DOI: 10.3745/KIPSTD.2008.15-D.3.355

이종 임베디드 시스템의 멀티태스킹을 위한 MDA(Model Driven Architecture) 기반의 설계

손 현 승[†] · 김 우 열[††] · 김 영 철[†††]

요 약

복잡한 임베디드 시스템의 멀티태스킹 지원은 실시간 운영체제가 요구된다. 이종의 임베디드 시스템 개발 환경에서 각각의 시스템에 최적화된 운영체제와 프로세서를 사용한다. 본 논문에서는 이종 임베디드 시스템 개발 시 기존의 크로스 컴파일러 대신, 운영체제의 API 정보 및 프로세서 레지스터 구성 정보의 UML 프로파일화 방식을 제안한다. 이는 각각의 임베디드 시스템에 적합한 프로파일을 이용해 이종의 시스템 개발 환경을 선택하여 자동 코드 발생을 통해 개발 기간 및 비용을 단축할 수 있다. 적용사례로서 이종 시스템 프로파일 정보를 이용해 이종의 실시간 운영체제 (brickOS와 uC/OS-II) 및 프로세서(Hitachi H8과 Intel PXA255)에 맞는 모델 및 코드를 생성하여 포팅 하였다.

키워드 : 통합 모델링 언어, 실행 가능 UML, 실시간 운영체제, 임베디드 시스템, UML 프로파일, 모델 기반 아키텍처, 다중처리

MDA(Model Driven Architecture) based Design for Multitasking of Heterogeneous Embedded System

Hyun Seung Son[†] · Woo Yeol Kim[††] · R. Young Chul Kim[†††]

ABSTRACT

The complicated embedded system for multi-tasking requires RTOS(real-time operating system). It uses the optimal OS and processor to each embedded system on the heterogeneous development environment. This paper is proposed to use UML profile of OS API and Processor Configuration, instead of cross-compiling for developing the heterogeneous embedded system. This reduces the development time and cost through generating the automatic source code with the profile information of each embedded system. We generate and port the code after modeling the two heterogeneous real time operating systems (brickOS and uC/OS-II) and the processors (Hitachi H8 and Intel PXA255) with our proposed profile of the heterogeneous embedded system.

Key Words : Unified Modeling Language, xUML(Executable UML), RTOS, Embedded System, UML Profile, Model Driven Architecture, Multi tasking

1. 서 론

사용자의 요구사항 증가는 임베디드 시스템을 복잡하게 만들고 있다. 또한 여러 기능들이 한 가지 자원을 사용하기 때문에 자원의 배분도 중요하다. 그래서 하드웨어의 제약이 매우 심하다. 실시간 운영체제[1, 6]를 적용하면 쓰레드 단위로 프로그래밍 하여 분할 설계가 가능하기 때문에 복잡성을 줄일 수 있다. 그렇지만 시스템 개발 시 어떤 운영체제가 적용되어야 될지 선택하기가 어렵다. 그래서 임베디드 시스템에 적합한 실시간 운영체제를 지원할 수 있도록 개발 환경이 구축되어야 한다. 이렇게 점점 복잡해져가는 시스템의 요구사항과 빠른 개발을 위해서는 기존의 임베디드 개발 방법으로는 부족하다.

기존 MDA[2, 3, 11, 14]는 플랫폼 독립적인 모델을 만들고 이를 종속적인 모델로 변환하는 방법으로 시스템을 개발한다. 이때 플랫폼 독립적인 모델을 재사용하여 다른 종류의 플랫폼으로 자동 변화하여 시스템을 빠르게 개발 할 수 있다. MDA의 핵심은 모델 변환에 있다. 이 모델을 변환하기 위해서 사용되는 것은 MOF, XMI, CWM, UML 프로파일, QVT가 사용된다[4]. 이들 모두 사용되지만 변환을 위한 핵심은 UML 프로파일 이다. UML 프로파일은 각각의 시스템 정보를 프로파일화 하여 모델의 변환이 가능하게 해준다.

※ 본 연구(2007년~2008년)는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업(2007013011430)으로 수행된 연구결과임.
† 준 회 원 : 홍익대학교 일반대학원 석사과정
†† 준 회 원 : 홍익대학교 일반대학원 박사과정
††† 정 회 원 : 홍익대학교 컴퓨터정보통신공학과 교수
논문접수 : 2008년 1월 4일
수 정 일 : 1차 2008년 3월 3일, 2차 2008년 4월 1일
심사완료 : 2008년 4월 1일

하지만 기존의 UML 프로파일은 임베디드 시스템에는 적합하지 않다.

본 논문에서는 기존의 UML 프로파일을 임베디드 시스템에 적용하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련연구로 UML 프로파일과 개발 프로세서에 대해서 알아본다. 제 3장에서는 UML 프로파일을 임베디드 시스템 환경에 맞도록 정의한다. 제 4장에는 적용사례로 이종의 임베디드 시스템에 UML 프로파일을 적용한다. 마지막으로 제 5장에서는 결론 및 향후연구를 언급한다.

2. 관련연구

2.1 기존 UML 프로파일

UML은 매우 일반적인 언어이기 때문에, 특정한 프로그래밍 언어(Java, C++, 등)의 개념을 표현하거나 혹은 특정한 애플리케이션 도메인(금융, 항공우주, 전자상거래, …)의 개념을 표현하기에는 부족하다. 이런 경우 스테레오타입과 확장 속성을 사용할 수 있는데, 이것을 체계적인 형태로 정의해서 하나의 패키지로 만든 것이 바로 UML 프로파일[10]이다.

UML 프로파일은 특정 요소를 여러 가지 관점에서 확장할 수 있도록 해 준다. 스테레오타입(stereotype)을 통해 요소들을 분류할 수 있는 기준을 제공하고, 확장 속성(tagged value)을 통해 요소에 정의되지 않은 또 다른 속성을 정의할 수 있도록 도와준다. 그리고 제약조건(constraint)과 데이터 타입(data type)을 추가적으로 정의할 수 있도록 허용한다.

<표 1>는 UML 프로파일의 그래픽 표기법 이다. 프로파일내부에 스테레오 타입과 메타 클래스로 구성될 수 있고 확장을 통해서 각각을 연결시킨다.

〈표 1〉 UML 프로파일 그래픽 표기법

Node Type	Notation
Stereotype	«stereotype» **Name**
Metaclass	«metaclass» **Name**
Profile	«profile» **Name**
Extension	——▶
ProfileApplication	«apply» - - - - - -≫

2.2 MDA 기반 임베디드 소프트웨어 개발 프로세스

MDA 기반의 임베디드 소프트웨어 개발 프로세스[14]는 요구사항분석, 타겟 독립 모델(TIM), 타겟 종속 모델(TSM), 타겟 의존 코드(TDC) 4가지 단계로 구성된다. 첫 번째 단계는 요구사항 및 분석단계로 요구사항에 대한 정보를 문서화하고 유스 케이스 다이어그램을 이용하여 요구사항을 모델링 한다. 두 번째 단계에서는 확장된 xUML[8,9,11]을 사용하여 타겟 독립 모델을 설계한다. 타겟 독립 모델은 하드웨어 플랫폼이나 운영체제에 의존적이지 않도록 시스템을 모델링 한다. 세 번째 단계는 타겟 종속 모델(TSM)로 타겟 독립 모델을 재사용하여 모델링 한다. 자동화 도구로 불가능한 부분을 추가하여 설계 한다. 마지막 단계는 타겟 의존 코드(TDC)로 최종 코드를 작성한다. 대부분 코드는 자동 생성되지만 생성할 수 없는 부분의 코드는 수작업을 통하여 작성한다. 생성된 소스코드를 타겟 모델에 탑재하는 작업이 바로 개발한 소프트웨어를 임베디드 소프트웨어 시스템에 적용시키는 것이다[10,11,14]. 모든 과정에서 생성된 소스코드와 모델은 재사용 저장소에 저장 후에, 기능을 변경하거나 추가하여 새로운 임베디드 시스템을 구축하고자 한다면 저장된 모델을 재사용한다. 자동 코드 생성 기법은 모델을 텍스트로 변환하는 방법[4]을 이용한다.

3. 제안한 임베디드 시스템용 UML 프로파일

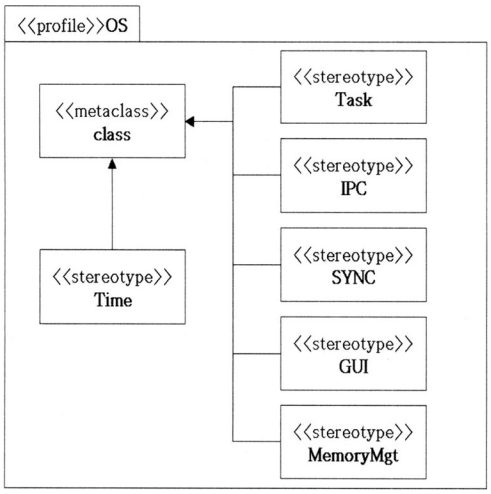
OMG에서 제안하는 MDA는 임베디드용이 아닌 커다란 정보시스템을 위한 개발 방법이다. 그렇기 때문에 기존의 MDA를 적용한 UML 도구들은 임베디드가 아닌 일반 응용 프로그램을 대상으로 하고 있다. 또한 UML 프로파일의 구성도 EJB, CORBA, EAI, .NET으로 이루어진다. 본 논문에서는 임베디드 시스템에 적용가능 하도록 UML 프로파일을 확장하였다.

임베디드 시스템에서 멀티태스킹이 가능하기 위해서는 기본적으로 운영체제가 요구된다. 그리고 운영체제API 정보와 프로세서의 레지스터 구성 정보가 필요하다. 본 논문에서 제안한 UML 프로파일은 임베디드 시스템에서 사용할 수 있도록 운영체제 API 프로파일과 프로세서 구성 프로파일이 사용 된다.

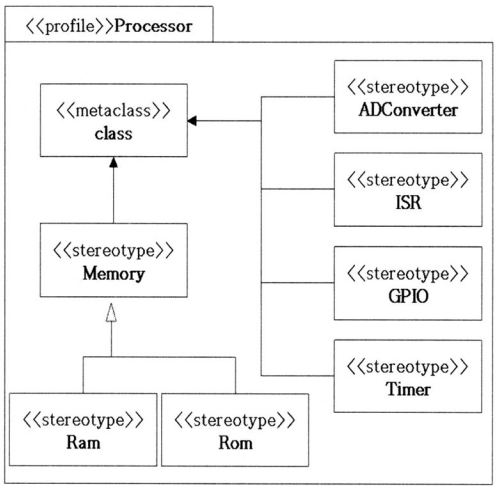
3.1 운영체제의 API 프로파일

운영체제에서 제공해주는 API를 메타데이터화 하여 운영체제프로파일을 만든다. 운영체제의 API 프로파일은 (그림 1)과 같다. 스테레오 타입의 기본형은 클래스로 하고 여기에 Task, IPC, SYNC, GUI, MemoryMgt, Time으로 총 6가지로 나누었다.

Task는 실시간 운영체제에서 가장 기본적인 단위이며 작업분할 단위로도 사용 하게 된다. 태스크는 쓰레드처럼 병렬 수행이 가능하고 같은 쓰레드끼리 메모리영역을 공유하게 된다. Task는 태스크의 생성, 삭제, 일시 정지등의 작업을 수행한다.



(그림 1) 운영체제 API의 메타 프로파일



(그림 2) 프로세서 구성 프로파일

IPC(Inter Process Commination)는 태스크와 태스크 사이의 통신을 담당 한다. 이러한 통신을 위한 수단으로 메일박스, 메시지 큐, 파이프를 제공한다.

SYNC(Synchronization)은 공유데이터 문제와 동기화를 위해서 사용된다. 공유데이터나 동기화문제를 해결하기 위해서 세마포, 뮤텍스, 이벤트가 제공된다.

GUI는 디스플레이 장치에 화면을 출력하게 해준다. 두 가지 종류의 디스플레이 장치 중 한 가지가 선택될 수 있는데 첫 번째는 텍스트 기반의 LCD, 두 번째는 그래픽 기반의 LCD 이다. Text 기반의 LCD는 비교적 하드웨어 처리가 간단하며 GUI구조도 매우 간단하게 구성시킬 수 있다. 주 명령어는 화면지우기, 커서 이동, 몇 번째 라인에 글자 출력 등을 가진다. 반대로 그래픽 기반의 LCD는 모든 처리를 픽셀단위로 처리해야 되기 때문에 매우 복잡해진다. 비교적 쉬운 처리를 위해 프레임 버퍼를 할당하고 여기에 그래픽 처리 알고리즘을 이용하여 화면에 표시될 대상을 그려주면 자동으로 화면에 표시될 수 있도록 구성되어 진다.

MemoryMgt(Memory Management)는 메모리 할당, 해제를 수행한다. 주로 동적메모리 관리를 담당하게 된다.

마지막으로 Time는 시간관련 함수로 시간 지연, 타이머 콜 등의 기능을 수행 한다.

3.2 프로세서 레지스터 구성(Configuration) 프로파일

프로세서 프로파일은 하드웨어 중 CPU의 아키텍처에 매우 의존적이다. 제조사 마다 서로 다른 아키텍처를 사용하기 때문에 이를 하나의 메타모델로 구성 하기는 매우 어렵다. 그래서 몇 가지 중요한 기능들만 프로파일로 사용하려고 한다. 프로세서 프로파일을 그래픽 표기법을 사용하여 표현한 것이 (그림 2)이다.

프로세서 프로파일은 운영체제 프로파일의 구현부로 볼 수 있다. 프로세서 프로파일은 Memory, ADConverter, ISR, GPIO, Timer 로 총 5가지로 나누었다. Memory는 메모리의 읽기와 쓰기 연산이 가능하도록 지원해준다. ADConverter (Analog Digital Converter)는 아날로그 장치와의 구성을 위해서 필요하게 된다. 아날로그 데이터를 디지털 데이터로 변환해 준다. ISR(Interrupt Service Routine)은 모든 장치들을 주기적으로 체크해야 하는 폴링 기반 시스템으로부터 해방 시켜준다. 시스템은 어떠한 이벤트가 발생되었을 때 처리해야 되는 ISR만 작성하면 주기적으로 하드웨어를 검사하지 않아도 된다. 보통 255가지의 인터럽트를 처리할 수 있도록 해주는데 이것은 CPU 아키텍처마다 다르다. 그래서 중요한 몇 가지 인터럽트만 처리해주고 확장할 수 있도록 해준다. GPIO(General Purpose In/Out)는 프로세서가 가지는 외부 핀을 통해 다른 하드웨어와 연결시켜 준다. GPIO 통해서 핀들을 관리하여 주면 프로세서에 관계없이 모든 핀들을 자유롭게 사용할 수 있다. 마지막으로 Timer는 시간을 처리해주는 역할을 수행한다. 프로세서 내부 혹은 외부의 타이머를 이용하여 시간을 관리 한다. 주로 ISR과 함께 사용되어진다.

4. 적용사례

4.1 이종의 임베디드 시스템

MindStorm[12]의 하드웨어 정보와 Empos-II[13]의 하드웨어 정보를 살펴본다. MindStorm의 CPU는 Hitachi사의 H8/300 씨리즈인 3292로 되어 있고 클럭은 16MHz로 작동된다. RAM은 32Kbyte, ROM은 16Kbyte의 저장공간을 가진다. 호스트(PC) 시스템에서 타겟(RCX) 시스템으로 데이터를 전송하기 위해서 적외선 직렬 포트 장치인 LEGO

USB Tower를 사용 한다. 화면 출력장치로는 작은 크기의 LCD를 사용한다.

Empos-II[13]의 CPU는 Intel사에서 제공하는 ARM기반의 PXA255이고 클럭은 400Mhz로 작동하게 된다. RAM은 64Mbyte ROM은 32Mbyte로 RCX와 비교했을 때 엄청난 성능을 가지고 있다. 또한 데이터를 전송하기위해 직렬 통신 포트를 사용하고 리눅스에서 제공하는 터미널인 minicom을 사용하여 데이터를 주고받는다.

4.2 타겟 독립 모델 설계(Target Independent Model)

타겟 독립적인 모델은 기능적 요구사항을 시스템에 적용하는 첫 번째 단계이다. 타겟 독립적 모델은 어떠한 하드웨어나 소프트웨어에 의존적이지 않도록 모델링 한다. 이 과정에서 생성되는 모델은 클래스 다이어그램, 병렬 메시지 다이어그램, 병렬 상태 다이어그램이다. 타겟 독립 모델을 완성하면 타겟 종속 모델로 변환을 할 수 있다. 이때 원하는 프로세서와 운영체제를 선택하여 타겟 종속 모델로 변환하게 된다.

아래는 각 다이어그램에 대한 부가적인 설명을 언급한다.

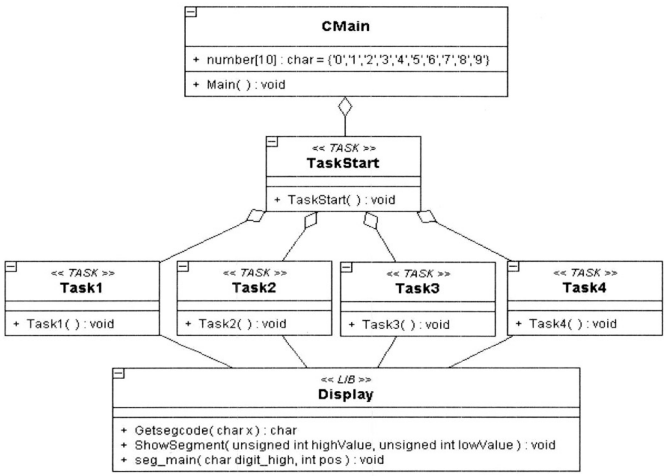
4.2.1 클래스 다이어그램

본 예제를 TIM단계에서 클래스 다이어그램을 모델링한 것이 (그림 3) 이다.

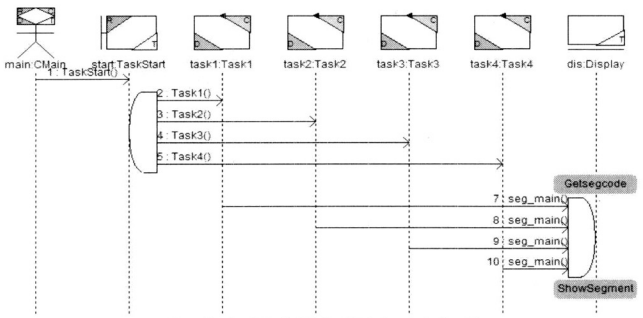
CMain 클래스에는 TaskStart라는 태스크가 있고 TaskStart 클래스에는 7세그먼트를 증가시키는 태스크들을 가지고 있다. 일반적인 클래스와 구별이 가능해야 한다. 그래서 멀티태스킹을 사용하여 독립적인 태스크로 수행되는 부분은 스테레오 타입 "<<TASK>>"로 표현하고 사용되는 라이브러리는 "<<LIB>>"로 표현 한다.

4.2.2 병렬 메시지 다이어그램

TIM단계에서 병렬 메시지 다이어그램[14]을 모델링한 것이 (그림 4)이다. 기존의 UML에서 시퀀스 다이어그램은 객체간의 동적 상호작용을 시간적 개념을 중시하여 모델링한다. 하지만 병렬 메시지 다이어그램은 기존의 다이어그램에서는 가능하지 않은 병렬 적인 개념과 롤 기반 모델링이 가능하다.



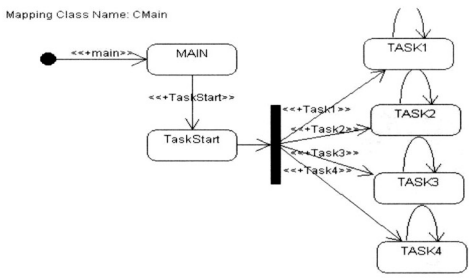
(그림 3) TIM의 클래스 다이어그램



(그림 4) TIM의 병렬 메시지 다이어그램

4.2.3 병렬 상태 다이어그램

태스크들의 수행상태는 최초 시작이 되면 MAIN상태로 전이되고 바로 TaskStart상태가 되어 각 태스크들을 병렬적으로 수행하도록 한다. 이것을 병렬 상태 다이어그램으로 표현한 것이 (그림 5)이다.



(그림 5) TIM의 병렬 상태 다이어그램

4.3 타겟 종속 모델 설계(Target Specific Model)

타겟 종속 모델은 타겟 독립적인 모델을 프로세서와 운영체제 프로파일을 적용하여 생성 한다. 프로파일의 매핑규칙에 의하여 독립모델이 종속 모델로 변하는 것이다. 다이어그램 상에서는 타겟 독립모델을 객체지향의 상속개념처럼 이어받게 된다. 그렇기 때문에 다이어그램 자체가 유사하다. 하지만 내부 속성들이 달라졌기 때문에 같다고는 할 수 없다. 이 단계는 자동화도구로는 불가능한 부분을 수작업을 통해서 완성한다.

다음은 생성된 타겟 종속 모델을 설명한 것이다.

4.3.1 MindStorm(brickOS)

(그림 5)에서 만든 타겟 독립 모델을 가지고 자동화 도구를 통해 MindStorm 형태로 변환한다. 변환된 모델은 brickOS의 속성으로 변환된다. 타겟 독립 모델의 구조 그대로 타겟 종속 모델로 변환된다. 이때 프로파일 정보들은 도구 내부에 숨겨져 있으므로 외관상으로는 확인하기 어렵다. 실제적인 구분은 타겟 의존 코드에서 확인이 가능하다.

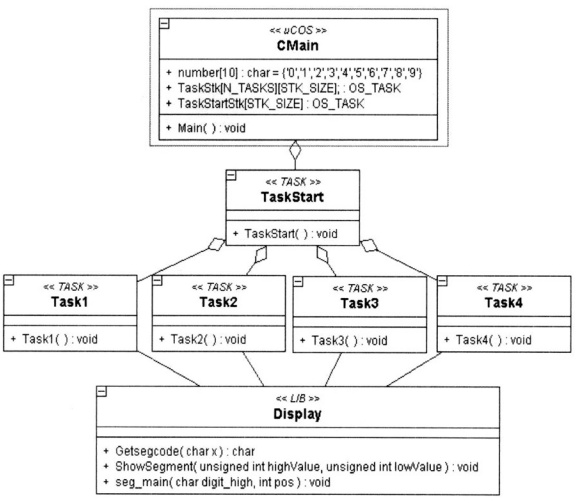
4.3.2 EMPOS-II(uC/OS-II)

EMPOS-II시스템에 맞도록 타겟 독립 모델을 변형한다. 타겟 독립 모델에서 타겟 종속 모델로 변환될 때 자동화 도구로 불가능 한 부분은 사용자가 추가한다. CMain 클래스의 속성부분에 OS_STK TaskStk [N_TASKS][STK_SIZE], OS_STK TaskStartStk[STK_SIZE]를 (그림 6)의 빨간 부분과 같이 추가 한다.

4.4 타겟 종속 코드 생성

타겟 종속 모델을 완성하면 코드자동생성기법[15]을 사용하여 C, C++, Java 중 원하는 코드를 생성 할 수 있다. 다이어그램들의 메타모델과 제시한 운영체제와 프로세서 프로파일을 사용하여 코드를 생성하게 된다. 레고의 RCX는 커널과 응용프로그램 코드가 분리된 방식을 사용한다. 그래서 먼저 커널을 RCX에 포팅 하여 brickOS가 탑재되어 수행되고 있어야 한다. 응용 프로그램을 처리하기 위해서 brickOS 내부에 idle 태스크를 포함 하고 있다. brickOS가 제공해주는 라이브러리인 cputc()함수를 사용하여 화면 출력한다.

uC/OS-II는 brickOS와 다르게 커널과 응용 코드가 같이 포함되어 있는 이미지로 생성된다. 그래서 응용 프로그램 내부에 커널소스가 혼합된 형태로 개발된다. 또한 idle 태스크가 응용프로그램에 태스크 시작 루틴에 포함된다. uC/OS-II는 라이브러리가 제공되지 않으므로 직접 만든 seg_main()함수를 이용하여 화면 출력한다.



(그림 6) uCOS의 타겟 종속 모델

5. 결 론

날로 복잡해져가는 임베디드 시스템이 멀티태스킹을 지원하기 위해선 실시간 운영체제의 사용이 필요하며, 이종의 임베디드 시스템 개발 환경에서는 각각의 시스템에 적합한 실시간 운영체제를 사용한다. 본 논문에서는 기존 MDA 기반 개발의 핵심인 UML 프로파일을 임베디드 시스템에 적합하도록 변환하여 임베디드 시스템 개발 시 실시간 운영체제 API 및 프로세서 구성 정보의 프로파일화에 대해 제안한다. 각각의 임베디드 시스템에 적합한 프로파일을 사용한다면 최적화된 개발환경 선택을 통해 개발 기간 및 비용을 단축할 수 있다. 적용사례로서 이종의 실시간 운영체제 및 프로세서를 사용하는 임베디드 시스템에 프로파일을 적용하여 개발기간의 단축할 수 있다.

향후 연구과제로 빠른 개발뿐만 아니라, 레거시 시스템에 적용할 경우에는 얼마나 신뢰성 있는 기능으로 오류 없이 사용할 수 있는 재사용성의 신뢰성에 대한 문제 해결과 확장된 UML 프로파일이 좀 더 범용 적으로 사용될 수 있도록 프로파일 정보의 최적화를 위한 연구가 진행 중이다.

참 고 문 헌

[1] Qing Li, 'Real-Time Concepts for Embedded Systems,' CMP, 2003.

[2] Pierre Boulet, Jean-Luc Dekeyser, Cedric Dumoulin, and Philippe Marquet, "MDA for Soc Design, Intensive Signal Processing Experiment," In FDL'03, Frankfurt, ECSI, September, 2003.

[3] A. Kleppe, J.Warmer, W.Bast, 'MDA Explained: The Model Driven Architecture: Practice and Promise,' Addison-Wiseley, 2003.

[4] OMG, http://www.omg.org/technology/documents/modeling_spec_catalog.htm

[5] Alessandro Rubini, 'Linux Device Drivers,' OReilly, 2005.

[6] David E. Simon, 'An Embedded Software Primer,' Addison-Wesley, 1999.

[7] OMG, UML Superstructure, v2.1.1, OMG document formal/07-02-03.

[8] Steve Mellor, Marc Balcer, 'Executable UML: A Foundation for Model-Driven Architecture, Addison- Wesley,' (2002)

[9] Leon Starr, 'Executable UML: How to build class models,' Prentice Hall PTR,(2002)

[10] 김우열, 김영철, "Adapting Model Driven Architecture for Modeling Heterogeneous Embedded S/W Components," ICHIT2006, Vol.2, 2006. 11.

[11] 김우열, 김영철, "A Study on Modeling Heterogeneous Embedded S/W Components based on Model Driven Architecture with Extended xUML," KIPS Trans: Part D, Vol.14-D, No.1, 2007. 2.

[12] LEGO, Mind Storms, http://mindstorms.lego.com/

[13] Hanback, EMPOS-II, http://www.hanback.co.kr/

[14] 김동호, 김우열, 김영철, "A Study on Design for Embedded S/W based on Model Driven Architecture," IWIT, Vol.6, No.1, 67-74, 06.03.

[15] 손현승, 김우열, 서윤숙, 김동호, 김동우, 김재수, 김영철, "이종 임베디드 소프트웨어를 위한 코드 생성 메커니즘 및 지원도구," 한국소프트웨어공학회, Vol.9, No.1, 170-177, 07.02.22

[16] ISO/IEC 9126, 'Software engineering : Product quality,' http://www.iso.org.

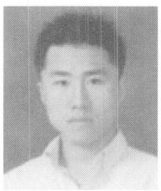
손 현 승

e-mail : son@selab.hongik.ac.kr

2007년 홍익대학교 컴퓨터정보통신(학사)

2007년~현 재 홍익대학교 일반대학원 소프트웨어공학전공(석사)

관심분야 : 임베디드 소프트웨어 자동화 도구 개발, 임베디드 RTOS 개발, 임베디드 MDA (Model Driven Architecture) 연구, 모델 검증 기법 연구



김 우 열

e-mail : john@selab.hongik.ac.kr

2004년 홍익대학교 컴퓨터정보통신(학사)

2006년 홍익대학교 일반대학원 소프트웨어공학전공(석사)

2006년~현 재 홍익대학교 일반대학원 박사과정

관심분야 : 상호운용성, 임베디드 소프트웨어 개발 방법론 및 도구 개발, 컴포넌트 시험 및 평가, 리팩토링

김 영 철

e-mail : bob@selab.hongik.ac.kr

2000년 Illinois Institute of Technology (공학박사)

2000년~2001년 LG 산전 중앙연구소 Embedded system 부장

2001년~현 재 홍익대학교 컴퓨터정보통신 부교수

관심분야 : 테스트 성숙도 모델, 임베디드 S/W 개발 방법론 및 도구 개발, 모델 기반 테스팅, CBD, BPM, 사용자 행위 분석 방법론