성과 안전성을 요구하는 임베디드 시스템에 사용되는 소프트웨어를 개발할 수 있도록, Safety-Critical Aided Development Environment (SCADE)를 같이 사용하여 임베디드 제어 소프트웨어를 개발하는 기법을 제시한다.

이 논문은 2장에서 Esterel과 SCADE를 함께 활용하는 기법에 대해 설명하고, 3장에서는 본 논문에세 제시한 기법으로 개발된 간단한 자동 감속 제어 소프트웨어를 레고 마인드스톰(LEGO Mindstorms)로 제작된 차량로봇을 이용하여 실험하고, 기존 Esterel만을 이용한 개발 기법과의 차이점을 설명한다. 마지막 4장에서 실험과정을 통해 도출된 기법의 장단점에 대해 설명을 하고 끝을 맺는다.

2. 개발 기법

임베디드 시스템에 사용되는 반응형 소프트웨어의 경우, 임베디드 시스템의 운영체제나 하드웨어간의 인터페이스가 존재하고, 인터페이스를 통해 외부로부터 신호를 받아들인다. 그리고 입력으로 들어오는 신호에 따라 어떤 반응(Reaction)을 할지 결정하고 실제로 그 기능을 수행한다. 이런 특성을 통해 임베디드 제어 소프트웨어를 인터페이스부(Interface Part)와 기능부(Function Part)로 구분 할 수 있다. 그리고 개발되어지는 소프트웨어의 신뢰성과 안전성을 높이기 위해 정형 명세 언어인 Esterel과 SCADE를 이용하여 인터페이스와 기능 부분에 대해서 각각 다음과 같은 부분을 모델링을 한다.

Interface Part : 소프트웨어 입/출력 인터페이스 설계 및 소프트웨어 지배적인 입장에서의 입력에 대한 소프트웨어 출력 및 필요한 기능의 사용여부에 대한 논리적인 부분을 명세.

Function Part : 인터페이스 명세에서 사용된 *function* 이나 *procedure*를 통해 호출 될 외부 함수나 프로시져에 대한 기능을 명세 SCADE를 이용하여 명세.

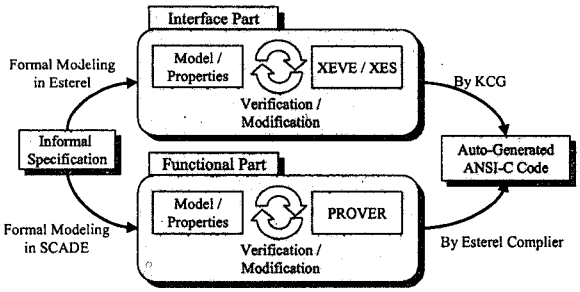

그림 1 임베디드 제어 소프트웨어의 개발 기법

즉, [그림 1]과 같이 임베디드 제어 소프트웨어 개발 과정에서 모델을 인터페이스부와 기능부로 나누어 특성에 맞는 도구를 이용하여 모델을 개발하고 코드를 생성한 후 코드를 통합한다.

임베디드 소프트웨어 개발 기법으로 Esterel과 함께 SCADE를 사용하여 다음과 같은 장점을 가진다. 첫째, 개발 소프트웨어를 검증 하고 확인 할 수 있도록 시뮬레이션 도구와 검증 도구를 지원한다. 이런 도구를 이용하여 소프트웨어가 반드시 만족해야 하는 특성(Property)에 대해서 검증을 수행하고, 그 결과에 따라 모델을 수정 할 수 있다. 둘째, 모델로부터 구현 코드를 자동으로 생성 시킬 수 있다. 검증 작업을 마친 모델로부터 자동으로 코드를 생성함으로써 사람의 개입으로 구현과정에서 발생하는 오류를 줄일 수 있다. 특히 SCADE의 경우, 유럽 항공분야 소프트웨어 기준인 DO-178B의 A등급에 해당되는 코드를 생성 할 수 있으며, 자동으로 생성된 코드의 경우 별도의 유닛 테스팅 과정 없이 사용 할 수 있다는 장점이 있다.[2] 마지막으로, SCADE로부터 자동으로 생성된 함수를 [그림 2]에서처럼 Esterel에서 호출하는 외부 함수로 사용함으로서, Esterel이 외부 함수에 대한 검증을 지원하지 못하는 약점을 보완해 줄 수 있다.

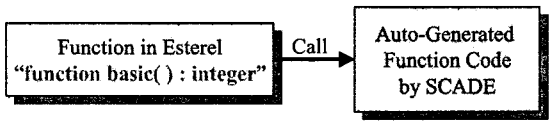

그림 2 Esterel와 SCADE의 관계

3. Case Study

3장에서는 2장에서 기술한 개발 기법을 이용하여 레고 마인드스톰 RCX에 탑재될 수 있는 간단한 자동 감속 제어 임베디드 소프트웨어개발 과정과 실험 결과에 대하여 설명한다.

3.1 차량로봇 제작 및 요구사항

실험에 사용되는 차량로봇은 프로토타입(Prototype) 용도로 비교적 손쉽게 구현 할 수 있는 장점을 가진 레고 마인드스톰을 사용하여 구현 하였다. 레고를 이용한 차량 로봇의 구현은, 앞에서 언급한 장점 이외에도, 레고 마인드 스톰의 RCX에 BrickOS를 사용함으로써 Esterel로 명세 되어진 소프트웨어를 손쉽게 임베디드 환경에 동작시켜 볼 수 있다는 장점을 가지고 있다.[3] 실험에 사용된 차량로봇은 하나의 감광(Light)센서를 입력으로 가지고, 두개의 모터를 출력으로 가지고 있으며, 개발된 소프트웨어는 다음과 같은 자연어 요구사항을 만족 한다.

a) 실험에 사용된 차량 로봇을 일종의 무인 차량으로 간주를 하고 자동 감속 제어 소프트웨어는 주어진 제동거리에 정확하게 차량을 정지시켜야 한다.
b) 차량 로봇은 일정한 속도로 진행을 하다가 감광센스가 바닥에 검은색 라인을 인식하는 순간부터 감속을 시행한다.
c) 인터페이스 입/출력에 대한 처리는 0.1초마다 이루어진다.
d) 감속이 시작되면 감속은 0.1초마다 이루어지며, 주어진 제동거리에 알맞게 차량을 정지시키기 위해 반드시 속도를 점차적으로 서서히 감속해야 한다.
e) 차량 로봇은 차량의 초기 속도와 제동거리를 초기값으로 가진다.

제동거리에 따른 제작된 로봇의 감속을 제어하는 기능을 구현하기 위해서는 레고 스텝 모터 계수에 따른 차량로봇의 단위시간당 이동거리($d = st, t = 0.1s$)를 계산하기 위해 모터 속도 계수(0부터 255)에 따른 차량 속도를 몇 번의 실험을 통해서 구하였다. 하지만 실험 데이터로부터 모터 계수 전체구간에 적용 할 수 있는 정확한 실험식이 산출되지 않아, 모터 속도 계수에 대한 구간을 구분하여 각 구간별로 실험식을 산출하였다.

a) Motor >= 250인 경우
$$s = 13.4$$
b) Motor < 250 and Motor >= 100인 경우
$$s = -0.0002x^2 + 0.1015x + 0.75$$
c) Motor < 100 and Motor >= 0인 경우
$$s = 0.0002x^2 + 0.0764x - 0.24$$

실험식에 의해 [그림 3]처럼 유도된 차량 속도그래프에서는 모터 계수 구간 0부터 100사이에 급격한 차량 속도 증가를 보이고 있지만, 실제적으로 모터 속도 계수가 45이하인 경우에는 차량의 무게로 인하여 차량이 움직이지 않는다. 감속을 제어하는 소프트웨어를 개발 할 때에는 실험식을 통해서 계산된 모터 속도 계수가 45이하인 경우에는 차량이 정지한 것으로 간주하였다.

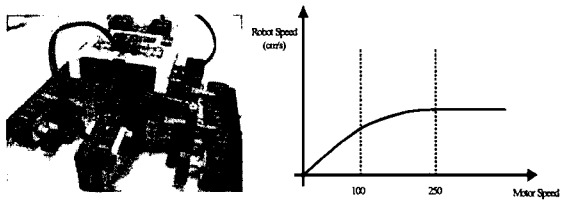

그림 3 차량 로봇과 구간별 속도 그래프

3.2 정형명세를 통한 모델링 및 검증

앞 절에서 살펴본 자연어 요구명세 b)와 c)에 따라서, 소프트웨어와 레고 RCX 사이의 입력 인터페이스는 외부 감광 센스와 RCX에서 발생하는 tick만 존재하고 이 두 입력 인터페이스를 통해 소프트웨어의 동작이 결정되어 지는 것을 알 수 있다.

그 결과 차량 로봇에 대한 인터페이스는 다음과 같이 직관적이고 간단하게 Esterel로 표현될 수 있다.

```
loop
        if ?LIGHT_2_VALUE <= Start_Light_Value
          and Robot_Status = 0 then
                emit START_BREAK;
                Robot_Status := 1;
        end if;

        if Robot_Status = 1 then
                call Calc_Speed(Motor_Speed, Distance)();
        end if;

        emit MOTOR_A_SPEED(Motor_Speed);
        emit MOTOR_C_SPEED(Motor_Speed);
each 10 tick;
```

Esterel "*loop ... each 10 tick*" 구문을 통해 시간적인 자연어 요구 사항 c)을 명세하였으며, *"LIGHT_2_VALUE"* 신호와 상태 변수를 이용해 차량의 상태를 변화 시켜 외부 함수를 호출 할 수 있게 명세 되었다. 그리고 모델이 자연어 요구사항 b)를 만족하는지 확인하기 위해 시뮬레이션 도구 XES를 이용하여 아래의 검증 특성을 확인하였다.

(?LIGHT_2_VALUE <= 10 and Status=0 → AG(START_BREAK))

자연어 요구사항 a)와 d)는 소프트웨어의 기능에 대한 요구이므로 3.1절에서 언급한 차량 로봇 속도 실험식 등을 이용하여 [그림 4]와 같이 모터 속도 계수(Motor_Step)와 제동거리(Distance)를 입력으로 받아 감속된 모터 계수와 줄어든 제동거리를 출력으로 가지는 SCADE 블록 다이어그램(Block Diagram)으로 모델링 할 수 있다. [그림 4]에 나타난 블록 다이어그램 모델은 모델의 최상위 계층이며 각각 하위 블록 다이어그램을 가지고 있다.

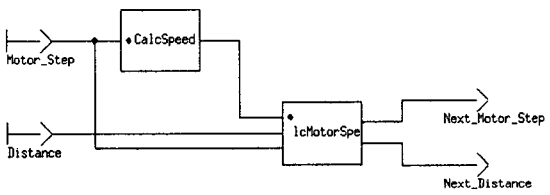

그림 4 SCADE를 이용한 기능부 모델링

SCADE로 작성된 모델은 Prover를 통해 검증 특성을 만족하는지 검증 작업을 할 수 있다. 실험에서는 모델이 자연어 요구 명세 d)를 만족하여 로봇 차량을 주어진 제동거리에서 서서히 정지 시킬 수 있는지를 확인하기 위해 "제동거리가 40cm 일 때, 모터 계수의 감속량이 20보다 작아야 한다." 라는 검증 특성을 적용하였다. 이런 자연어로 된 검증특성을 SCADE로 검증하기 위해 아래와 같은 논리식으로 변경을 하고,

(Distance ≤ 40 → Next_Motor_Step < 20)

논리식을 [그림 5]에 같은 블록 다이어그램으로 다시 작성할 수 있다. 그리고 Prover를 이용하여 "Result" 값(참/거짓)의 결과를 통해 모델이 특성을 만족하는지 여부를 확인하였다. 검증 결과 제동 결과 40cm에 대해서는 요구사항 d)가 만족함을 알 수 있었다.

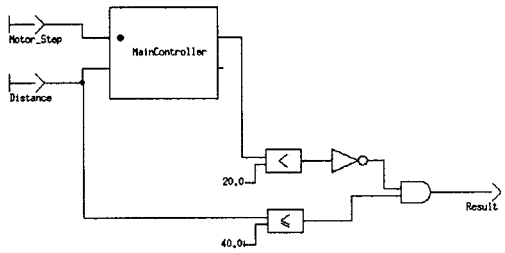

그림 5 SCADE 모델의 검증

3.3 실험결과

SCADE로부터 함수를 자동 생성하고, 자동 생성된 함수의 이름에 맞게 Esterel에서 사용되는 외부 호출 함수 이름의 수정과 SCADE 검증에 사용된 조건으로 실제 실험을 하기 위해 초기 속도 250에 제동 거리를 40cm으로 설정한 후 역시 코드를 자동생성 하였다. 크로스 컴파일과정을 거쳐 레고 차량 로봇에 소프트웨어를 탑재하고 몇 번의 실험 결과 차량은 제동을 시작한 후 주어진 제동 거리 40cm에서 전 후 0.5cm 미만의 오차를 두고 차량이 멈추는 것을 확인 할 수 있었다. 그리고 제동 시작 지점을 알리는 표시가 없는 경우 제동이 발생하지 않음을 알 수 있었다.

4. 결 론

본 논문에서는 두 개의 정형기법 도구인 Esterel과 SCADE를 활용하여 임베디드 제어 소프트웨어를 개발하는 기법을 제시하였다. 기존 Esterel로 작성된 소프트웨어에서 사용된 외부 C 함수의 경우 특별한 검증과정이 없어 안전성이 중시되는 임베디드 시스템에 사용될 수 없었으나, SCADE를 통해 검증이 완료된 함수를 Esterel의 외부 함수로 사용함으로써, 소프트웨어 전반적인 신뢰성을 높일 수 있었고, 그 활용도를 높일 수 있음을 실험을 통해 알 수 있었다. 기존 Esterel로만 작성된 소프트웨어의 경우, 거리나 시간에 관련된 로봇의 행위에 대해 반복적인 시행착오를 통해 개발되어 졌으나, 본 논문에서 제시된 기법을 통해 주어진 조건에 좀더 정확하고 신뢰성이 행위를 할 수 있는 임베디드 소프트웨어를 개발 할 수 있었다. 하지만 큰 소프트웨어 대한 실험을 하지 못한 관계로 앞으로 좀 더 다양하고 복잡한 제어 소프트웨어의 설계를 통해 문제점 파악 및 보안을 해 나아가도록 하겠다.

5. 참고문헌

[1] 강인혜, 양진석, "초보자를 위한 Esterel 프로그래밍", 홍릉과학출판사, 01. 2005

[2] G.Berry, "The Effectiveness of Synchronous Languages for the Development of Safety-Critical Systems", Esterel Technologies, 2003