모델 기반 임베디드 소프트웨어의 개발 경험

이정림° 박사천 권기현
경기대학교 정보과학부
{jrim°, sachem, khkwon}@kyonggi.ac.kr

Experience in Model Driven Development of Embedded Software

Jungrim Lee°, Sachoun Park, Gihwon Kwon
Department of Computer Science, Kyonggi University

요 약

임베디드 시스템 개발의 성공여부는 점점 하드웨어에서 소프트웨어쪽으로 비중이 옮겨지고 있다. 그런데 임베디드 소프트웨어 개발은 하드웨어를 설계하고 구현하는 것에 드는 비용보다 더 많은 비용으로도 효과적이지 못한 실정이다. 또한 임베디드 소프트웨어 개발자들에게는 제한된 자원과 여러 가지 환경 변수를 고려해야 하는 부담이 따른다. 모델기반의 개발 방법은 비용-효율적이고 고품질의 소프트웨어를 개발하기 위한 연구이다. 본 논문은 임베디드 소프트웨어인 로봇 작동 프로그램을 모델기반으로 개발한 경험과 결과를 소개함으로써 임베디드 소프트웨어 개발자들이 쉽게 모델기반의 개발 방법을 적용할 수 있도록 한다.

1. 서 론

임베디드 시스템 개발에 있어서 성공여부는 하드웨어 보다 소프트웨어쪽으로 비중이 옮겨지고 있다. 고객의 다양한 요구사항을 충족시키기 위해서는 소프트웨어의 유연성이 강조되기 때문이다. 그런데 임베디드 소프트웨어의 개발은 제한된 자원과 여러 가지 환경변수를 고려해야 하는 어려움이 있고, 에러 발생시에 디버깅이 PC 프로그램보다 훨씬 어렵다. 이를 해결하기 위한 방법으로 모델 기반의 소프트웨어 개발 방법이 연구되고 있고 현재 많은 CASE도구 회사들에서 모델기반의 임베디드 소프트웨어 개발을 위한 지원 도구를 내놓고 있다. 코드에 의한 관리보다는 모델에 의한 관리가 훨씬 경제적이며 또한 에러 디버깅이 훨씬 수월하기 때문이다.

본 논문의 목적은 모델 기반의 로봇 소프트웨어 개발 경험을 소개함으로써 임베디드 소프트웨어 개발자들이 쉽게 모델기반의 방법을 사용해서 고품질·고효율의 소프트웨어를 개발할 수 있도록 돕는데 있다. 본 연구의 개발 환경은 대학에서 임베디드 교육용으로 널리 사용되는 레고 마인드스톰을 사용했고, 운영체제로는 자바 언어로 된 LeJos, CASE도구로는 Rational사의 Rose RT를 이용했다. 모델링 대상은 두 개의 에이전트(도망자와 추적자)가 미로에서 상반된 서로의 목적을 달성하는 메이즈(Maze) 게임으로 했다. 실제의 개발결과 모델 내부에서 기술해준 코드의 비율이 자동 생성된 코드 대비 20% 미만이었다.

2장에서는 모델기반의 개발과 Rose RT의 개관에 대해서 살펴보고 3장에서 모델링 대상인 메이즈 게임을 설명한 후 4장에서는 Rose RT 도구를 사용해서 게임을 모델링하고 코드를 생성해서 컴파일한 후 테스팅하는 과정을 설명한다. 마지막으로 5장에서 결론을 맺는다.

2. 배경연구

임베디드 소프트웨어는 항공우주, 의료, 자동차, 무선 통신, 가전 등 산업의 전분야에서 개발되고 있고 이러한 소프트웨어 개발 프로세스의 현 수준은 몇 가지 중요한 원칙들 위에 놓여있다. 즉, 개발 초기에 위험을 발견하고 그것을 **완화해** 나가는 **점진적 개발**, 소스코드 수준만으로 효율적으로 구축할 수 없었던 크고 복잡한 시스템 개발에 추상 모델을 제공함으로써 개발자로 하여금 어플리케이션의 중요 특성들을 발견하게 하고 하위 수준의 구현으로부터 독립시켜주는 **모델 기반 개발**, 코드와 설계모델의 관리를 용이하게 하며 추적성을 도모하는 **모델-코드 상호적 연관성**, 설계 모델의 모델기반 코드 생성의 핵심인 **실행 가능한 모델**, 극도로 복잡한 어플리케이션에 대한 이해와 개발을 돕는 **추상 설계 단계에서의 디버그 와 테스트** 등이 그것이다[1].

임베디드 소프트웨어의 모델기반 개발을 지원하는 대표적인 도구로는 IBM Rational사의 Rose RT, I-Logix 사의 Rhapsody 등이 있다. 본 논문에서 사용한 Rose RT는 기본 UML을 지원하며 특별히 실시간 시스템을 위해 ROOM 개발 방법론에 기초해서 다음과 같은 특성이 추가되었다.

- ***Capsules:*** 캡슐은 클래스의 스테레오 타입으로 상태 다이어그램을 사용한 이벤트 기반의 행위 모델링과 캡슐간의 통신관계의 모델링을 위해 추가된 구문을 가지는 클래스이다.
- ***Protocols:*** 캡슐들 사이의 메시지 교환을 송신자와 수신자 관점에서 정의한다.
- ***Ports:*** 캡슐간의 메시지 교환을 목적으로 하는 포트는 캡슐에 의해서 생성되고 파괴된다.

- *State Diagrams:* UML에 정의된 표기를 사용하여 캡슐의 행위를 구성하며 소스코드를 생성한다.
- *Capsule Structure Diagram:* 포트와 캡슐의 역할을 명세하는 다이어그램으로 UML 1.3의 협력 다이어그램에 기반 한다. 캡슐간의 통신관계를 명세 한다[2].

3. 문제

메이즈 게임[1]은 플레이어가 도망자가 되어 미로 속에서 추적자를 따돌리고 탈출하는 게임이다. 두 에이전트는 처음에 각자의 출발점에 놓여지며 도망자부터 벽이 없는 방향으로 번갈아 움직일 수 있다. 도망자는 상하좌우 4방향으로 한번에 한 칸 움직일 수 있거나 그 자리에 머물 수 있다. 추적자는 도망자를 향하여 한번에 두 칸씩 움직일 수 있다. 그런데 추적자는 도망자를 향하여 먼저 좌우로 움직이도록 프로그램 되어 있다. 아래 메이즈 게임에서 굵은 선은 벽이고 붉은색 원은 도망자, 검은색 원은 추적자이다. 도망자는 벽을 교묘히 이용해서 추적자를 따돌리고 탈출할 수 있게 된다.

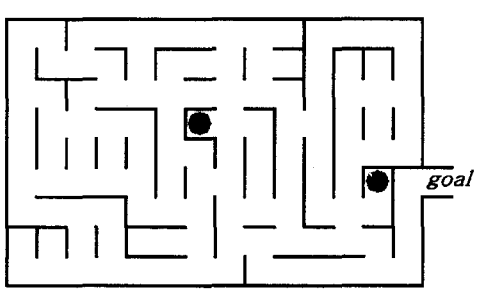

그림 1. 메이즈 게임 15판

우리는 먼저 이 게임을 모델 체커 SMV의 입력언어로 모델링하고 $\neg E[safe U goal]$의 속성으로 모델 체킹하여 도망자의 탈출 경로를 얻어내었다. [3]의 푸쉬푸쉬 게임 50판의 경우 48개의 셀로 구성되고 2^{37}의 상태공간을 사용했던 반면 메이즈 15판은 126개(14×9)의 셀로 구성되고 2^{14}의 상태공간(메모리: 16,372KB, 시간: 16.694초, BDD 수: 529,694)에서 해결되었다.

아래 그림 2 는 레고 마인드스톰으로 구현한 메이즈 게임이다. 두대의 RCX 로 도망자와 추적자를 만들고 흰색 보드에 검은색 테이프로 셀을 만들었다. 마인드스톰에 적재한 소프트웨어는 자바 언어이고 4 장에서 설명할 순서와 방법으로 얻어냈다.

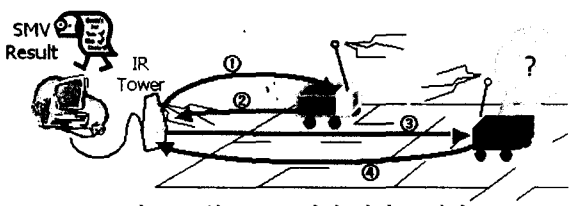

그림 2. 로봇으로 구현한 메이즈 게임

1) http://www.tnelson.demon.co.uk/mazes/

각 RCX는 적외선 통신 포트를 가지고 있는데 본 논문에서 구현한 두 개의 에이전트 중 도망자는 컴퓨터에 연결된 IR 타워로부터 자신의 위치 정보를 수신 받아 목표점까지 움직여가고, 추적자도 도망자의 위치정보를 수신 받고 자신의 위치를 계산한 후 움직이도록 만들었다.

4. Rose RT를 이용한 모델기반 개발

4.1 모델링

모델링의 첫 단계로 요구사항 분석을 위해 하나의 Usecase와 그에 대한 시나리오를 작성하고, 캡슐과 클래스를 추출해내었다. 그 후 시나리오와 추출된 캡슐을 통해 Sequence 다이어그램으로 전체 흐름을 나타내었다.

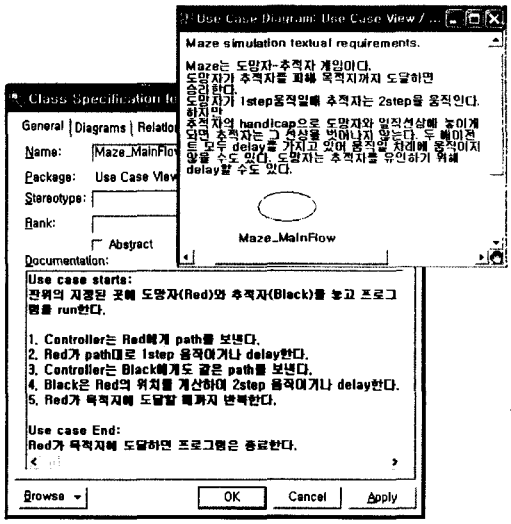

그림 3. Usecase와 시나리오

다음 단계로 Class 다이어그램에서 캡슐간의 관계를 형성하였다. 이때 캡슐은 다른 캡슐과 메시지를 어떻게 주고 받을지에 대한 구조 정보를 갖는 Structure 다이어그램과 행위를 나타내는 State 다이어그램으로 구성된다. 그림 4는 Class, Structure, State 다이어그램들의 연관성을 보인 그림이다.

1) Class 다이어그램으로 Maze 캡슐이 Red(도망자). Black(추적자). Controller(컴퓨터) 캡슐로 구성되는 집합 연관 관계를 정의하고 통신을 위한 프로토콜이 정의되어있다.

2) Maze 캡슐의 Structure 다이어그램으로 각 캡슐간의 Capsule role과 연결을 정의한다.

3) Controller 캡슐의 Structure 다이어그램으로 내부 또는 외부로 통신하기 위한 포트들이 정의되어있다.

4) Red 캡슐의 통신을 위해 RCX의 포트를 여는 State 다이어그램이다.

5) Red 캡슐이 메시지를 기다렸다가 받은 메시지에 따라 이동 여부를 결정하는 State 다이어그램이다.

6) Red의 이동시 회전, 전진, 보정 행위를 하는 State 다이어그램이다.

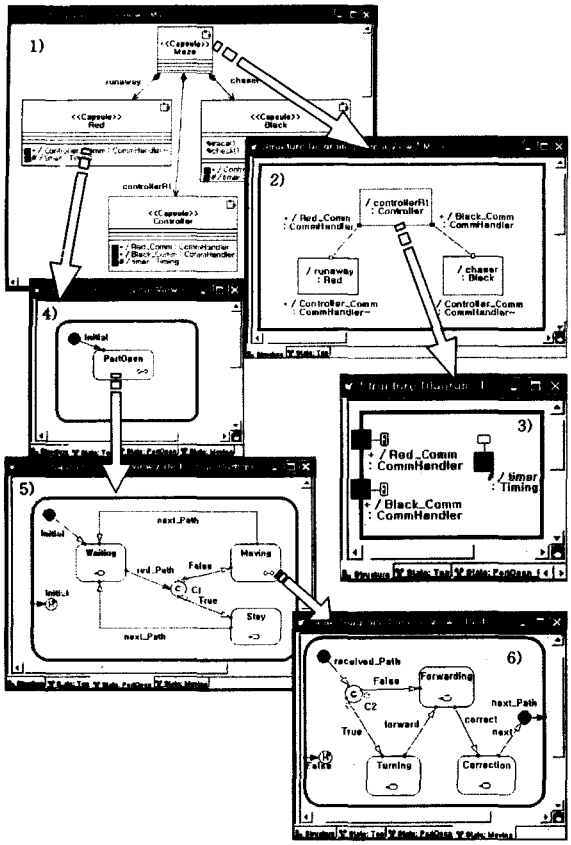

그림 4. 다이어그램간 중첩관계

4.2 코드 생성과 컴파일링

RCX에 적재할 Firmware로써 Lejos(Lego Java OS)를 선택하고 그 위에 응용 프로그램 언어로 자바를 선택했다. 따라서 RCX에서 사용되는 언어는 표준 자바보다 제한적이지만 자바 언어와 비슷하기 때문에 코드 생성을 위해 Rose RT에서 제공하는 자바 언어를 사용하였다. 이때 모델에서 코드를 생성하고 컴파일하기 위해 Component 다이어그램에 해당하는 캡슐, 클래스와 관련 라이브러리를 포함한 컴포넌트를 만든다.

Rose RT의 자바 코드 생성기는 모델 내 요소들의 모델 속성들을 포함하는 코드를 생성한다. Rose RT는 병행성, 메시지 전송, 시간 서비스와 같은 특정 실시간 서비스를 지원하는 서비스 라이브러리를 제공하는데 이와 함께 Lejos 컴파일러를 이용하여 코드를 생성하고 컴파일했다. 이렇게 생성된 코드를 보면 직관적으로 코드 기반의 소스와는 좀 다른 형태를 보인다. 하지만 컴파일 에러가 발생하여 디버깅할 때 소스 코드 내에서 하는 것이 아니라 모델 내의 에러가 발생한 state를 알려주고 코드를 보여주기 때문에 디버깅이 매우 수월하고 비주얼하여 어렵지 않게 이해할 수 있었다. 이렇게 점진적으로 모델링 해가면서 코드 생성, 컴파일 그리고 테스팅하였다.

4.3 결과 분석

실제 소스기반의 개발은 에이전트마다 프로그램을 탑재하고 테스팅 하는데 모델링해서 시뮬레이션하는 것보다 매우 많은 시간이 걸렸고 점진적으로 코드를 생성하고 테스팅 하기가 어려웠다. 테스팅을 위해서 에이전트의 LCD 창이나 비프음을 이용하기도 하고 제공되는 에뮬레이터도 있지만 이것도 한계가 있어 이 같은 방식은 많은 노력을 감수하여야 했다. 하지만 모델 기반의 개발은 점진적으로 모델을 그려가며 쉽고 빠르게 눈으로 확인하고 디버깅할 수 있어 개발 프로세스 내내 지속적으로 모델을 테스트할 수 있었다. 물론 시뮬레이션과 달리 실제 동작시켜보는 것은 외부 환경의 자극으로 예상치 못한 일들이 발생할 수 있다. 모델 기반 개발에서 실제의 소스 코드는 State 다이어그램의 요소인 상태와 전이 내부에 액션 코드로 입력한다. 아래 표 1은 각 캡슐과 클래스에 대해서 모델상에서 기술한 코드와 실제 생성된 코드의 라인 수를 비교한 것이다. 이렇게 입력한 코드의 라인 수는 전체 코드의 약 16%를 차지했다. 비록 국지적인 결과이기는 하지만 전체 코드의 길이에 비해서 상당히 적은 양의 코딩으로 구현이 가능함을 알 수 있다.

표 1. 모델 내의 코드와 생성된 코드의 라인 수 비교

캡슐/클래스	모델 내 코드	생성된 코드	비율
Red	96	544	17.6%
Black	121	711	17.0%
Controller	28	612	4.5%
Move	65	65	100.0%

5. 결론 및 향후 연구

본 논문에서는 임베디드 소프트웨어를 개발하기 위해 모델 기반의 개발 방법을 사용하였다. 적용한 도메인은 레고 마인드스톰으로 구현한 메이즈 게임이었고, 개발 도구로 Rose RT를 사용했다. 실제 코드의 20% 미만의 코드만이 모델 내에서 작성되어 시스템을 구현하였고 모델 기반 테스팅과 디버깅은 개발 사이클을 줄여주었다.

하지만 코드 기반으로 작성된 코드와 모델 기반으로 생성된 코드의 품질에 대한 비교와 모델과 코드 수준의 검증에 관한 연구가 향후 과제로 남아있다.

참고 문헌

[1] B. P. Douglass, " Real-Time UML Second Edition *Developing Efficient Objects for Embedded Systems*," Addsion-Wesley, 2000

[2] Rational Software, " Tutorials Rational Rose Real Time," Version 2002.05.00, 2002.

[3] 이태훈, 권기현, " 상태폭발 문제 해결을 위한 릴레이 모델 체킹," 한국 소프트웨어공학 학술대회 논문집, Vol. 6, No. 1, pp.298~305, 2004.

[4] G. Farrari, A. Gombos, S. Hilmer, J. Stuber, "Programming Lego Mindstorms with Java: The Ultimate Tool for Mindstorms Maniacs," Syngress, April 2002.