멀티 프로세서용 임베디드 소프트웨어의 PSM 모델링을 위한 이클립스 플러그인*

오기영°, 홍장의
충북대학교 전자계산학과 소프트웨어공학연구실
ohgy@selab.cbnu.ac.kr°, jehong@chugnbuk.ac.kr

An Eclipse Plug-In for Platform Specific Model of Embedded Software in Multiprocessor Environment

Gi-Young Oh°, Jang-Eui Hong
Chungbuk National University Department of Computer Science

요 약

멀티프로세서 환경에서 동작하는 임베디드 시스템을 개발하기 위해서는 소프트웨어 모델뿐만 아니라 하드웨어 플랫폼에 대한 모델이 필요하다. 이는 개발하고자 하는 소프트웨어가 하드웨어 플랫폼에 어떻게 배치되어 실행할 것인가에 대한 고려가 요구되기 때문이다. 특히 MPSoC(Multiprocessor SoC)에서는 소프트웨어를 배치할 하드웨어 플랫폼에 대한 정보가 필요하기 때문에 설계 과정에서 이들에 대한 모델링이 요구된다. 따라서 본 연구에서는 하드웨어 플랫폼 아키텍처를 정의할 수 있는 이클립스 기반의 플러그인을 개발하고, 이를 이용한 PSM 모델링 방안을 제시한다.

1. 서 론

최근 임베디드 소프트웨어의 적용 범위가 넓어지고, 응용 영역이 확장되면서 효과적이고 체계적인 임베디드 소프트웨어 개발 기술이 연구 제안되고 있다[5]. 이러한 연구들은 임베디드 소프트웨어의 아키텍처 모델링[11]을 비롯하여 제품 계열공학, 컴포넌트 재사용 및 패턴, 가상 프로토타입 환경에서의 임베디드 소프트웨어 검증 등 매우 다양한 영역에서 이루어지고 있다.

이러한 연구들 중의 한 분야는 임베디드 소프트웨어와 하드웨어를 통합 설계하는 부분이며, 특히 최근에는 모델 기반 아키텍처를 채택하는 임베디드 소프트웨어 개발에 많은 관심을 두고 있다[6]. 모델 기반 개발을 위해서는 임베디드 소프트웨어에 대한 요구 기능 중심의 PIM (Platform Independent Model) 모델링과 플랫폼 의존적인 PSM (Platform Specific Model) 모델링이 필요하게 된다. 그런데 일반적인 분산처리 환경에서의 PSM 모델들은 UML의 배치(Deployment) 다이어그램을 통해 소프트웨어 컴포넌트가 플랫폼 상에 어떻게 배치될 것인가를 표현하는데 반해 MPSoc 상에서 동작하는 임베디드 소프트웨어의 경우는 기존의 배치 다이어그램과는 다른 특성들이 포함되어 모델링 되어야 한다. 즉, 다양한 입출력 디바이스 장치를 비롯하여 처리기의 특성, 버스 및 메모리의 특성들이 하드웨어 아키텍처로 표현되어야 한다.

이를 위해서 본 연구에서는 하드웨어 아키텍처를 모델링하기 위한 이클립스 기반의 플러그인을 개발하고, 이를 이용하여 임베디드 소프트웨어의 PSM을 모델링 할 수 있도록 하였다. 개발된 플러그인은 하드웨어 아키텍처에 대한 메타 모델을 정의한 후에, 이클립스 RCP환경에서 다른 플러그인 즉, GEF (Graphical Editing Framework), EMF (Eclipse Modeling Framework) 등과 통합하여 임베디드 소프트웨어 모델링을 할 수 있도록 지원한다.

본 논문의 구성은 2장에서 기존 이클립스 플랫폼의 구조 및 플러그인들에 대하여 살펴보고, 3장에서는 임베디드 소프트웨어의 하드웨어 아키텍처를 모델링하기 위한 기본 요소들과, 이의 메타 모델에 대하여 설명한다. 4장에서는 정의된 하드웨어 아키텍처 플러그인을 이용한 비주얼 모델링 도구 개발에 대하여 설명하고, 5장에서는 결론 및 향후 연구내용에 대하여 제시하였다.

2. 이클립스와 플러그인

2.1 이클립스 플랫폼 구조

이클립스 플러그인은 이클립스 플랫폼에 연결되어 다른 플러그인에 영향이 없이 독립적으로 실행이 가능한 소프트웨어 컴포넌트이다. 그림 1은 이클립스 플랫폼이 여러 개의 플러그인과 연결되는 것을 보여준다. 이클립스 플랫폼은 내부적으로 워크벤치, 워크스페이스, 헬프, 팀 등으로 이루어져 있다[2].

워크벤치는 사용자 인터페이스 플러그인으로 응용프로그램을 개발하는데 있어서 메인 윈도우와 파일, 에디터, 윈도우, 도움말과 같은 최초의 풀다운 메뉴와 툴바를 만든다. 워크스페이스는 플러그인이 액세스할 수 있는 리소스에 대한 논리적인 컨테이너이다. 여기서 리소스 모델이란 워크스페이스에서 액세스할 수 있는 콘텐츠들을 논리적으로 표현한 것이다. 헬프는 이클립스에서 제공하는 도움말이다. 팀은 CVS (Concurrent Versions System)를 통해 소프트웨어 개발 작업을 관리하는데 사용하게 될 가장 일반적인 액션들을 요약한

* 본 과제는 정보통신부 선도기반기술개발사업의 지원으로 수행되었습니다.

것이다.

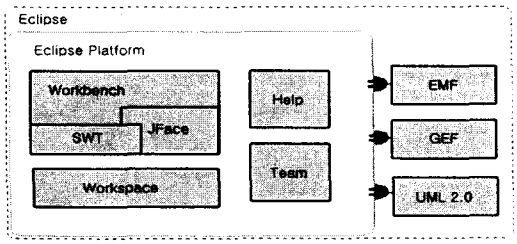

그림 1. 이클립스 구조

2.2 이클립스 플러그인

이클립스는 오픈 소스이며, 응용 도구 개발을 지원하기 위한 다양한 플러그인들을 제공하고 있다. 이클립스 사이트에서 제공되는 플러그인중 비주얼한 그래픽 에디터를 개발하기 위해 사용되는 주요한 플러그인은 다음과 같다[7].

- 이클립스 Workbench
- 이클립스 JFace
- 이클립스 SWT(Standard Widget Toolkit)
- 이클립스 EMF(Eclipse Modeling Framework)
- 이클립스 GEF(Graphical Editing Framework)
- 이클립스 UML 2.0

이중에서 그래픽 에디터 개발의 핵심적인 역할을 수행하는 EMF, GEF 플러그인에 대하여 보다 상세하게 살펴보면 다음과 같다.

(1) EMF(Eclipse Modeling Framework)

EMF는 메타모델을 정의하여 메소드를 자동으로 생성하고 관리할 수 있게 하는 플러그인이다[8]. 여기서 메타모델이란 그림 2에서 보는바와 같이 각 구성요소들의 속성들을 정의해 놓은 것이다. 예를 들어 책(Book)이라는 구성요소는 제목(title), 페이지수(pages), 카테고리(category)라는 속성을 가지고 있고, 필자(Writter)라는 구성요소는 이름(name)과 우편번호(zipCodes)라는 속성을 가지고 있다. 메타 모델을 정의하는데 있어서 중요한 고려 사항은 메타 모델이 모든 프로그래밍 언어에 적용할 수 있어야 하고 다수의 사람들에 의해 이해될 수 있는 형태여야 한다는 것이다.

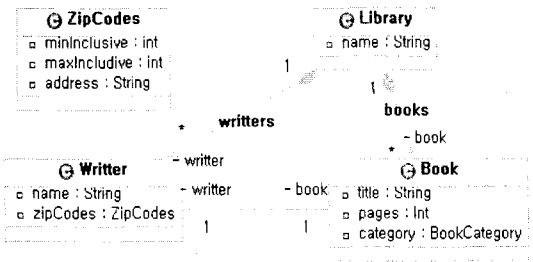

그림 2. 도서관리 시스템 메타모델

EMF가 모델을 정의하기 위해 지원하는 형식은 Ecore Model, XML Schema 등이 있다. 그림 3은 Ecore Model을 사용하여 모델을 정의한 예제이다. 예제에서 보여주고 있는 모델은 우편번호(Zip Code)를 정의하기 위한 것으로 위쪽은 XML 형태[1]로, 아래쪽은 Ecore 정의 언어를 사용하여 나타낸 것이다. 그림 4는 XML Schema를 사용하여 모델을 정의한 예제이다. 예제에서 보여주고 있는 모델은 Book이라는 이름을 가지고 속성으로는 title과 pages를 가지고 있는 구성요소의 한부분이다[3].

```
<xsd : simpleType name = "zipCodes">
      <xsd : restriction base = "xsd:integer">
            <xsd : minInclusive value = "1000"/>
            <xsd : maxInclusive value = "99999"/>
       </xsd : restriction>
</xsd : simpleType>
```

```
EDataType
      name = "zipCodes"
      instanceClass ="int"
      EAnnotation
         source = ".../ExtendedMetaData"
         details = "name -> zipCodes,
                    baseType -> .../XMLType#integer
                    minInclusive -> 1000,
                    maxInclusive -> 99999"
```

그림 3. Ecore Model

```
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
    name="library" nsURI="http:///library.ecore" nsPrefix="library">
  <eClassifiers xsi:type="ecore:EClass" name="Book">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="title"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="pages"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
</ecore:EPackage>
```

그림 4. XML Schema

그림 5는 부서관리 시스템 예제로서 EMF를 정의한 모델을 가지고 프로젝트를 생성했을 경우의 출력 결과이다. 각 구성요소들과 구성요소별 속성들이 트리구조로 나타난 것을 볼 수 있다. 트리구조에서 각 구성요소들을 편집할 수 있고, 구성요소별 속성들도 편집할 수 있다. 편집이 완료된 모델은 코드 생성과정을 통해서 소스로 나오게 된다.

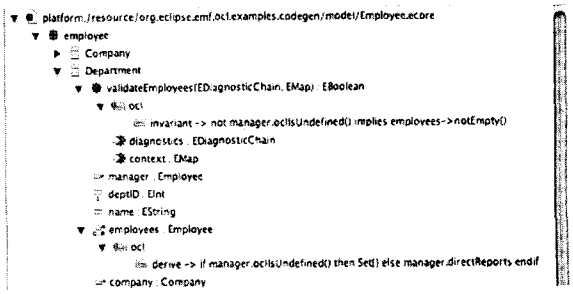

그림 5. EMF를 사용해서 생성한 트리구조

(2) GEF(Graphical Editing Framework)

GEF 플러그인은 이클립스에서 제공하는 프레임워크중의 하나로 다이어그램과 같은 Graphical Object들을 편집할 수 있다. GEF는 그래픽을 Object로 다루는 방법을 취하고 있다. Object를 다루는 방법을 취함으로써 추후에 편집하는 과정이

용이하다. 또한 구성요소 이동이나 화살표 모양의 선 처리 등이 GEF에서 제공된다.

그림 6은 UseCase Diagram[4]을 GEF를 사용해서 개발한 도구이다[9]. Actor와 UseCase는 GEF에서 제공하는 Ellipse, Point, drawLine 객체를 사용하였다. 선의 경우 GEF에서 제공하는 PloylineDecoration, Graphics Line dot 등의 객체를 사용하여 생성할 수 있으며 꺾인 선 처리는 Bendpoint 객체를 사용해서 쉽게 처리할 수 있다. 드로잉 심볼을 나타내는 툴바 또한 GEF에서 제공하는 GraphicalEditorWithFlyoutPalette를 사용하여 쉽게 생성할 수 있다[12, 13].

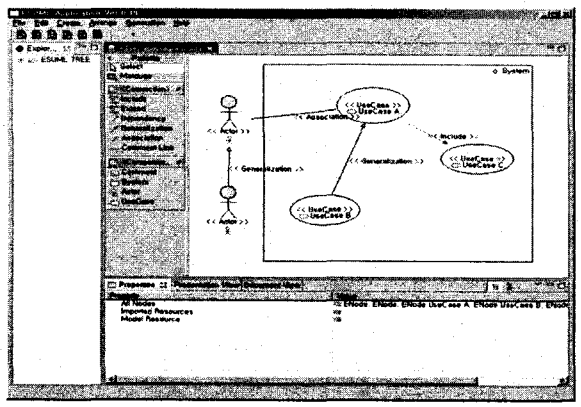

그림 6. GEF를 사용해서 개발한 도구

이상에서 설명한 것과 같은 이클립스 플러그인들은 상호작용을 통해 다양한 응용도구를 쉽게 개발할 수 있도록 지원한다. 그러나 본 연구의 서론에서 기술한 바와 같이 임베디드 소프트웨어를 모델링하기 위한 하드웨어 아키텍처용 플러그인이 제공되지 않고 있다. 따라서 다음 절에는 하드웨어 아키텍처를 표현하기 위한 플러그인 모델에 대하여 설명한다.

3. 하드웨어 아키텍처 다이어그램

3.1 하드웨어 아키텍처 메타모델

하드웨어에 대한 아키텍처를 작성하기 위해서는 하드웨어 아키텍처를 구성하는 요소들에 대한 심볼이 정의되어야 한다. 본 연구에서 정의한 하드웨어 아키텍처 다이어그램 심볼은 표 1과 같다. 대체적으로 심볼은 하드웨어 구성요소들의 유형(type)을 표현한 스테레오 타입을 갖는 사각형으로 정의되었다.

표 1. 하드웨어 아키텍처 다이어그램의 심볼

기호	설명	기호	설명
<< CPU >>	프로세서(processor)		연결자 (connector)
<< DSP >>	DSP	<< IO device >>	입출력 (I/O device)
<< Memory >>	메모리 (memory)	<< Peripheral >>	주변장치 (Peripheral)
<< Bridge >>	브리지 (bridge)	<< name >>	기타 (Other Components)
	버스 (bus)		

표 1에서 정의된 하드웨어 아키텍처 다이어그램의 각 구성요소들을 메타모델로 표현하면 그림 7과 같다.

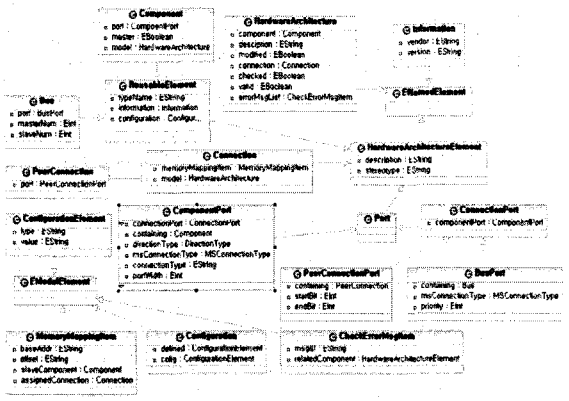

그림 7. 하드웨어 아키텍처 다이어그램 메타모델

그림 7에서 보는바와 같이 하드웨어 아키텍처 다이어그램을 표현하는 모델 요소들은 메타모델에서의 Component, Connection 등의 클래스에 의해 정의되며 다른 클래스들은 이들 간의 관계 및 인터페이스를 나타내는 특성(feature)들이다.

3.2 하드웨어 아키텍처 플러그인

3.1절에서 설명한 하드웨어 아키텍처 메타 모델을 사용하여 이클립스에서 제공하는 EMF 플러그인을 통해서 EMF 프로젝트를 생성한다. 그림 8은 EMF를 통해서 생성된 하드웨어 아키텍처 모델이다.

작성된 하드웨어 메타 모델을 EMF 플러그인을 사용해서 Generation하면 하드웨어 메타 모델에 대한 소스가 생성된다. 생성된 소스는 모델, 에디터, 임플, 프레젠테이션, 프로바이더, 유틸로 6개의 패키지로 구성된다. 그림 9는 생성된 패키지를 보여주고 있다.

그림8에서 모델 패키지는 EMF로 구성한 메타 모델에 대한 인터페이스를 포함하고 있고, 에디터 패키지는 이클립스 플랫폼에 연결하기 위한 플러그인을 포함하고 있다. 임플 패키지는 모델 인터페이스에 대한 구현 부분을 저장하고 있고, 프레젠테이션 패키지는 EMF로 구성한 메타 모델을 그림 5와 같은 트리 구조로 보여줄 수 있도록 하는 클래스를 저장하고 있다. 프로바이더 패키지는 각 모델에 대한 프로퍼티를 저장하고 있고, 유틸 패키지는 모델과 모델을 연결하는 팩토리와 리소스 관리를 하는 클래스들을 저장하고 있다.

4. PSM 모델링에서의 활용

그림 8에서 나타난 소스 코드와 GEF 플러그인을 사용하여 비주얼한 하드웨어 아키텍처 다이어그램 편집기(Drawer)를 개발할 수 있다. 이러한 편집기를 개발하는 이유 중의 하나는 한 칩에 다수의 프로세서를 갖는 MPSoc의 경우 하드웨어 아키텍처에 대한 다양한 모델링이 가능하기 때문이다[10].

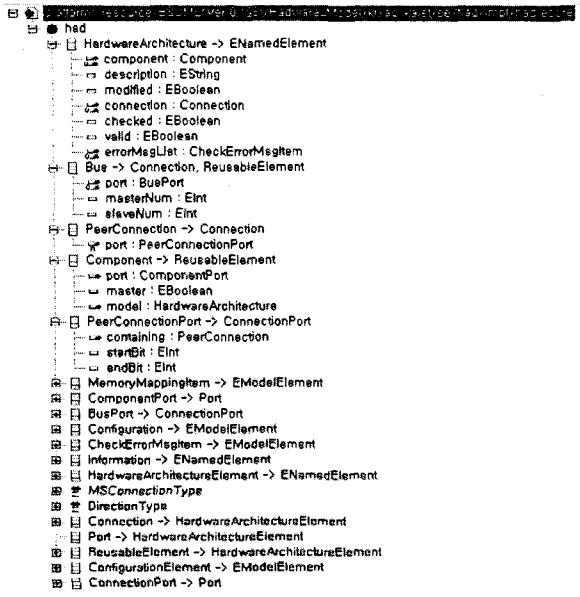

그림 8. EMF를 사용해서 개발한 하드웨어 아키텍처 모델

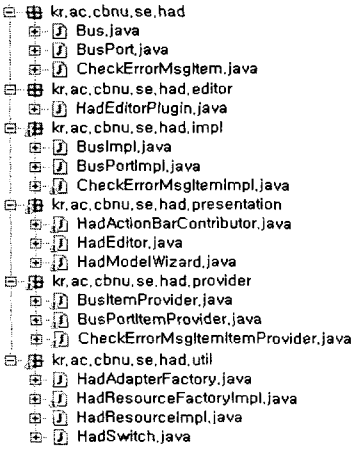

그림 9. EMF를 Generation한 후 생성된 소스

일반적으로 UML을 이용한 임베디드 소프트웨어를 개발하는 경우, 유즈케이스 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램, 스테이트 머신 등을 이용하여 대상 시스템의 PIM 모델을 개발하게 된다. 그런데 이러한 모델로 표현된 소프트웨어의 기능이 하드웨어 아키텍처의 어느 구성요소에 탑재되는 가에 따라 소프트웨어의 성능 및 기능 요구사항에 대한 충족 여부가 결정될 수 있다.

즉, MPSoC용 임베디드 소프트웨어의 개발에 있어서 PSM 모델은 PIM 모델의 분할을 통해 실행 가능한 태스크들로 분할하고, 이들을 하드웨어 구성요소에 할당한 모델이다. PIM 모델을 PSM 모델로 매핑하기 위해서는 PIM 모델로부터 도출된 태스크의 타입에 따라 적합한 하드웨어 요소에 할당하는 것인데, 예를 들면, 다수의 데이터를 사용하는 반복적인 연산처리 태스크는 DSP로 할당하는 것이다.

5. 결론 및 향후 연구 내용

본 논문에서는 하드웨어 아키텍처를 모델링하기 위한 이클립스 기반의 플러그인을 개발하였다. 개발된 플러그인은 하드웨어 아키텍처 다이어그램에 대한 메타정보를 가지고 있다. 그리고 이클립스 플랫폼에서 다른 플러그인과 연동하여 동작하는 것을 확인하였다.

이러한 하드웨어 아키텍처 플러그인은 MPSoC용 임베디드 소프트웨어 모델링에서 PSM 모델을 개발하기 위해 필요하게 되며 실제 물리적인 하드웨어를 개발하기 전에 소프웨어의 기능과 성능을 확인할 수 있는 기회를 제공해 준다.

향후의 주요 연구내용은 제시한 플러그인을 사용해 개발된 도구를 이용하여 MPSoC용 임베디드 소프트웨어의 PSM 모델링을 수행하고 이에 대한 기능 및 성능 만족을 점검하기 위해 분석 및 시뮬레이션을 수행하는 것이다.

6 참고문헌

[1] 김윤명, XML을 위한 Java Programming, 가남사, 2002

[2] 짐 단주, 스콧 페어브라더, 댄켄, 존켈러만 and 팻 매카시, 자바 개발을 위한 이클립스 바이블, 피어슨 코리아, 2005

[3] Eclipse.org Home, www.eclipe.org

[4] H.-E. Eriksson, M. Penker, B. Lyons and D. Fado, UML 2 Toolkit, Wiley Publishing, Inc., 2004

[5] Edward A. Lee, "Embedded Software", Advance in Computer, Vol.56, 2002

[6] J.M. Fernandes and J. Lilius, "Functional and Object-Oriented Views in Embedded Software Modeling", 11th IEEE ECBS'04, Czech, May 2004

[7] E. Gamma, L. Nackman, and J. Wiegand, Contributing to Eclipse Principles, Patterns, and Plug-Ins, Addison Wesley, 2006

[8] E. Gamma, L Nackman, and John Wiegand, Eclipse Modeling Framework, Addison Wesley, 2004

[9] E. Gamma, L. Nackman, and J. Wiegand, Eclipse Rich client Platform, Addison Wesley, 2006

[10] A.A. Jerraya, et al., "Programming models and HW-SW Interfaces Abstraction for Multi-Processor SoC", DAC 2006, San Francisco, July 2006, pp.280-285

[11] N. Medvidovic, et al., "Software Architecture and Embedded Systems", IEEE Software, Vol. 22(5), Sep. 2005, pp. 83-86

[12] B. Moore, et al., Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework, Addison Wesley, 2004

[13] M. Scarpino, et al., SWT/JFace In Action, 에이콘, 2006