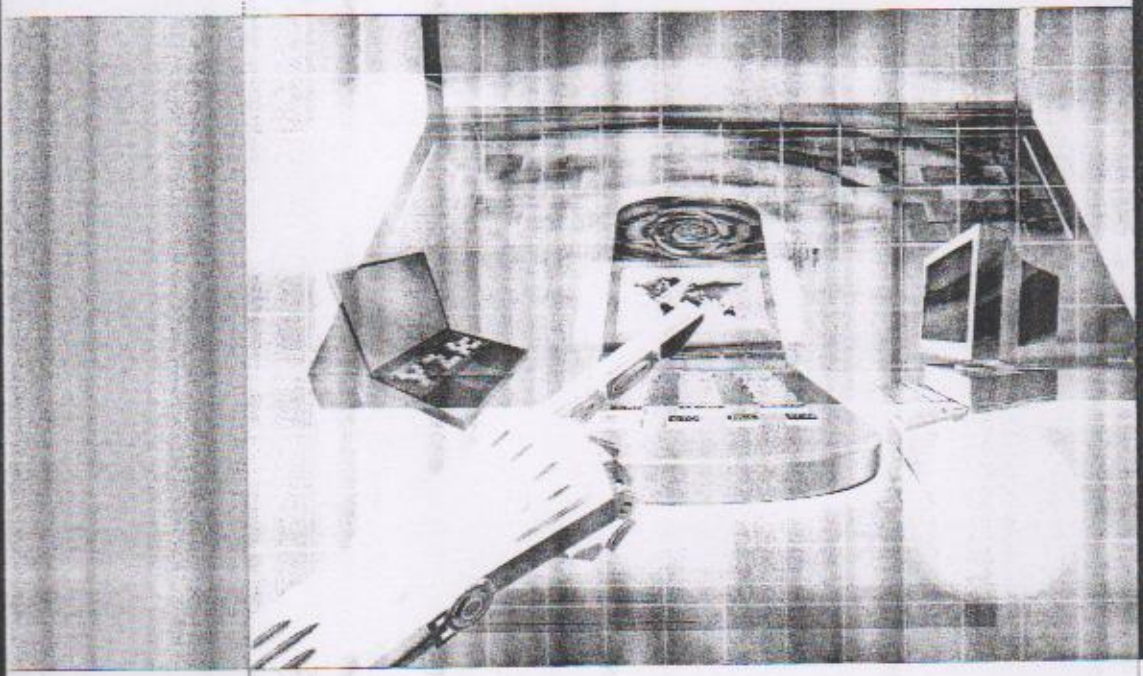
2006
한국 모바일 학회
춘계학술대회

Society of Mobile Technology Spring Conference, 2006



▶ 일 시 : 2006년 6월 2일(금) 10:00 ~ 19:00
▶ 장 소 : 경원대학교
▶ 주 최 : 한국모바일학회 (www.smt.or.kr)
▶ 주 관 : 경원대학교
▶ 후 원 : 한국인터넷진흥원, 경원대학교 U-Healthcare센터,
한국인터넷기반진흥협회, 한국무선인터넷솔루션협회
▶ 협 찬 : KT, KTF, 삼성 SDS, 시스게이트

SMT Proceedings of SMT, 2006, Vol.3. No.1

오전세션 - Track 2 - 광대역 데이터 및 멀티미디어 전송

오후세션 - Track 1 - 모바일 서비스 및 플랫폼

오후세션 - Track 2 - 모바일 정보 서비스

오후세션 - Track 3 - 모바일 정책 및 기술 동향

임베디드 소프트웨어 모델링을 위한 MDA 기반의 방법

김우열*, 김영철
홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
e-mail : {john*, bob}@selab.hongik.ac.kr,

An MDA(Model Driven Architecture) based Approach for Modeling Embedded S/W Components

Woo Yeol Kim*, R. Young Chul Kim
Dept. of Computer & Information Communication, Hongik University
e-mail : {john*, bob}@selab.hongik.ac.kr

요약문

본 논문은 임베디드 소프트웨어가 하드웨어, 운영체제, 프로그래밍 언어, 그리고 플랫폼에 종속적이기 때문에 이종의 도메인 타겟 상에서 각각의 코드를 개발하는데 목적이 있다. 이러한 소스코드 레벨에서의 재사용성에 관한 문제를 해결하고자 '임베디드 소프트웨어 컴포넌트 모델링을 위한 MDA 기반의 방법'을 제안한다. 제안한 방법을 따르면 메타 모델(TIM: Target Independent Model)을 만들고 각각의 도메인에 맞는 타겟 종속적 모델(TSM: Target Specific Model)로 변형한 후, 그에 따른 소스 코드(TDC: Target Dependent Code)를 개발한다. 그리고 기 개발된 메타 모델은 이종의 임베디드 소프트웨어를 개발할 때 재사용 된다. 이처럼 제안한 방법을 통하여 임베디드 소프트웨어에서 MDA 기반의 상호운용성을 가능케 할 것이다.

Abstract

This paper is proposed to develop each suitable code on the heterogeneous domain targets of embedded software environments depended on their own H/Ws, operating systems, programming languages, and platforms. We suggest 'An MDA based Approach for Modeling Heterogeneous Embedded S/W Components' to solve these problems of reusability in the level of source-code. Through our proposed approach, we produce 'Target Independent Meta Model', with which transforms 'Target Specific Models', and then generate 'Target Dependent Code' via each TSM. As a result, the TIM metamodel is reused when heterogeneous embedded softwares are developed. That is, it will achieve interoperability based on MDA through our proposed approach.

Key Words: Embedded Software, Component, MDA(Model Driven Architecture)

Ⅰ. 서론

최근 복잡한 기능을 제공하는 임베디드 소프트웨어를 좀 더 빠르고 효율적으로 개발하기 위해 소프트웨어를 재사용하는 방안에 대한 연구들이 진행 중이다. 그러나 임베디드 소프트웨어는 하드웨어, 운영체제, 프로그래밍 언어, 그리고 플랫폼에 종속적인 특성을 지니고 소스 코드를 중심으로 개발이 진행되기 때문에 소프트웨어의 재사용이 어렵다[1].

소프트웨어 공학에서 MDA는 플랫폼에 독립적인 메타모델을 설계한 후 필요한 기술 모델을 변경하여 그 모델을 통해 코드 생성을 자동화하는 메커니즘이다. 만약 응용 프로그램이 다른 구현 환경에서 필요하다면 그 환경에 대한 모델을 선택하고 다시 코드를 생성하면 된다. 이때 응용 프로그램 모델을 수정할 필요는 없다. 이와 같은 방법으로 모델의 재사용과 관련된 코드의 생산성을 높일 수 있다. 그래서 우리는 임베디드 소프트웨어 개발 시 MDA 메커니즘을 적용하고자 한다. 그러나 원래의 MDA는 커다란 시스템을 구축하기에 알맞은 아키텍쳐이다. 또한 MDA는 반드시 자동화 도구를 사용하여 개발해야 하는 단점이 있다.

본 논문에서는 개발 프로세스로서 기존의 임베디드 제품 개발 시 사용되었던 Multiple V-Model에 MDA메커니즘을 접목한 Enhanced Multiple V-Model을 제안한다. 제안한 방법을 사용하면 설계를 재사용할 수 있고, 소프트웨어를 정량화 할 수 있으며, 개발 시간 또한 단축시킬 수 있을 것이다. 또한 모델의 재사용으로 인하여 하드웨어에 독립적으로 소프트웨어 개발 또한 가능해질 것이다[2].

본 논문의 구성은 다음과 같다. 제2장은 관련 연구로서 대표적 소프트웨어 개발 프로세스인 MDA와 Product Line을 비교/분석하고 모델링 단계에서 채택한 xUML(Executable UML)에 대해 살펴본다. 제3장에서는 임베디드 소프트웨어를 모델링하기 위해 우리가 제안한 MDA 기반의 방법에 대해 알아본다. 제4장에서는 모델링 사례로서 이종의 임베디드 시스템을 모델링 해본다. 마지막으로 제5장에서는 결론 및 향후 연구에 대해 언급한다.

Ⅱ. 관련연구

1. MDA 개발 프로세스

MDA는 설계 환경이나 시스템에 종속적이지 않고 소프트웨어를 개발하기 위한 기술이다. MDA는 변환법칙에 의해서 도구 혹은 사람의 손을 통해 플랫폼 독립적인 모델(PIM: Platform Independent Model)을 특정 플랫폼에 종속적인 모델(PSM: Platform Specific Model)로 변환하고, 이 플랫폼 종속적인 모델을 구현소스로 변환하고자 하는 사상이다. 모델간의 변환은 기존의 전통적인 개발 프로세스와 다를 것 없지만 MDA에서는 모델이 UML프로파일로 작성되기 때문에, PIM에서 PSM으로 변환이 모두 매핑 규칙에 의해 자동화 되므로 개발자의 노력을 최소화한다[4]. 이로 인해서 MDA는 플랫폼에 독립적인 표준화 기술로 새롭게 대두되고 있다. MDA의 개발은 기술이나 구현될 기술의 특이성에 영향을 받지 않게 어플리케이션이나 시스템의 기능이나 행위에 초점을 둔다. 따라서 새로운 기술이 나와도 어플리케이션이나 시스템의 기능이나 행위 모델링을 반복할 필요가 없다. 다른 아키텍처는 일반적으로 특정 기술에 종속된다. MDA에서 기능과 행위는 한 차례의 모델화로 충분하다. 여기서 플랫폼에 독립적이어야 한다는 것은 하드웨어, 운영체제 그리고 프로그래밍 언어 모두에 독립적임을 의미한다.

[표1] Product Line 과 MDA 비교

Product Line	MDA
1) 한번의 개발 라이프사이클 후에, 핵심 자산 재사용하는 기법	1) 한번의 개발 라이프사이클 중에, 하나의 메타 모델을 재사용하는 기법
2) 고품질의 임베디드 시스템을 적시에 경제적으로 개발할 수 있는	2) 고품질의 임베디드 시스템을 적시에 경제적으로 개발할 수 있는
단일 제품군 기반 개발 방법 (Product-Line based Development)	***이종 제품군 기반 개발 방법*** (Heterogeneous Product based Development)
3) Feature Driven	3) xUML + UML 2.0 적용
4) 커스터마이징 용이	4) *코드 자동 생성기 필수*
5) UML 2.0 적용	

Product Line[3]과 MDA를 비교했을 때, 가장 중요시 되는 차이점은 개발의 시기이다. [표 1]에서도 나타나듯, Product Line은 한 번의 개발 라이프 사이클이 수행된 후에 생성된 핵심 자산을 재사용하여 새로운 시스템 개발이 가능하다. 이에 반해 MDA는 한 번의 개발 라이프 사이클 중에 하나의 메타모델을 만든 후, 메타모델을 재사용하여 새로운 시스템 개발이 가능하다. 그리고 두 가지 모두 고품질의 임베디드 시스템을 적시에 경제적으로 개발할 수 있다는 점은 동일하지만 Product Line은 단일 제품군을 개발하기에 용이한 반면, MDA는 동종뿐만 아니라 이종의 제품군 개발에도 사용이 용이하다. 하지만 MDA가 장점만을 가지고 있는 것은 아니다. MDA는 모델의 자동 변환을 통해 여러 플랫폼을 쉽게 지원하고 시간을 많이 잡아먹는 코드 작성 부분을 줄일 수 있으며 쉽게 유지보수가 되도록 하는 자동화 도구, 특히 코드 자동 생성기가 필수 요건이라는 단점을 안고 있다.

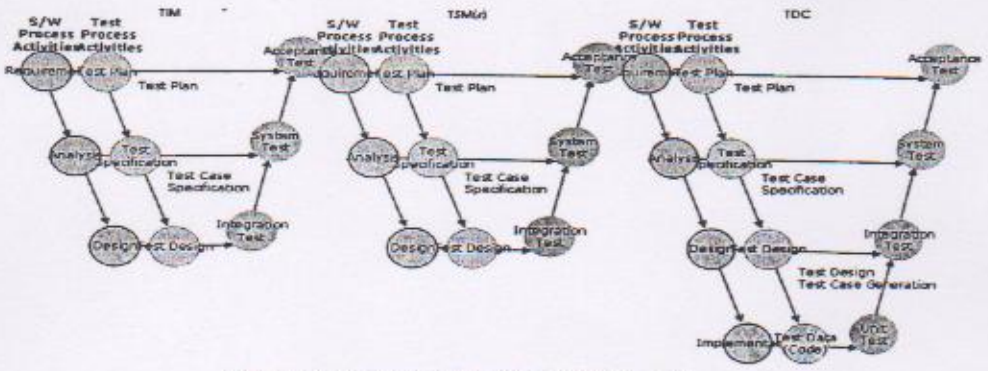
2. xUML(Executable UML)

Executable UML[4]은 기호로 스펙을 설명하는 언어이다. Executable UML은 UML(Unified Modeling Language) 표기법에 "실행에 관련된 개념(executable semantics)들"과 "시간에 관련된 규칙(timing rules)들"을 더한 것이다. Executable UML을 이용하면, 클래스(class)와 상태(state)와 액션 모델(action model)로 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.

타겟 코드는 단일 프로세서 환경에서 실행될 수도 있고, 여러 개의 프로세서들로 이루어진 프로세서 네트워크 환경에서 실행될 수도 있다. 기존의 모호한 모델과 달리, Executable 모델은 시간에 관련된 문제와 동기화에 관련된 문제, 그리고 자원의 획득과 이용에 관련된 문제들을 자세히 서술한다. 결론적으로, Executable UML은 복잡한 실시간(realtime) 분산(distributed) 임베디드(embedded) 시스템의 스펙을 서술하기에 적합하여 많이 이용된다.

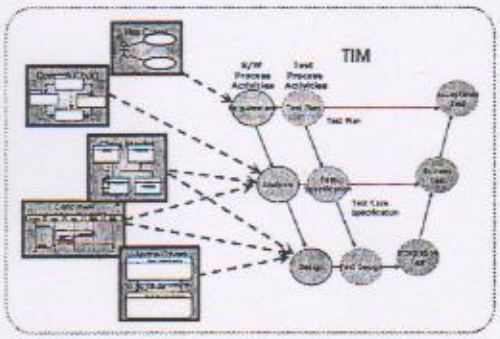
Ⅲ. MDA 기반의 임베디드 소프트웨어 개발 프로세스

지금까지 임베디드 시스템을 개발하는데 쓰여왔던 Multiple V-Model은 하나의 특정한 도메인을 목적으로 모델링한다[4]. 이는 단일제품군 개발에는 적용될 수 있으나, 이종의 제품군 개발 시에는 적용이 힘들다. 그래서 본 논문에서는 <그림 1>과 같이 기존의 임베디드 소프트웨어 개발 프로세스에 MDA 메커니즘을 접목시켰다[5, 6]. <그림 1>에서 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 개발이 가능하다.



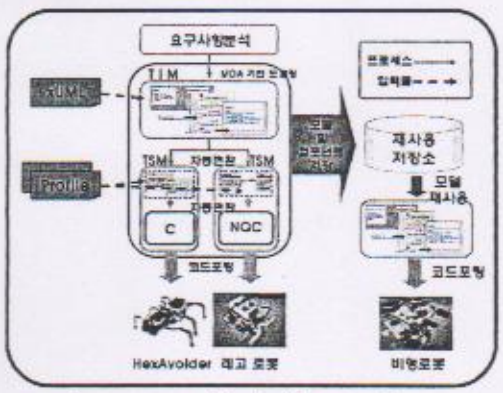
<그림 1> Multiple V-model과 MDA의 접목

<그림 2>는 임베디드 소프트웨어 개발 프로세스에서 xUML 다이어그램이 적용되는 것을 도식화한 것이다.



<그림 2> TIM 단계에서의 xUML 적용

요구사항(Requirement)에서는 유스 케이스로 수집을 하고, 다음 분석(Analysis) 및 설계(Design)는 거의 동시에 이루어진다고 봐도 무방하다. 도메인차트로 문제 영역에 대한 분할을, 클래스 다이어그램으로 정적 모델링을, 그리고 Concurrent 메시지 다이어그램으로 동적 모델링을 한 후 OCL이 포함된 Concurrent 상태 차트로 설계 단계까지 마무리 하게 된다.



<그림 3> MDA 기반의 개발 프로세스

제안한 방법의 전체적인 개발 프로세스를 <그림 3>에 나타내었다. 요구사항 분석을 거친 후에 TIM(Target Independent Model) 단계에서 xUML을 사용하여 모델링하고 TSM(Target Specific Model)과 TDC(Target Dependent Code) 단계는 자동화 도구를 통한 각각의 프로파일이 적용되어 작업이 이루어지기 때문에 TIM 단계에서 신뢰성 있는 모델이 생성되어야 한다. 그리고 이때 생성된 모델 및 컴포넌트는 향후 재사용 된다.

Ⅳ. 모델링 사례

본 논문에서는 MDA를 기반으로 임베디드 소프트웨어를 모델링 하는 방법을 제안하였다.

<그림 4> 임베디드 소프트웨어 시스템의 구조

<그림 4>는 모델링 사례로서 임베디드 소프트웨어 시스템의 구조를 나타낸다[7]. 이는 센서에 값이 들어오면 시스템에 메시지를 보내고 시스템은 실시간으로 반응하여 모터에 움직임 값을 보내주는 것을 도식화 한 것이다.

우리는 적용사례로서 프로그래밍이 가능한 이종의 임베디드 소프트웨어 시스템인 레고사의 MindStorm과 마이크로 로봇사의 HexAvoider를 이용하였다.

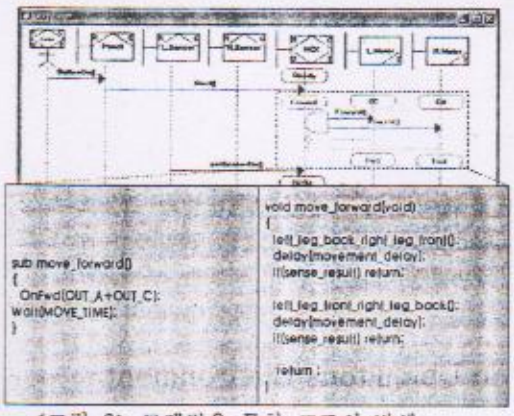
구분		LEGO	HexAvoider
적용 사양	CPU	HITACHI	AT89C51 (Flash 4K)
	OS	BrickOS	None
	컴파	NQC	C

<그림 5> MindStorm과 HexAvoider의 비교

<그림 5>는 MindStorm과 HexAvoider를 비교한 것이다. 그림과 같이 MindStorm은 HITACHE의 CPU가 내장되어있고 레고사의 BrickOS가 사용되어 NQC로 동작하며, HexAvoider는 AT89C51 CPU가 내장되어 운영체제 없이 C언어로 동작하는 임베디드 시스템이다. 이처럼 두 로봇은 CPU, OS, 그리고 사용하는 프로그램 언어까지 상이한

이종의 임베디드 소프트웨어 시스템이다.

<그림 6>은 모델링 한 결과를 수작업으로 코드화 한 것이다.



<그림 6> 모델링을 통한 코드의 발생

임베디드 시스템은 하드웨어, 운영체제, 프로그래밍 언어에 종속적이기 때문에 <그림 6>과 같이 하나의 동적 모델에서 여러개의 코드가 생성된다. 이처럼 각각의 시스템이 상이한 구조를 지니기 때문에 메타모델이 중요시되는 MDA기반으로 코드를 생성하였다.

V. 결론

임베디드 소프트웨어는 하드웨어, 운영체제, 프로그래밍 언어, 그리고 플랫폼에 종속적이기 때문에 서로 다른 도메인 타겟상에서 각각의 코드를 개발해야 한다.

본 논문에서는 이종의 임베디드 소프트웨어 개발 시 재사용성에 관한 문제를 해결하고자 MDA기반의 모델링 방법을 제안하였다. 제안한 방법으로 메타 모델을 만들고, 이를 기반으로 이종의 임베디드 시스템에 맞는 종속 모델을 만든 후, 코드를 생성하여 설계의 재사용이 가능함을 확인할 수 있었다. 뿐만 아니라 모델의 재사용으로 인하여 임베디드 소프트웨어에서 MDA기반의 상호운용성을 확인할 수 있었다.

향후 연구로는 모델을 검증하기 위한 연구와 시뮬레이션뿐만이 아닌 코드 생성까지 자동화할 수 있는 도구가 필요하다. 그리고 코드 변화 시에 적용되어야 하는 프로파일에 관한 연구와 제안한 프로세스를 지원해줄 수 있는 프레임워크와 방법론의 완성이 요구된다.

VI. 참고문헌

[1] Axel Jantsch, 'Modeling Embedded System and SOCs', Mogan Kaufmann, 2004.

[2] 김우열, MDA 기반의 임베디드 소프트웨어 모델링에 관한 연구, 홍익대학교 석사학위논문, 2005.

[3] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, Vol. 9, No. 4, Jul./Aug. 2002, pp.58-65.

[4] 김동호, 김우열, 김영철, "MDA기반의 임베디드 소프트웨어 설계에 관한 연구", 한국인터넷방송통신TV학회, 논문지, 제6권, 제1호, 2006. 3, pp67-74.

[5] 김우열, 김영철, "실시간 임베디드 소프트웨어 모델링을 위한 xUML 확장에 관한 연구", 한국정보처리학회, 춘계학술발표대회 논문집, 2006. 5.

[6] 김우열, 김동호, 문소영, 김영철, "xUML을 사용한 MDA 기반 임베디드 소프트웨어 컴포넌트 시스템을 위한 설계 재사용", 한국 정보과학회, 추계학술발표대회 논문집, 제32권, 제2호, 2005. 11, pp.475-477.

[7] Jean-Louis Houberdon, Jean-Philippe Babau, "MDA for embedded systems dedicated to process control," Workshop on MDA in SIVOEES, in conjunction with UML'2003, October 2003.