기능 요구사항 시뮬레이션을 이용한 임베디드 시스템 및 소프트웨어의 요구사항 검증

임재훈
고려대학교 컴퓨터정보통신대학원 소프트웨어공학과
e-mail: huroman@korea.ac.kr

Functional requirements simulation for requirements verification of embedded system and software

Jae Hoon Lim
Dept. of Software Engineering, Korea University Graduate School of Computer & Information Technology

요 약

기술발전에 따라, 보다 다양한 분야와 영역에 다양한 형태의 임베디드 시스템이 사용됨에 따라, 그에 대한 신뢰성과 안전성에 대한 요구가 증가하면서, 하드웨어 뿐만 아니라 소프트웨어까지도 포함한 부분에 대한 철저한 명세와 그에 따른 검증이 요구되고 있어, 임베디드 시스템 및 소프트웨어의 요구사항 검증을 위해 요구사항 시뮬레이션이라는 기법을 적용하고, 그 효용성을 확인하고자 한다

1. 서론

컴퓨팅 기술과 인터넷의 발전에 따라, 보다 다양한 영역에서 기존의 아날로그 기반 시스템들이 디지털 시스템으로 대체되고 있다. 다만, 고신뢰성/고안전성을 요구하는 분야에서는 이에 대한 보수적인 입장이 있었으나, 최근 들어 이 역시 빠르게 변하고 있다. 이에 따라 임베디드 시스템에 대한 철저한 설계와 검증이 요구되고 있다. 특히 시스템과 소프트웨어에 대한 요구사항은 설계와 검증을 위한 첫 단추임에도 불구하고, 현장에서의 엔지니어링 수준은 아직 갈 길이 멀다. 이에 현장에서 보다 손쉽게 적용할 수 있도록 도구를 활용한 요구사항 시뮬레이션 기반의 요구사항 검증을 위한 방법을 적용하고 이를 보다 용이하게 적용할 수 있는 방안을 모색한다.

2. 본론

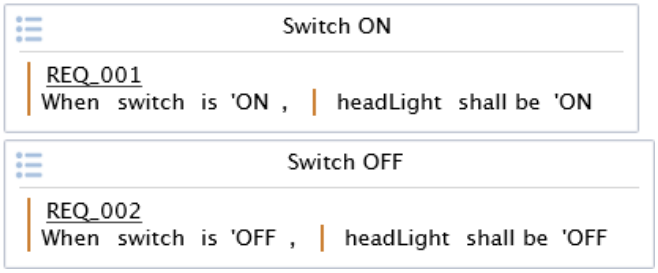
요구사항 검증을 통해 기능 요구사항을 보다 명확하게 하기 위해서, 요구사항이 가지고 있는 기능적 모호성 및 모순점을 파악하려 여러가지 기법을 사용하였으나, 특히 요구사항 시뮬레이션을 이용하여 입력에 근거한 요구사항의 동적 특성을 파악할 수 있는 방법을 찾아보았다. 특히 Argosim 社의 STIMULUS 를 이용한 시뮬레이션에서 특히 의미있는 결과를 찾아서 이를 정리하고자 한다. 우선 고수준의 시스템 요구사항을 정의하기 위한 인터페이스를 정의한다. 여기서는 차량의 Headlight (전조등) 제어 시스템을 그 예로 들고자 한다.

Name	Type	Default Value	Enum Encoding	Comment
switch	Enum	'OFF		switch status
'AUTO			0	
'ON			1	
'OFF			2	
headLight	Enum	'OFF		head light status
'ON			0	
'OFF			1	
lightIntensity	real			Light intensity

[그림 1] Glossary 의 정의

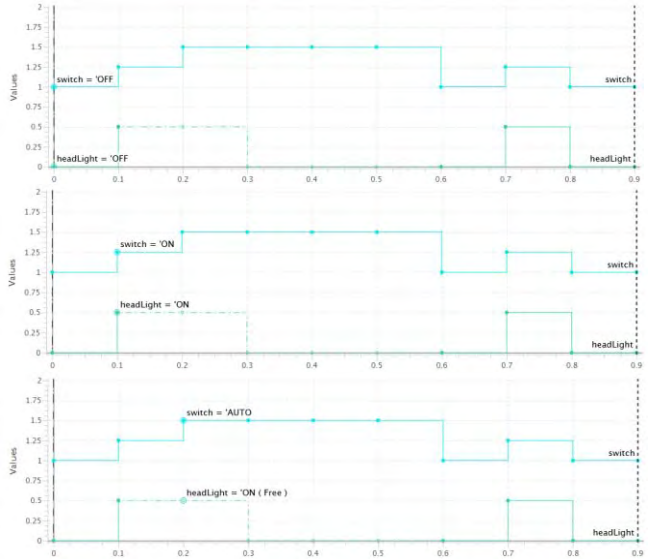
차량의 Headlight 의 점멸을 제어하는 Switch (입력)은 ON, OFF, AUTO 의 3 가지 형태의 입력값을 가지며, AUTO 의 경우, LightIntensity (입력, 외부 밝기(0 에서 1 사이의 실수)의 값에 따라, 자동으로 HeadLight (전조등 점멸, 출력)의 출력을 결정하는 예를 들었다.

우선 다음과 같은 2 개의 기본적인 요구사항을 정의하였다.



[그림 2] 기본적인 요구사항의 정의

이 기본적인 요구사항을 Switch 입력값과 연결하여 우선 시뮬레이션 해 보았다. (10 회, Random Value)



[그림 3] Switch 와 Headlight 간의 시뮬레이션

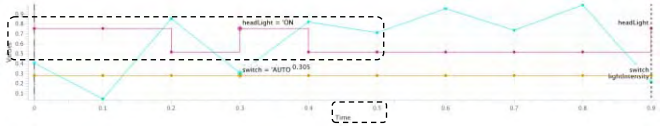
Switch 값이 ON 또는 OFF 인 경우 Headlight 는 요구사항에 부합하는 값이 나온다. Swtich 값이 AUTO 인 경우에는 시뮬레이션이 점선으로 표시되며 있는 이 경우(입력값)에 대한 요구사항이 없다는 것을 의미한다. (Missing Requirements)

AUTO 인 경우에 대해 집중적인 시뮬레이션을 위해 Switch 입력값은 AUTO 로 고정하였다. 외부 밝기가 60% (0.6) 미만이면 Headlight 를 켜고, 외부 밝기가 이 이상일 경우, Headlight 를 끄는 것으로 3 번째 요구사항을 아래와 같이 정의하였다.

When switch is 'AUTO ,
  When (lightIntensity is less than (60 %)),
    headLight shall be 'ON ...
  When (lightIntensity ≥ (60 %)),
    headLight shall be 'OFF

[그림 4] Switch:AUTO 요구사항의 정의

그리고 이를 10 회 시뮬레이션 하였다. 그 결과는 다음과 같다.



[그림 5] Switch:AUTO 의 시뮬레이션 (채터링 발생)

위 그림에서 보는 바와 같이, 해당 구간(검은 점선 사각형)에 있어 Headlight 가 짧은 시간 (0.5 초) 동안 점멸을 반복(2 회)하는 (채터링 현상) 이상 동작을 보이게 된다. 이로부터 3 번째 요구사항은 개선되어야 함을 알 수 있다.
또한, 외부 밝기의 단기적이며 (1 초 이내) 불연속적인 변동이 있을 경우 (터널 진입, 일몰시 가로등의 영향 등, 이 경우에도 채터링 현상이 발생 가능)에 대한 요구사항이 부재함을 알 수 있다.

이를 수정하기 위해, 히스테리시스 제어를 포함하는 4 번째 요구사항을 만들었다. 입력이 충분히 유지되는지를 확인하기 위한 시간 지연도 포함시켰다. 4 번째 요구사항은 아래와 같다.

When switch is 'AUTO ,
  When lightIntensity < 60 % has been true for more than 1[second] ,
    headLight shall be 'ON
  When lightIntensity > 70 % has been true for more than 1[second] ,
    headLight shall be 'OFF

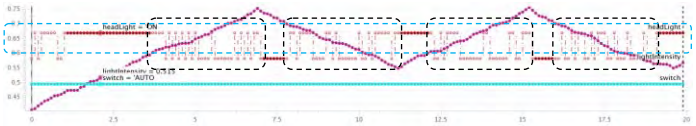
[그림 6] 히스테리시스 제어를 고려한 요구사항

그리고 비현실적인 외부조도의 급격한 변화를 줄이기 위해, LightIntensity 입력에 다음과 같은 제약을 부여하였다. 효과적인 시뮬레이션을 위해, 외부조도가 55%에서 75%사이에서 변동하도록 제약을 걸었다.

lightIntensity ∈ [0 , 1]
lightIntensity goes up and down respectively to 75 % and 55 %
(derivative of lightIntensity) ∈ [-0.1[1 / second] , 0.1[1 / second]]

[그림 7] LightIntensity 입력에 대한 제약조건 부여

이에 대한 시뮬레이션 결과는 다음과 같다. (200 회)



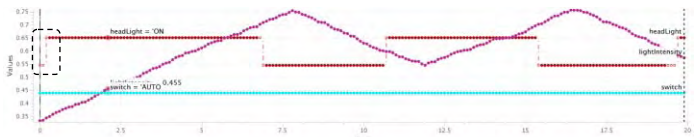
[그림 8] 요구사항 시뮬레이션의 결과 (200 회)

채터링 현상을 사라졌으나, 여전히 많은 Missing Requirements 구간 (검은 점선 사각형)을 발견할 수 있다. 시뮬레이션 결과로부터 우선 히스테리시스 구간 (파란 점선 사각형) 내에서의 요구사항이 불명확하다는 것을 알 수 있다. 그래서 아래와 같이 요구사항을 다시 정의하였다.

When switch is 'AUTO ,
  When lightIntensity < 60 % has been true for more than 1[second] ,
    headLight shall be 'ON
  When lightIntensity > 70 % has been true for more than 1[second] ,
    headLight shall be 'OFF
  When lightIntensity > 60 % and headLight was 'OFF ,
    headLight shall be 'OFF
  When lightIntensity < 70 % and headLight was 'ON ,
    headLight shall be 'ON

[그림 9] 히스테리시스 구간 내에서의 요구사항 추가

그 시뮬레이션 결과는 아래와 같다.(200 회)



[그림 10] 히스테리시스 구간 요구사항의 시뮬레이션

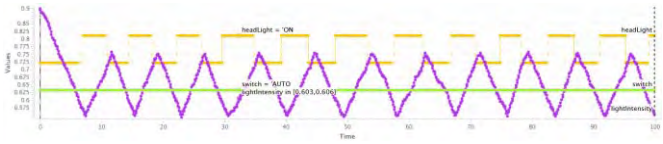
위의 시뮬레이션 결과로부터, 차에 시동을 걸어 Headlight 제어시스템이 처음 시작하는 초기 상태에서의 외부 밝기 값이 히스테리시스 제어 구간에 있을 때의 요구사항이 없다는 것을 확인할 수 있다.

이로부터 4 번째 요구사항의 결함을 조기에 식별하여 이를 반영한 5 번째 요구사항을 다시 정의하였다. 5 번째 요구사항은 초기 상태에서의 기본값을 다음과 같이 정의하였다.

```
When ( switch  is 'AUTO ),
    Initially
        If  lightIntensity  <  70 %  then
                headLight  shall be  'ON
        else
                headLight  shall be  'OFF
    afterwards
        When  lightIntensity  <  60 %  has been true for more than  1[second] ,
                headLight  shall be  'ON
        When  lightIntensity  >  70 %  has been true for more than  1[second] ,
                headLight  shall be  'OFF
        When  lightIntensity  >  60 %  and  headLight  was  'OFF  ,
                headLight  shall be  'OFF
        When  lightIntensity  <  70 %  and  headLight  was  'ON  ,
                headLight  shall be  'ON
```

[그림 11] 초기 동작을 고려한 요구사항

그 시뮬레이션 결과는 아래와 같다. (1000 회)



[그림 12] 초기 동작 고려한 요구사항의 시뮬레이션

시뮬레이션을 통해 요구사항에 만족하는 결과가 나왔기에 이를 바탕으로, 기존의 다른 요구사항과 함께 시뮬레이션 해 보고자 한다. (Switch ON, OFF 인 경우도 포함),

```
Switch is AUTO
When ( switch  is 'AUTO ),
    Initially
        If  lightIntensity  <  70 %  then
                headLight  shall be  'ON
        else
                headLight  shall be  'OFF
    afterwards
        When  lightIntensity  <  60 %  has been true for more than  1[second] ,
                headLight  shall be  'ON
        When  lightIntensity  >  70 %  has been true for more than  1[second] ,
                headLight  shall be  'OFF
        When  lightIntensity  >  60 %  and  headLight  was  'OFF  ,
                headLight  shall be  'OFF
        When  lightIntensity  <  70 %  and  headLight  was  'ON  ,
                headLight  shall be  'ON

Switch is ON or OFF
When ( switch  is 'ON ),     headLight  shall be  'ON
When ( switch  is 'OFF ),    headLight  shall be  'OFF
```

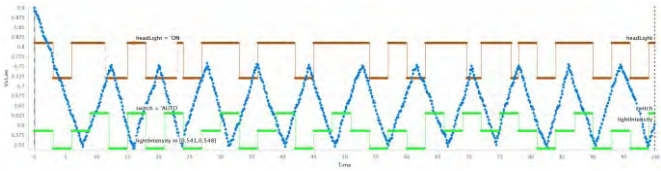
[그림 13] 전체 요구사항

그리고, 외부 스위치 값의 입력은 최소 3 초를 유지하도록 하였다.

```
switch  is stable for  3[second]
```

[그림 14] Switch 값에 제약 부여 (3 초 유지)

전체 요구사항의 시뮬레이션 결과는 아래와 같다. (1000 회)



[그림 15] 전체 요구사항의 시뮬레이션

3. 결론

요구사항 시뮬레이션이라는 다소 생소한 기법을 사용했지만, 프로젝트 후반부 (실차 테스트, 필드 테스트 등)에서 발견될 수 있는 여러가지 요구사항 결함 들을 조기에 식별할 수 있었고, 이를 통해 요구사항에 포함된 결함의 Early Detection 을 할 수 있기에, 요구사항 시뮬레이션을 활용함으로, 소프트웨어 개발 프로젝트를 성공적으로 진행할 수 있게 됨을 알 수 있었다.

참고문헌

[1] Ian Sommerville “Software Engineering”
[2] Jeremy Dick, Elizabeth Hull “Requirements Engineering”