정형검증도구를 통한 요구공학기술을 연계한 스마트 살충제의 요구사항 분석과 개념적 설계

박선영
건국대학교

dksldl@konkuk.ac.kr

Requirement analysis and conceptual design of Smart Insecticide with required engineering technology through formal verification tool

Sun Young Park
Konkuk Univ.

요 약

본 논문은 요구공학기술과 정형검증도구인 SMV 를 통해 제품의 개발 시에 발생 할 수 있는 문제점 등을 요구 사항 분석과 설계 단계에서 유추해 볼 수 있다는 것을 확인 하고자 하였다. 그를 위한 케이스로 스마트폰과 연계 가능한 자동/살충 포집 장치인 '스마트 살충제'의 요구사항 분석과 모델링, 발생할 수 있는 가상 시나리오를 기준으로 SMV 를 이용하여 시뮬레이션 하고 검증 하였다.

Abstract

In this paper, we tried to confirm that the problems that may occur during the development of the product through the requirements engineering technology and the verification tool SMV can be deduced from the requirements analysis and design stage. This case is simulated and verified using SMV based on the requirements analysis, modeling, and hypothetical scenarios of 'Smart Insecticide', an automatic / insecticidal device capable of linking with smartphone.

Ⅰ. 서 론

과학기술의 발달로 인해 다양한 기능을 하는 홈 케어 제품이 시장을 주도 하고 있다. 기존에 단순한 구조의 제품이 스스로 상황을 보고 상태를 변경하는 제품으로 변모 하고 있으며, 그에 따라 다양한 개발 및 테스트가 이루어 지고 있다. 요구사항분석과 설계는 개발 단계 이전에 이해관계자와 커뮤니케이션을 하는 것으로 프로젝트를 하는데 중요한 요소 중 하나이다.

이해관계자를 인식시키기 위한 다양한 방법이 있으나 이 중 요구 공학은 커뮤니케이션을 하는데 좋은 수단으로 이용되고 있다.

또한 정형검증도구를 이용하면 요구사항에 의해 구상된 객체의 모델링과 검증을 통해 상태의 전이로 인해 발생 할 수 있는 여러 상황을 시뮬레이션 해봄으로써 발생할 수 있는 여러 문제를 미리 이해 당사자들에게 이해 시키는데 도움이 될 수 있다.

이에 불편함에서 온 아이디어인 자동 살충/포집 장치인 '스마트 살충제'를 구상하고 요구 공학과 정형검증 도구를 이용 하여 검증을 수행 하였다.

Ⅱ. 관련 연구

2.1 요구공학(Requirements Engineering)

요구공학은 소프트 웨어 공학의 추상적 이고 논리적인 결과에 대해 개발자를 포함한 모든 이해 당사자들이 동일한 관점을 갖기 위해, 커뮤니케이션을 하기 위한 목적 이나 개발기준을 수립하기 위한 목적으로 발달해 왔다.

일부 연구[5] 에서는 '소프트웨어 요구 사항은 실제 요구 사항이 분명 하지 않은 상황에서 요구 사항으로 인해 설계 결정을 왜곡하는 경우가 발생할 수 있다' 라는 내용을 기술한 바도 있다. 그럼에도 불구하고 요구 공학은 단순히 소프트웨어를 구현하는 것 뿐 아니라 구현되지 않은 소프트웨어의 안전성을 검증하는 지표로도 지속적으로 활용 되고 연구되는 분야 중 하나이다.

요구 공학을 통해 다양한 사례의 케이스의 요구사항 도출 및 분석 모델링, 검증, 명세, 관리를 수행[2] 할 수 있으며 분야의 예시도 웹 응용 제품에서부터 자율주행 자동차에 이르기 까지 다양 하다.

본 연구에서는 요구 공학을 통해 기존에 만들어 지지 않은 제품의 요구 사항을 도출 하고 모델링 하여

테스트 시나리오 구성을 하는 일련의 단계를 통해 제품의 요구사항과 발생할 수 있는 시나리오를 도출 하여 필요성과 가능성을 검증 하였다.

2.2 정형 검증(formal Verification)

정형 검증은 정형적으로 명세된 시스템의 안정성을 증명하는 방법이다. 정형 검증에는 크게 자동 증명(theorem proving) 과 모델 검증(model checking) 이 있다. 모델 검증[3]은 시스템의 동작을 유한 상태 기계(finite state machine) 으로 명세하고 그 시스템이 만족 해야할 특성을 CTL[4] 이나 LTL[4] 같은 언어로 표현한다. 그 후에 해당 모델이 유한 상태 기계로 명세된 시스템에서 만족 하는지를 검사하는 방법이다.

2.3 CTL

CTL(Computation Tree Logic)[3] 은 유한 모델에서 임의의 상태가 주어진 논리식을 만족하는지를 효율적으로 검증하는 고정 특성을 갖추고 있고, 경로 지시자(path quantifier)에 시제 연산자(temporal logic)가 결합한 형태로 이루어진다. 시제 연산자에는 G(globally or invariantly),F(sometime in the future), X(next time), U(until)가 있다. 경로 지시자는 트리(tree)의 가지 구조(branching structure)를 표현하는 것으로, A(for all computation path) 와 E(for some computation path)가 있다.

2.4 NUSMV

NuSMV 는 유한 상태 시스템의 검증을 위한 소프트웨어 도구로 FBK-IRST 와 Carnegie Mellon University 가 공동으로 개발하였다.[1] BDD(Binary Decision Digrams)라는 논리 표현 기술로 각 단계의 검증을 표현한다. NuSMV 는 CTL 과 LTL 을 이용하여 SMVL 로 모델링한 표현이 시스템에 적합한지를 검증 하도록 하는 다양한 형식(traces, simulation, counterexample)을 제공한다.

Ⅲ. 케이스의 필요성과 요구사항 도출 과정

3.1 '스마트 살충제'의 필요성

초 여름부터 가을까지 성행하는 모기는 인간의 피를 빠는 과정을 통해 불쾌감 뿐 아니라 다양한 질병의 온상이 되기도 한다. 질병 관리 본부에 의하면 국내 모기에 의해 감염 될 수 있는 질병은 말라리아, 일본뇌염 외에도 최근에는 지카바이러스 등을 옮길 수 있는 것으로 보고 되고 있다. 살충제재의 논란은 이전부터 있어 왔고 일부 제품에 대해서는 살포용으로 금지 되는 등의 일이 있어왔다. 사람이 직접 손에 들고 살포용으로 사용 하는 것은 그 자체로 살충제제의 흡입위험이 도사리고 있다.

현재 일반 가정에서 상용화 되어 이용 하는 방식은 일반적으로 모기 기피(모기가 싫어하는 파장이나 냄새를 살포)와 모기 유인 후 처치가 있는데 모인 유인 후 처치의 경우 대체로 모기가 좋아하는 파장을 빛으로 발사한 후 모기가 모이면 상시로 켜져 있는 FAN 에 의해 흡입 되는 방식이다. 그 외에도 타이머만 탑재한 자동 분사 기능만을 가진 자동 살충 제품도 존재 한다.

최근 몇 년간 '스마트'를 내세우고 센서를 이용하여 동작 하는 여러 종류의 제품이 나오고 있으나 살충제의 경우는 대체로 위와 같은 상시로 동작하는 제품만 상용화 되어 있다. 이에 따라 유인이 잘 수행되었는지를 알 수 있는 센서를 통해 방제 작업을 인간이 하는 것과 유사하게 진행 할 수 있는 방법을 고안해보고자 한다.

3.2 요구사항과 객체 도출

'스마트 살충제'를 모델링 하기 위해 초기에 도출한 요구 사항은 다음과 같다.

- 곤충이 근처에 있다는 것을 인식할 수 있어야 함
- 노즐이 일정 기간 노후 되거나 하면 교체할 수 있도록 알려야 함
- 사람에게는 살충제가 뿌려지지 않도록 해야 함
- FAN 은 모기를 포집 할 시에만 동작 하고 상시로 동작하지 않아야 함

이후 요구 사항을 해소 하기 위해 자료 조사 등을 통해 기존 제품을 일부 참조하여 가상으로 구성한 객체는 Figure 1 과 같다.

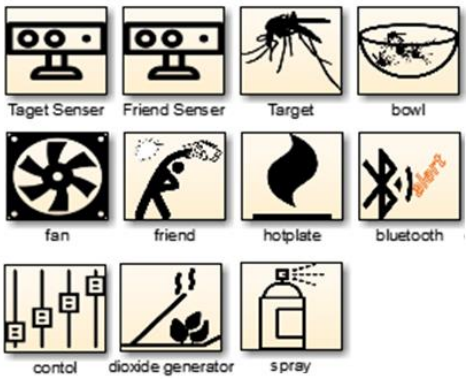

Figure 1 모델링을 위한 객체 선정

모델 체크용으로 센서 앞에 등장하거나 하는 부분의 시나리오를 수행 하기 위해 target(모기) 와 friend(사용자) 객체가 있으며 각 객체를 체크 하기 위한 센서와 센서의 on/off 여부를 파악하여 각 유인→살충→포집 의 3 단계를 수행하기 위한 기기들의 처리를 하기 위한 control 이 존재 하도록 하였다. 개체들 중 dioxide generator 와 bowl, Spray 등은 사용 범위가 한정된 객체로 사용량에 따라 사람이 채우거나 비우는 시나리오를 수행할 수 있도록 변수를 추가하였다.

SMV 상의 main 에 관련 객체의 상태를 넣을 경우 시뮬레이션 시 각 객체의 상태를 랜덤하게 변경 하며 각 개체 들이 수행 되면 상태가 변경되는 것을 traces 를 통해 확인 할 수 있으며 Figure 2 와 같이 수행 하였다. 모델을 원활히 변경 하기 위해 Eclipse 의 xtext plugin 2.11.0 과 NuSMV Tools 0.1.0 plugin 을 사용하였으며 traces 사용을 위해 NuSMV 2.5.4[1] 를 설치 하였다.

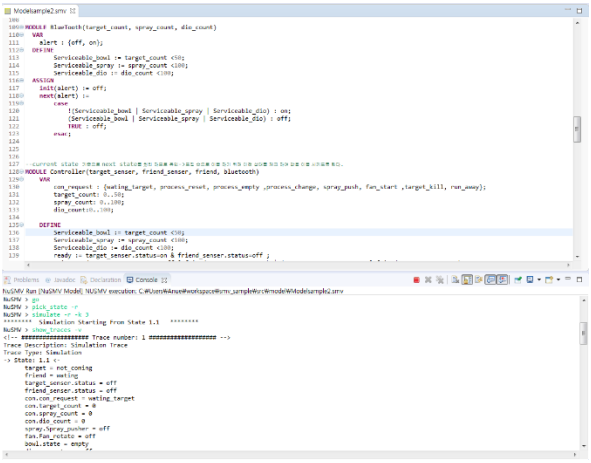

Figure 2 모델 작성과 상태의 전이 확인 예시

그리고 CTL 로 검증 항목을 작성하고 모델 체커로 수행 하여 control 과 객체들의 상태가 유기적으로 잘 작동하는지에 대해 확인 하였다. 작성 된 CTL 의 일부 예시는 다음과 같다.

- SPEC AG (friend_senser.status=on -> EX con.con_request!=spray_push) 사용자의 인식에 의해 스프레이를 뿌리지 않도록 함
- SPEC AG((spray_count>100) -> EF bluetooth.alert = on) 소모품이 닳으면 알림을 해야 함

이 외에도 모델 검증을 위해 다양한 CTL 을 작성 하고 모델 체커로 검증을 수행 하였다. 본 과정은 NuSMV 2.1 의 GUI edition 인 GNuSMV 로 검증 항목의 true/false 항목의 체크와 false 인 경우 관련 값의 traces 항목도 확인 할 수 있다. 예시는 Figure 3 과 같다.

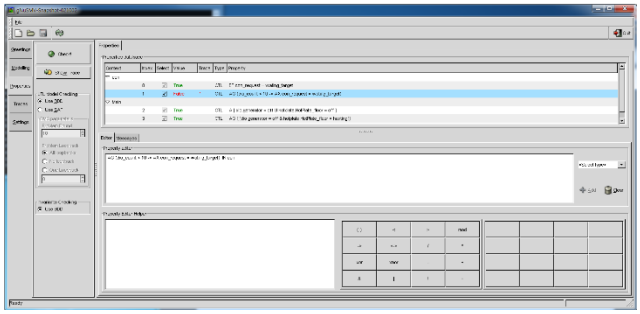

Figure 3 모델의 검증

그 결과 일부 검증이 false 로 표현되는 것을 발견 하였으며 그 검증과정을 true 로 처리 하기 위해 추가적인 요구사항이 필요함을 발견하고 그에 따라 모델을 수정 하였다 추가로 발생된 요구 사항은 다음과 같다.

- Control 상태 중 모기가 도망가는 상태가 고려 되어야 하며 run_away 상태로 명기하여 해당 상태일 때는 모두 일시 대기 하고 다시 target 을 기다리는 상태로 전이할 수 있도록 모델링을 수행하였다.
- SMV 특성상 state 에 대한 전이의 개념으로 시간의 경과를 측정 하나 대기 단계 등의 과정이나 실제로 dioxide generator 등의 경과에 대해서는 timer 등을 탑재 하여 별도의 state 로 전이되도록 만들어 처리 해야 실제 상황에 대해 반영될 수 있음을 알게 되었고, 타이머를 추가할 경우 구성할 수 있는 추가적인 기능적 아이디어(타이머를 통해 포집 시간과 포집 양, target 이 자주 등장하는 시간대를 알림할 수 있음) 도 도출 하였다.

이런 단계를 통해 NuSMV 등의 model checker 로 만들어지지 않은 제품의 요구 사항에 따라 객체를 선정하고 시나리오와 검증 항목에 따라 모델의 적합성을 시뮬레이션이 가능함을 확인 하였다.

IV. 결 론

소프트웨어 공학에서의 요구 공학의 제일 큰 장점은 구성되지 않은 제품의 시뮬레이션을 해봄으로써 개발 시 발생할 수 있는 문제점을 미리 도출해 보고 그를 현업에 적용 할 수 있다는 점이다.

실제 개발자가 아닌 이해 당사자들이 시뮬레이션 된 결과를 통해 구현단계에서 발생 할 수 있는 문제를 미리 파악하고 그에 맞춰 요구 사항을 수정 할 수 있도록 도움을 준다.

본 '스마트 살충제'의 케이스에서는 개별 모듈이 디테일 하게 구현된 것은 아니나 객체의 상태 변화를 통해 다른 객체에 주는 영향을 구현 단계가 아닌 모델 단계에서 파악해 볼 수 있었다. 또한 그 결과를 바탕으로 추가적으로 필요한 요구 사항을 도출 하고 기능의 확장을 이룰 수 있음도 파악할 수 있었다.

현업에서는 보통 요구 사항을 도출하고 그를 이해 당사자에게 이해시키기 위해서 구현을 먼저 수행하게 되고 이후 완성 된 데모를 바탕으로 UML 등을 완성본에 맞춰 그리게 되는 것이 관행이었다.

그로 인해 중간에 구현된 것을 본 이해 당사자가 요구 사항을 급하게 수정 하거나 하여 마감일 직전까지 추가 요구 사항을 수행하기 급급한 상황이 되는 경우도 자주 있어왔다.

요구공학기법과 정형검증도구를 활용 하면 이를 통해 구현 전에 먼저 요구 사항에 따른 문제점을 도출 하고 이해 당사자들에게 이해 시킬 수 있다면, 이 같은 관행이 개선되고 프로젝트의 완성도를 높이는 데에 시간과 비용을 들일 수 있을 것이다.

ACKNOWLEDGMENT

본 연구가 나올 수 있도록 열정을 가지고 인도해 주신 유준범 교수님께 감사의 인사를 드립니다.

참 고 문 헌

[1] A. CimattiGiunchigliaF. " www.fbk.eu. " http://nusmv.fbk.eu/NuSMV/index.html .

[2] Addison-Wesley. "Software Engineering (9th ed.)." Sommerville, Ian, 2009.

[3] L.McMillanK. " Symbolic Model Checking. " Kluwer Acadecmic Publisher, 1993.

[4] PnueliManna and A.Z. " The Temporal Logic. " Springer-Verlag, 1996.

[5] RalphPaul. "The illusion of requirements in software development." "Requirements Engineering" (Springer-Verlag London Limited), 2013: 293-296.