xUML을 사용한 MDA 기반 임베디드 소프트웨어 컴포넌트 시스템을 위한 설계 재사용

김우열° 김동호 문소영 김영철
홍익대학교 컴퓨터정보통신공학과
{john°, dhkim, moon, bob}@selab.hongik.ac.kr

Towards a Reusable Design for Embedded Software Component System Based on MDA with Executable UML

Wooyeol Kim° R. Youngchul Kim
Dept. of CIC, Hongik University, Jochiwon, Korea

요 약

향후 유비쿼터스 컴퓨팅 환경에서의 임베디드 소프트웨어는 다원화된 네트워크 환경에서 동작하게 될 것이다. 임베디드 소프트웨어가 이 기종의 시스템에서 다양한 형태의 응용 프로그램으로 쉽게 탑재되기 위해서는 설계와 코드의 재사용이 필수적이다. 임베디드 시스템에서 소프트웨어 설계의 재사용이 가능하다면 개발 시 소요되는 시간과 비용이 절감될 것이다. 그러나 임베디드 소프트웨어는 많은 부분이 시스템에 의존적이기 때문에 재사용이 어렵다는 단점을 가지고 있다. 본 논문에서는 이러한 단점을 해결하고자 기존의 MDA 메커니즘과 개선한 Multiple V-model의 접목을 시도하고 임베디드 소프트웨어 컴포넌트 설계의 재사용을 제안하다. 그리고 적용사례로서, 이기종의 임베디드 시스템들에 소프트웨어 컴포넌트를 탑재하였다.

1. 서론

향후 유비쿼터스 컴퓨팅 환경에서의 임베디드 소프트웨어는 극히 다원화된 네트워크 환경에서 동작하게 될 것이며, 그 기능과 형태가 매우 다양해질 것이다. 따라서 이 기종의 기기 위에서 다양한 형태의 응용 프로그램이 쉽게 탑재될 수 있도록 해야 한다[1]. 하지만 임베디드 소프트웨어는 많은 부분이 시스템에 의존적이기 때문에 다른 타겟으로의 탑재가 힘들다는 단점이 있다.

기존의 MDA는 설계 모델을 점진적으로 변환하여 소프트웨어를 자동으로 생성하는 개발 방식이다. 이를 지원하기 위한 MDA의 핵심은 소프트웨어 개발 프로세스 모델이며 시스템을 모델링하는 행위가 곧 제품을 구현하는 생산 작업이 된다. MDA는 자동화 도구를 이용하여 PIM을 PSM으로 변환하고, 다시 PSM을 코드로 생성하는 변환 작업을 수행한다[2].

이러한 임베디드 소프트웨어 컴포넌트 개발 방법과 기존 MDA 메커니즘이 접목된다면 기존에 비해서 재사용성 증가, 소프트웨어 정량화, 개발 시간의 단축 등의 장점을 얻을 수 있다. 또한 모델을 재사용함으로써 하드웨어에 독립적으로 소프트웨어 컴포넌트의 개발이 가능해질 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 MDA의 개발 프로세스와 실행 가능한 모델인 Executable UML에 대해 언급하고, 3장에서는 제안한 MDA 기반 임베디드 소프트웨어 컴포넌트 개발 프로세스에 관해 살펴볼 것이다. 4장에서는 제안한 프로세스의 적용사례로서 이기종의 임베디드 시스템에 개발한 컴포넌트를 적용해 보았다. 마지막으로 5장에서는 결론 및 향후 연구를 설명한다.

2. 관련연구

MDA는 OMG에서 정의한 소프트웨어 개발 프레임웍으로서, 소프트웨어 개발 프로세스에서 메타모델을 강조한다. MDA는 아키텍처의 관심 분리 원칙 (Separation of Concern)을 적용해서 이식성 (Portability), 상호운용성 (Interoperability), 재사용성 (Reusability)을 증가시킬 목적으로 사용되며, 모델은 다른 모델로 변환하므로 소프트웨어 개발의 자동화를 지원할 수도 있다. MDA는 CIM(Computation Independent Model), PIM(Platform Independent Model), PSM(Platform Specific Model)의 세 가지 관점으로 시스템을 명세한다[3,4].

MDA 개발 프로세스에는 플랫폼에 독립적인 정보를 가지고 있는 PIM에 주로 초점을 맞추어 개발한다. PIM에서 PSM 변환 시 도구에서는 자동적으로 플랫폼에 종속적인 기술 정보가 추가된다. 따라서 다양한 플랫폼의 많은 시스템을 빠른 시간에 개발함으로서 생산성을 높일 수 있다. MDA에서는 같은 PIM에서 다른 플랫폼에 맞는 다양한 PSM을 생산 할 수 있는 이식성을 제공하며, PSM에서 수정 혹은 변화가 있더라도 시스템 유지보수 시 최종 산출물이 아닌 모델을 변경하여 분석에서 개발까지 자동화 도구를 이용하기 때문에 모델과 구현간의

일관성을 유지할 수 있는 장점을 가지고 있다[4,5].

본 논문에서는 MDA를 임베디드용 개발 프로세스에 적용 시키고자 PIM은 TIM(Target Independent Model)로, PSM은 TSM(Target Specific Model)로, 그리고 Code는 TDC(Target Dependent Code)라고 변환하였다.

Executable UML은 기호로 스펙을 설명하는 언어이다. Executable UML은 UML 표기법에 "실행에 관련된 개념들(executable semantics)"과 "시간에 관련된 규칙들(timing rules)"을 더한 것이다. Executable UML을 이용하면 클래스(class)와 상태(state)와 액션(action) 모델로 이루어진 Executable 시스템 스펙을 만들 수 있다. Executable 시스템 스펙은 하나의 완전한 프로그램처럼 실행된다. 기존 스펙과 달리 Executable 스펙은 실행과 테스트, 디버깅이 가능하며, 시스템 성능 측정을 위해서도 이용할 수 있다. 스펙(모델)에 대한 테스트가 끝나면, 이를 타겟 코드(target code)로 변환(translate)할 수 있다[6,7,8].

그림 1 Executable UML 구성 요소[9]

Executable UML은 그림 1과 같이 근간이 되는 UML 1.4에 의미적으로 약한 요소를 배제시키고 실행 가능하도록 정확하게 정의된 행위에 관한 의미들을 추가함으로서 탄생되었다.

하지만 Executable UML 역시 모델 자체가 코드를 의미하는 것은 아니다. 단지 여기에서의 모델은 전통적인 UML에는 없는 Java나 C++등과 같은 언어로 재작성 하기 위해 초점이 맞춰져 있다는 것이다.

3. 개발 프로세스

기존 임베디드 소프트웨어 컴포넌트 개발 프로세스는 모델링을 한 후에 프로토타입을 만들게 되고, 이 프로토타입을 기반으로 최종 생산품을 만드는 수순이었다. 이 때 재사용의 개념은 고려하지 않고 오직 하나의 제품에 적합한 공정만을 중요시 하였다. 이러한 문제점을 해결하고자 우리는 그림 2와 같이 기존의 임베디드 소프트웨어 개발 프로세스와 MDA 메커니즘 접목을 시도 한다.

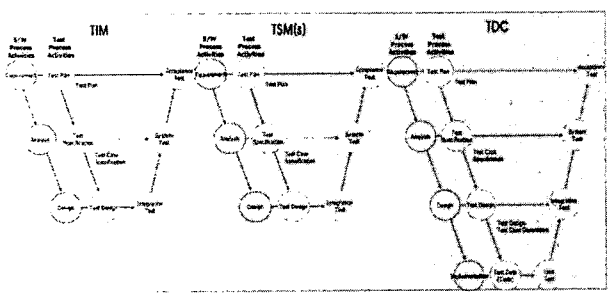

그림 2 Multiple V-model과 MDA의 접목

그림 2에서 주목할 점은 개발 초기단계부터 테스트 작업을 계획하여 개발 프로세스와 테스트 프로세스가 동시에 병렬적으로 수행된다는 것이다. 이는 지금까지 개발과 테스트가 따로따로 이루어지는 단점을 개선해 더욱 안정된 소프트웨어 컴포넌트의 개발 및 재사용이 가능하다.

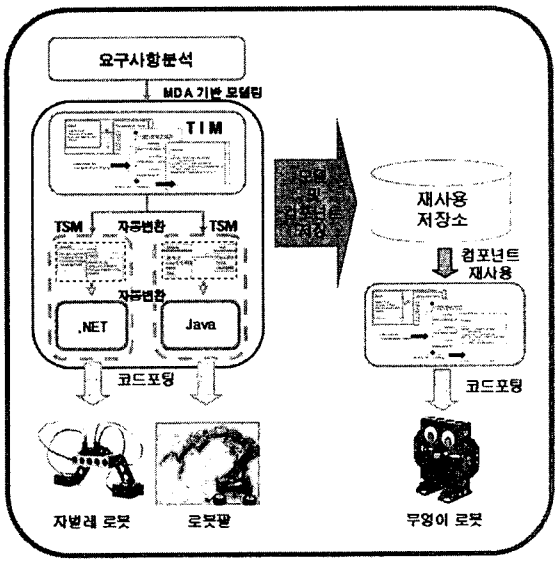

그림 3 임베디드 소프트웨어 컴포넌트용 개발 프로세스

그림 3은 MDA와 Multiple V-model을 접목해 본 논문에서 제안한 임베디드 소프트웨어 컴포넌트용 개발 프로세스를 도식화 한 것이다. 우선 요구사항에 맞추어 TIM을 설계한다. 다음으로는 자동화 도구를 사용하여 각자 타겟에 종속적인 모델인 TSM으로 변환되고, TSM 또한 도구를 사용하여 코드화 된다.

이후 생성된 소스코드를 우리의 타겟 모델에 탑재하는 작업이 바로 개발한 소프트웨어 컴포넌트를 임베디드 시스템에 적용시키는 것이다. 이때 생성된 소스코드와 모델은 재사용 저장소에 저장되어지고 후에 기능을 변경하거나 추가하여 새로운 임베디드 시스템을 구축하고자 한다면 저장된 모델을 재사용하여 개발 시간을 단축시킬 수 있다. 최종적으로는 모델 단계에서의 재사용을 통해 우리의 최종 목표인 상호운용성도 높일 수가 있다.

4. 적용사례

본장에서는 개발된 소프트웨어 컴포넌트를 임베디드 시스템에 탑재해 컴포넌트와 모델의 재사용을 검증한다.

4.1 자벌레 로봇의 설계

우리는 간단한 임베디드 시스템으로서 자벌레 로봇(Looper Robot)을 이용하였다. 자벌레 로봇은 AMTEL사의 CPU를 사용하는 DigiBlock 로봇의 일종으로서 6개의 서보모터를 제어하여 자벌레가 기어가는 형상을 그대로

표현한 임베디드 시스템이다. 그림 4(a)는 자벌레 로봇의 기어가는 동작을 할 수 있는 기능을 가진 Motion 클래스를 설계한 것이다. 이는 임베디드 시스템에 독립적인 모델이다. 그림 4(b)는 TSM, 즉 시스템에 종속적인 모델이며 그림 4(c)는 자벌레 로봇의 동작을 담당하는 스켈레톤 코드이다.

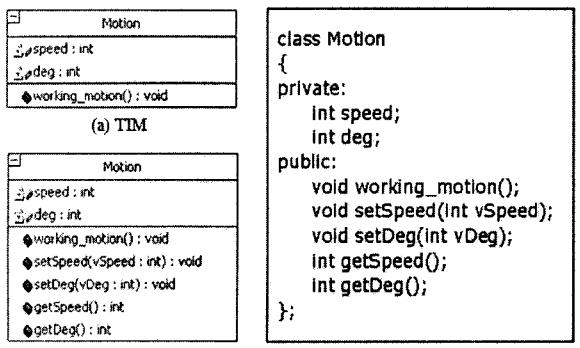

그림 4 자벌레 로봇의 동작 설계

4.2 컴포넌트의 재사용

부엉이 로봇(Owl Robot)은 자벌레 로봇과 같은 CPU를 사용한다. 하지만 자벌레 로봇의 기어가는 동작을 담당했던 컴포넌트를 부엉이 로봇에서는 허리를 구부려 인사를 하는 동작을 담당하는 컴포넌트로 사용해 컴포넌트의 재사용을 확인하였다.

4.3 모델의 재사용

본 단락에서는 앞서 살펴본 컴포넌트의 재사용이 아닌 모델 재사용을 확인하고자 레고 마인드스톰을 사용하여 로봇팔(Robot Arm)을 구성하였다. 이는 자벌레 로봇이나 부엉이 로봇과는 달리 Java를 기반으로 동작하는 임베디드 소프트웨어 컴포넌트 시스템이다.

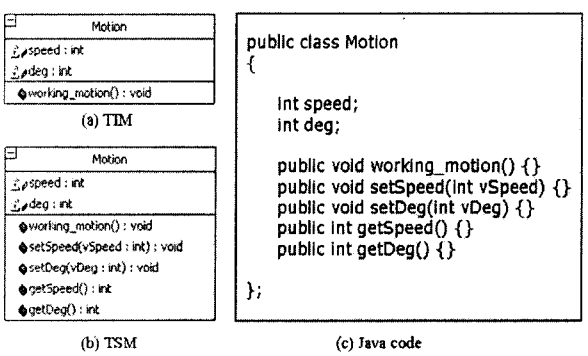

그림 5 모델의 재사용

그림 5는 MDA를 기반으로 자벌레 로봇의 움직이는 기능을 구현하는데 사용했던 모델을 재사용하여 로봇팔에 알맞은 코드를 생성한 결과이다. 그림 4에서의 모델과는 일치하지만 코드부분에서는 차이가 나는 것을 알 수 있다. 구현된 소프트웨어 컴포넌트는 로봇 팔을 굽히는 용도로 사용된다.

5. 결론 및 향후 연구

임베디드 소프트웨어는 많은 부분이 시스템에 의존적이기 때문에 개발 단계에서 생성한 설계와 코드의 재사용이 어렵다는 단점이 있다.

본 논문에서는 이러한 단점을 해결하고자 기존의 MDA 메커니즘과 개선한 임베디드 소프트웨어 컴포넌트 개발방법의 접목으로 설계의 재사용을 통해 개발 시간 단축과 이를 통한 이종 시스템간 소프트웨어 컴포넌트의 상호운용성을 시도하였다. 또한 적용사례로써 개발된 소프트웨어 컴포넌트를 이기종의 임베디드 시스템에 탑재해 각기 다른 동작을 하는 것을 확인할 수 있었다.

향후 연구로는 소프트웨어 개발 시 Executable UML을 적용해 모델링 단계에서 코드생성까지 모든 과정을 자동화 할 수 있는 도구의 개발이 요구된다. 더 나아가 상태머신과 OCL을 이용하여 동시 발생 문제와 같은 시스템의 동적인 모델링이 가능한 도구의 연구도 필요하다.

6. 참고문헌

[1] 이관우, "특집을 내면서," 한국정보과학회 소프트웨어공학회지, 제17권, 제3호, pp.1, 2004. 9,

[2] 민현기, "컴포넌트용 프로파일을 적용한 MDA 기반의 컴포넌트 개발 기법," 한국정보과학회 소프트웨어공학회지, 제17권, 제4호, pp.21-23, 2004. 12

[3] OMG, MDA Guide Version 1.0.1, omg /2003-06-01, June 2003.

[4] 김수동, "MDA의 핵심 개념," 한국정보과학회 소프트웨어공학회지, 제17권, 제4호, pp.3-5, 2004. 12

[5] Kleppe, A. Warmer, J. and Bast, W., MDA Explained, Addison-Wesley, 2003.

[6] Leon Starr, Executable UML: How to build class models, Prentice-Hall, 2002

[7] 김인기 역, Executable UML: 클래스 모델 만들기, 정보문화사, 2003

[8] Stephen J. Mellor, Marc J. Balcer, Executable UML: A Foundation for MDA

[9] Kennedy Carter Ltd., Executable UML: An Online Tutorial, www.kc.com