ESUML: UML 기반 임베디드 소프트웨어 모델링 방법론+*

전상욱° 이희진, 홍장의*, 배두환
한국과학기술원 전자전산학과
*충북대학교 전기전자컴퓨터공학부
{sujeon°, hjlee, bae}@se.kaist.ac.kr, *jehong@chungbuk.ac.kr

ESUML: UML-based Modeling Method for Embedded Software

Sang-Uk Jeon° Hee Jin Lee, Jang-Eui Hong*, Doo-Hwan Bae
Division of Computer Science, KAIST
*School of Electric and Computer Engineering, Chungbuk National University

요 약

소프트웨어 개발을 위한 분석 및 설계 언어로 UML이 일반화 되어 있다. UML이 갖는 객체지향 개념의 장점으로 인하여 일반 소프트웨어 개발뿐만 아니라 임베디드 소프트웨어 개발에 있어서도 UML의 사용이 증가하고 있다. 특히 UML 모델 중에서 시퀀스 다이어그램은 시스템의 다이나믹스 및 동적 시나리오 표현에 매우 유용하고, 모델링이 직관적이어서 소프트웨어 엔지니어들이 선호하고 있다. 본 연구에서는 시스템 행위를 중심으로 하는 임베디드 소프트웨어 모델링 방법으로 ESUML 방법론을 제시한다. 제시한 ESUML 방법론에서는 Use Case, Class, Interaction overview, Sequence 다이어그램과 Action Language를 이용하여 시스템을 효과적으로 모델링 하도록 하였다.

1. 서 론

임베디드 소프트웨어의 분석 및 설계를 위해서는 개발자가 쉽게 접근할 수 있는 모델링 방법이 필요하다. 또한 모델링 절차가 간단하고 직관적이어야 한다. 전문적 지식이 많이 요구되는 방법론 또는 모델링 기법은 개발 과정에서의 적용을 회피하거나, 적용 결과에 대한 품질이 저하될 수 있기 때문이다. 또한 임베디드 소프트웨어는 궁극적으로 하드웨어 플랫폼상에 탑재되어야 한다. 탑재된 소프트웨어에 문제가 발생하게 되면, 최악의 경우 하드웨어 플랫폼 자체를 다시 개발해야 하는 문제까지도 발생할 수 있기 때문에 요구 사양을 기준으로 하는 신뢰성있는 소프트웨어 모델링이 매우 중요하다.

ESUML(Embedded Software with UML) 방법론은 UML 2.0을 기준으로 임베디드 소프트웨어를 분석, 설계하고자 하는 목적으로 개발되었다. ESUML은 요구사항의 확정 단계부터, 분석 및 설계, 그리고 코드 생성 과정을 지원하는 방법론이다. 특히 제시하는 방법론에서는 이벤트 기반의 시퀀스 다이어그램을 중심으로 시스템의 행위를 모델링 함으로써, 직관성과 적용성을 향상시킨 방법론이라고 할 수 있다.

2. 임베디드 소프트웨어 개발 방법론

임베디드 소프트웨어를 개발하기 위한 개발 방법론은 여러 가지가 있다. 그 중에서 구조적 분석 및 설계 방법론은 가장 오래되고, 개념이 상대적으로 쉬운 체계적인 개발 방법론이다. 이 방법론에서는 시스템 요구사항을 기능적 분할을 통해 자료흐름도로 표현하고, 이를 태스크로 묶는 방법을 제공하고 있다. 임베디드 소프트웨어가 대부분 C 언어를 기반으로 개발하고 있는 상황에서 아직도 널리 적용되고 있는 방법 중의 하나이다[1].

객체지향 개념을 지원하는 임베디드 소프트웨어 개발 방법론의 대표적인 것은 H. Gomaa가 제안하는 COMET 방법론이다[2]. 이 방법론은 병렬 실시간 특성을 갖는 시스템을 개발하도록 지원한다. 이외에도 시스템에 대한 구조적 모델, 기능적 모델, 그리고 동적 모델을 구분하여 시스템을 모델링 하는 OMT와 OCTOPUS 방법론[3], 그리고 실시간 특성을 반영하는 Real-Time UML 방법론[4]이 있다.

객체지향 개념을 갖는 이들 방법론들은 시스템을 표현하기 위한 다양한 방법의 제공으로 풍부한 시스템 모델링이 가능하지만, 방법론의 적용이 무겁고, 구체적인 개발 가이드라인이 제공되지 못하여 방법론 정착까지는 오랜 시간이 걸린다는 단점을 갖는다.

최근에 CASE 도구와 함께 제공되는 임베디드 소프트웨어 개발 방법들은 단순한 클래스 다이어그램과 상태차트만을 이용하여 소프트웨어를 개발하기도 한다[5]. 그러나 상태차트가 소프트웨어 엔지니어들이 쉽게 작성하기 어려운 모델이기 때문에 반복적인 수정이 지나치게 많아진다는 단점이 있다. 따라서 본 연구에서는 UML 2.0을 근간으로 하는 직관적이며 경량의 방법론인 ESUML 임베디드 소프트웨어 개발 방법론을 제시하였다.

3. 행위 중심의 모델링

ESUML 방법론은 임베디드 소프트웨어에 대한 요구사항

+ 본 연구는 정보통신부 정보통신연구진흥원이 지원하는 선도기반기술 개발과제의 지원을 받았음.

을 표현하기 위한 Use Case 다이어그램, 소프트웨어의 정적 구조를 표현하기 위한 클래스 다이어그램, 그리고 시스템의 동적 구조를 사용하기 위해 IOD(Interaction Overview Diagram), 시퀀스 다이어그램을 사용한다[6].

3.1 Global Behavior

임베디드 소프트웨어 요구사항은 분석과정에서 IOD에 나타난다. 즉, IOD는 시스템 수준에서의 제어 구조 및 흐름을 표현한다. UML에서의 Activity 다이어그램과 유사한 IOD는 외부로부터의 이벤트 입력과 외부로의 출력, Fork 및 Join, Merge 등과 같은 시스템 행위를 쉽게 표현할 수 있다. 또한 Activity Node가 하나의 시퀀스 다이어그램을 포함하는 프레임으로 정의될 수 있기 때문에 계층적인 시스템 모델링을 가능하게 한다.

ESUML에서는 Use Case 다이어그램에 나타난 각각의 Use Case 단위로 IOD를 작성하는데, 작성된 IOD들은 액터와 Use Case간의 관계를 기준으로 상위의 IOD 다이어그램 하나로 통합된다.

3.2 Local Behavior

IOD의 노드로 나타나는 참조 프레임(Ref frame)은 내부의 행위를 표현하기 위해 시퀀스 다이어그램으로 모델링된다. UML 2.0에서의 시퀀스 다이어그램은 기존의 MSC(Message Sequence Chart)가 제공했던 다양한 동적 특성의 표현 방법을 포함하였으며, 행위 제어에 대한 풍부한 표현력의 제공으로 임베디드 소프트웨어를 모델링하기 위한 실시간성, 병렬성, 이벤트 처리 등과 같은 특성을 잘 모델링할 수 있다[6]. 특히 ESUML에서는 시퀀스 다이어그램이 시스템의 하위 행위를 표현하도록 하였으며, 또한 Action Language를 이용하여 정확한 행위를 표현하도록 하였다[7].

4. ESUML 방법론

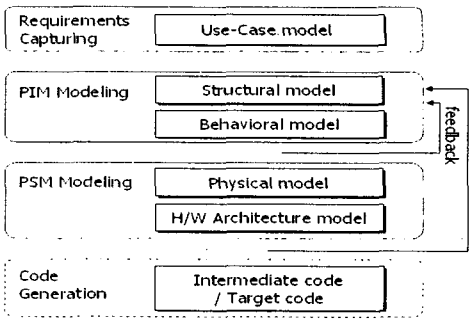
ESUML은 임베디드 소프트웨어를 개발하는 산업체의 엔지니어가 쉽게 적용할 수 있도록 정의된 UML 2.0 기반 개발 방법론이다. ESUML 방법론이 지원하는 특성을 살펴보면 다음과 같다.

4.1 지원 개념(Concepts)

ESUML 방법론은 임베디드 소프트웨어의 개발을 위한 개념을 지원한다. 이벤트 중심의 모델링과 행위 중심의 모델링 가능하도록 하였다. 특히 가전이나 휴대 단말기 등의 분야에서 유용하게 사용될 수 있을 것이다.

4.2 절차(Process)

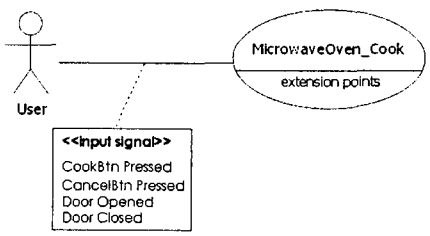
ESUML은 모델 기반 접근 방식을 지원한다. 도메인 모델의 융통성과 재사용성을 높이고, 하드웨어 플랫폼의 변경에 의한 영향을 최소화하기 위하여 PIM, PSM 모델링 단계를 갖는다. <그림 1>은 ESUML 모델링 절차를 보여준다.



<그림 1> ESMUL 모델링 절차

4.2.1 Requirements Capturing 단계

요구사항 획득 단계는 임베디드 소프트웨어에 대한 요구사항을 Use Case 다이어그램으로 모델링하고, 이로부터 성공 실패 시나리오를 도출하는 단계이다. ESUML 방법론에서는 Use Case 계층화 모델링을 지원하며, 특히 Use Case와 액터와의 관계를 표시할 때, 이들 관계에 스테레오타입의 입력 정보를 그림 2와 같이 기술한다.



<그림 2> Use Case 모델링 예제

4.2.2 PIM 모델링

PIM 모델링 단계에서는 대상 시스템에 대한 정적구조와 동적 행위를 모델링 하는데, 정적구조는 클래스 다이어그램에 의해서, 동적 행위는 IOD와 시퀀스 다이어그램에 의해서 표현된다. 동적 행위 모델링은 세부적으로 추상화 모델링과 상세화 모델링으로 구분하고, 추상화 모델링은 IOD로, 상세화 모델링은 시퀀스 다이어그램으로 표현한다.

동적행위 모델링을 위해 사용되는 IOD와 시퀀스 다이어그램에는 추가적인 정보 기술을 위하여 Action Language를 사용하며, 다수의 Use Case에 대한 IOD 통합 모델은 그림 3과 같이 모델링된다.

4.2.3 PSM 모델링

PSM 모델링 단계에서는 소프트웨어에 대한 물리적 모델과 하드웨어 아키텍처 모델을 작성한다. 소프트웨어 물리적

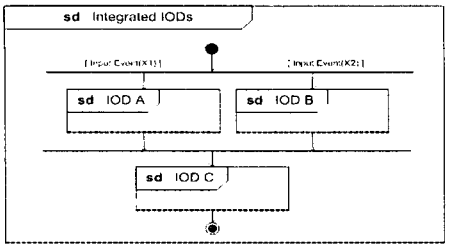


그림 3. IOD 통합에 의한 광역행위 표현

모델이라 함은 IOD 모델을 그림 4와 같이 병렬성 및 처리 특성을 고려한 소프트웨어 분할 모델이다. 대상 소프트웨어가 다중 처리기를 갖는 시스템에 탑재되는 경우 다수의 태스크로 분할되고 해당 하드웨어 컴포넌트에 할당된다. 하드웨어 아키텍처는 처리기, 메모리, 버스 구조 등의 물리적 레이아웃을 표현하는 모델이다.

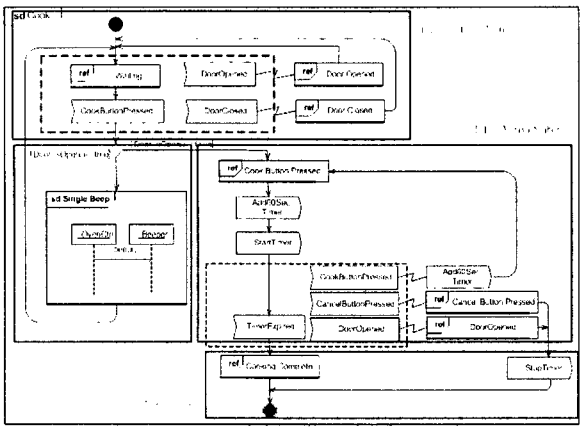


그림 4. ESUML: 소프트웨어 물리적 모델

4.2.4 코드 생성

코드 생성단계는 생성된 모델로부터 플랫폼에 독립적인 중간 코드(intermediate code)로 생성하거나 최종 타켓 코드로 생성한다. 중간 코드는 소프트웨어가 탑재될 운영체제나 디바이스 소프트웨어에 독립적인 코드를 생성하도록 지원하기 위함이며, 타켓 언어가 C++, Java 등과 같은 객체지향 언어나, C 언어와 같은 절차적인 언어 모두를 지원하기 위한 목적으로 생성한다.

4.3 단계별 산출물(Artifacts)

ESUML 방법론에서 지원하는 단계별 산출물에 대한 정의는 표 1과 같다.

5. ESUML 방법론 지원 도구 개발

<표 1> ESUML 단계별 산출물

단계	산 출 물
Req. Capturing	- Use case diagram - Scenario description
PIM modeling	- Class diagram - Interaction overview diagram - Sequence diagram
PSM modeling	- Hardware architecture model - Partitioned physical software model
Code Generation	- intermediate code

ESUML 방법론을 지원하기 위한 도구의 개발이 진행되고 있다. 본 방법론의 지원은 Window XP 환경하에서 JDK 1.5 버전을 가지고 구현하고 있다.

개발 중인 지원도구에서는 단순히 소프트웨어 모델링 기능이외에도 모델에 대한 검증을 수행하기 위해 정적 분석과 시뮬레이션 기능을 추가하고 있다. 정적분석에서는 모델에 대한 일관성 및 상태기반 검증 방법을 지원하며, 시뮬레이션을 시나리오에 근거한 동적 행위 검증을 지원한다.

6. 결론

본 연구에서는 현재 진행 중인 정통부의 선도기반기술개발 과제의 연구 결과에 대한 사항을 제시하였다. 임베디드 소프트웨어 활용 영역의 확장으로 상용의 소프트웨어를 개발하려는 산업체에서 방법론 적용의 용이성과 편의성, 산출물의 재사용성 등을 고려한 방법론이라고 할 수 있다. 본 ESUML 방법론의 특징은 (1) IOD와 시퀀스 다이어그램을 중심으로 하는 시스템 모델링, (2) 보다 효과적이고 직관적인 모델링을 위한 UML 표기법의 확장 지원, (3) 다수의 모델링 방법론과의 통합을 위한 중간형태의 코드의 생성 지원 등이라 할 수 있다.

참고문헌

[1] M. Page-Jones, Structured System Design, Yourdon Press, 1988
[2] H. Gomaa, Designing Concurrent, Distributed and Real-Time Applications with UML, Addison Wesley, 2000
[3] M.Awad, J.Kuusela and J. Ziegler, Object-Oriented Technology for Real-Time Systems, Prentice Hall, 1996
[4] B. P. Douglass, Real-Time UML 3rd edition. Addison Wesley, 2004.
[5] Accelerated Technology, An Advanced Embedded System Programming using xtUML. Mentor Fraphics, http://www.acceleratedtechnology.com/
[6] OMG, UML 2.0 Superstructure Specification, Oct., 2004
[7] OMG, Action Semantics for the UML, OMG ad/2001-08-04, Aug. 2001