Journal of Korea Multimedia Society Vol. 12, No. 2, February 2009(pp. 231-243)

임베디드 시스템의 재사용 프레임워크를 위한 정적 메타모델 설계

조은숙[†], 김철진[††], 이숙희[†††]

요 약

임베디드 시스템 시장이 확대되면서 최근 들어 자동차, 선박, 로봇 등과 같은 다양한 분야에서 임베디드 소프트웨어에 대한 관심이 증가되고 있다. 이와 관련하여 임베디드 운영체제, 임베디드 소프트웨어 모델링 기법, 임베디드 소프트웨어 테스팅 등 다양한 연구들이 이루어지고 있다. 그러나 지금까지 임베디드 분야가 하드웨어 분야에 치중되어 왔기 때문에 임베디드 시스템 개발에 있어서 체계적인 공학적 접근 방식이 미약한 상태이다. 이 뿐만 아니라 임베디드 시스템 개발에 있어서 재사용성을 고려한 프레임워크 기반의 설계 기법 등은 반영되지 못한 실정이다. 이렇게 개발됨으로써 시스템 내에 죽은 코드(Dead Code)들이 산재해 있을 뿐만 아니라 시스템의 재사용성이 매우 저조한 결과를 보이고 있다. 본 논문에서는 이러한 재사용성 향상을 위한 임베디드 시스템의 프레임워크를 제시하고, 이 프레임워크에 대한 정적 메타 모델을 제시한다. 이러한 메타 모델을 통해 임베디드 시스템의 재사용 프레임워크의 구조를 표현 할 뿐만 아니라, 이를 기반으로 다양한 임베디드 시스템 형태에 따라 모델을 쉽게 확장하여 설계할 수 있도록 한다.

A Design of Static Meta-Model for Reuse Framework of Embedded System

Eun-Sook Cho[†], Chul-Jin Kim[††], Sook-Hee Lee[†††]

ABSTRACT

Currently interests of embedded software in various areas such as automotive field, ship field, robot field is increasing according to expand market of embedded systems. Various researches such as embedded operating systems, embedded software modeling technique, embedded software testing, and so on are going in progress. However systematic engineering approach in embedded system development is poor because embedded areas focus on hardware parts until now. Furthermore, framework-based design technique considering reusability is not reflected in embedded system development. Those development results in many of dead codes scattered in system, and results in poor reusability of system. This paper suggests a framework of embedded system for reusability and a static meta-model for reuse framework. Proposed meta-model expresses not only the structure of reuse framework, but also allows a designer to extend and design easily models of embedded system based on reuse framework according to various embedded system types.

Key words: Reuse Framework(재사용 프레임워크), Embedded System(임베디드 시스템), Meta-Model(메타 모델), Variability(가변성)

※ 교신저자(Corresponding Author) : 김철진, 주소 : 경기도 수원시 영통구 매탄3동(443-742), 전화 : 031)277-6099, FAX : 031)277-6400, E-mail : cjkim777@gmail.com
접수일 : 2008년 9월 17일, 완료일 : 2008년 12월 30일
[†] 정회원, 서일대학 소프트웨어과 조교수
(E-mail : escho@seoil.ac.kr)
[††] 정회원, 삼성전자 책임연구원
[†††] 정회원, 서경대학교 인터넷정보학과 교수
(E-mail : olecsh@skuniv.ac.kr)
※ 본 연구는 2008년 서일대학 학술연구비에 의해 연구되었음

1. 서 론

최근 들어 전자, 자동차, 선박, 로봇 등과 같은 다양한 분야에서 임베디드 시스템 내의 임베디드 소프트웨어 분야에 대한 연구가 급증하고 있는 추세이다. 국내의 경우만 보더라도 삼성전자, 현대 자동차 등과 같은 국내 대기업들이 임베디드 소프트웨어 관련 프로젝트나 솔루션 개발 등에 관심을 갖고 있으며, 로봇 분야에 있어서도 UML(Unified Modelling Language)과 같은 모델링 언어를 이용하여 로봇 제어 소프트웨어를 설계하는 추세를 보이고 있는 실정이다[1]. 이뿐만 아니라 RCW Mirus사 2001년 보고에 따르면 2002년 임베디드 시스템 세계시장은 약 1,000억 달러 규모이며, 그 중에서 임베디드 소프트웨어 분야는 약 200억 달러 규모로 매년 평균 20%의 성장률을 보여, 2007년 약 500억 달러 수준으로 전망하였다[2]. 이러한 상황 가운데 임베디드 기술과 SoC 분야는 소프트웨어 기술을 어떻게 함께 접목하여 연동할 것인가에 대한 부분이 중요한 이슈로 부각되고 있다. 예를 들면, 임베디드 소프트웨어 개발 방법, 임베디드 소프트웨어 개발 프로세스, 임베디드 운영체제, 임베디드 미들웨어, 임베디드 소프트웨어 테스팅, 그리고 SoC 모델링 기법 등 다양한 기술들이 연구되고 있는 실정이다. 그러나 아직도 임베디드 시스템 개발에 있어서 소프트웨어 공학적 측면에서의 체계적인 기법들이 거의 제대로 적용되고 있지 않고 있는 상태이다.

특히 임베디드 시스템은 기존 시스템과 달리 하드웨어와 소프트웨어 요소들이 서로 결합된 시스템이기 때문에 임베디드 시스템 설계 시 실시간성(Real-time), 반응성(Reactive), 소규모(Small Size), 경량화(Low Weight), 안전성(Safe), 신뢰성(Reliable), 견고성(Harsh Environment), 저비용(Low Cost) 등의 요소들을 고려하여 설계해야 한다[3,4]. 그러나 현재 임베디드 시스템 개발에 이러한 요소들을 제대로 고려하지 않은 채 설계되고 있다.

또한 현재 개발되는 대부분의 소프트웨어 시스템의 형태를 보면 프레임워크를 기반으로 하고 있다. 스트럿츠(Struts), 스프링스(Springs), 이클립스(Eclipse) 등과 같은 프레임워크들이 대표적으로 많이 사용되고 있다[5-7]. 그러나, 임베디드나 유비쿼터스 소프트웨어 개발에는 프레임워크들이 사용되고 있지 않을 뿐만 아니라, 재사용을 지원할 만한 프레임워크 개발이 매우 미흡한 상태이다. 특히 동적 재구성을 핵심으로 하는 프레임워크 개발은 임베디드 시스템 개발에 있어서 소프트웨어 개발의 생산성 및 재사용성의 향상을 극대화 시키기 때문에 매우 중요하다고 볼 수 있다[8].

그러나 현재 대부분의 임베디드 시스템 개발에 있어서 프레임워크를 기반으로 임베디드 시스템을 개발하는 형태가 거의 이뤄지고 있지 않고 있다. 이로 인해 현재 개발되어 있는 임베디드 시스템들의 내부 코드들을 살펴보면 시스템 작동과 전혀 상관이 없는 코드들이 무수하게 잔재하고 있으며, 임베디드 시스템 개발에 있어서 재사용성이나 가변성을 반영하여 설계하는 사례 또한 거의 존재하지 않고 있다. 이러한 흐름이 계속 이어지게 될 경우, 향후 임베디드 시스템은 유지보수의 어려움, 개발비의 증가, 유비보수비의 증가, 시스템 품질 저하 등과 같은 다양한 위기에 직면하게 될 것이다.

따라서 본 논문에서는 이러한 문제점들 가운데 재사용성이나 가변성을 향상시키기 위한 재사용 프레임워크 구조를 정의하고 이 구조에 대한 메타 모델을 제시하고자 한다.

이러한 재사용 프레임워크를 기반으로 임베디드 시스템을 개발할 경우에 있어서는 다양한 디바이스들에 대한 가변성을 동적으로 지원할 수 있게 된다. 이는 결과적으로 임베디드 시스템의 재사용성을 향상시키는 결과를 가져올 뿐만 아니라 시스템의 복잡도 또한 줄일 수 있는 효과를 기대할 수 있게 된다.

본 논문은 다음과 같이 구성된다. 2장에서는 임베디드 시스템과 재사용 프레임워크 및 메타 모델에 대한 관련 연구를 다루며, 3장에서는 재사용 프레임워크에 대한 정적 메타 모델을 제시한다. 4장에서는 재사용 프레임워크 메타 모델을 기반으로 한 홈 네트워크 시스템 설계 모델을 사례 연구로 제시한다. 5장에서는 기존 객체지향 설계와 본 논문에서 제시하는 재사용 프레임워크를 이용하여 설계한 사례를 비교 평가한다.

2. 관련 연구

2.1 임베디드 시스템

임베디드 시스템이라 함은 특정한 기기에 주어진

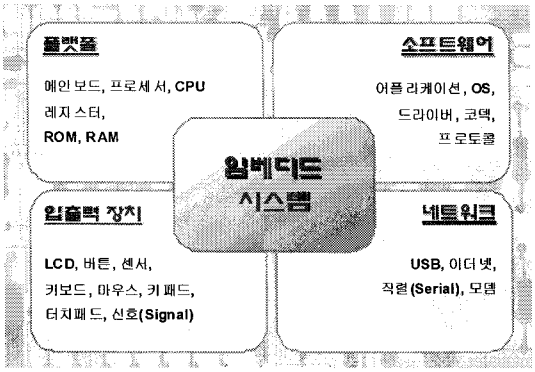

그림 1. 임베디드 시스템 구성 요소

작업을 수행하도록 구동시키는 시스템이라 할 수 있다. 첨단 기능이 들어있는 가전제품이나 컴퓨터, 엘리베이터, 공장 자동화 시스템 등등 특정한 기기를 운용할 수 있는 운용체제라면 임베디드 시스템이라 할 수 있다[4,8,9]. 이러한 임베디드 시스템 가운데 하드웨어를 제외한 나머지 부분을 임베디드 소프트웨어라고 말할 수 있다. 그림 1은 임베디드 시스템을 구성하는 구성 요소들을 분야 별로 표현한 것이다. 이러한 요소들이 서로 상호 유기적으로 잘 연관되어 상호작용하기 위해서는 이들 요소간의 관계를 제대로 표현할 수 있는 임베디드 시스템에 대한 메타모델이 필요하다. 본 논문에서는 이러한 임베디드 시스템 내에 들어가는 이러한 구성요소들과 이들 간의 상호 흐름을 내포하는 재사용 프레임워크에 대한 메타모델을 설계하여 제시하고자 한다.

2.2 재사용 프레임워크

흔히 소프트웨어에서의 프레임워크를 하드웨어의 메인 보드에 비유해서 표현한다. 프레임워크란 여러 유사한 애플리케이션들이 공통으로 필요로 하는 모듈로 구성된 뼈대 구조이다. 하드웨어 메인 보드에 여러 재사용 부품인 메모리, CPU, 네트워크 카드, 그래픽 카드, 사운드 카드 등등의 컴포넌트들이 조합되어 있고, 이들 간에 서로 신호나 데이터를 주고 받을 수 있는 회로들이 설계 및 구현되어 있다. 이처럼 프레임워크 내에도 공통 모듈들이 재사용 가능한 컴포넌트 형태들로 구성되어 있으며, 컴포넌트들 간의 상호 작용을 위한 흐름이 설계되어 구현되어 있다[10-12].

최근에는 대부분의 소프트웨어 개발 프로젝트에서 소프트웨어 개발의 생산성 향상과 재사용성 및 유지보수성을 향상시키기 위해 프레임워크 기반의 소프트웨어 개발 방식을 취하고 있다[13-15]. 그러나 임베디드 시스템과 같은 경우는 아직도 이러한 프레임워크 형태의 개발 방식이 도입되고 있지 않고 있다. 이로 인해 임베디드 시스템 내의 전체 코드 가운데 죽은 코드가 차지하는 비중이 점점 증가되고 있다. 이는 결국 임베디드 시스템의 유지보수에 있어서 막대한 비용을 초래할 뿐만 아니라 시스템의 신뢰성을 보장하기가 어렵다. 특히 무인 자동차 시스템이나 국방 관련 무기 시스템에 이러한 형태의 개발이 이루어지면 이로 인해 발생되는 위험과 손실은 막대한 비용을 초래할 것이다. 따라서 본 논문에서는 임베디드 시스템의 품질과 유지보수성 그리고 재사용성을 향상시키기 위해 프레임워크를 설계하고 프레임워크에 대한 정적 메타모델을 설계한다.

2.3 메타 모델

메타모델이란 어떠한 개념들을 구성하는 주요 요소들과 그들 간의 관계를 보여주는 개념 도이다. UML에서는 MOF(Meta Object Facility)를 통해 메타모델을 정의하고 있는데, 객체지향 모델 작성에 사용되는 UML 모델의 필수 요소와 문법, 구조를 정의하는 메타 모델로 제시하고 있다. 또한 MDA(Model Driven Architecture)에서는 MOF를 CWM(Common Warehouse Metamodel)이나 UML의 메타모델에 대한 공통 모델로 제공하고 있다[16,17].

메타모델은 특정 개발 플랫폼에 종속되지 않은 형태로 제공되기 때문에 메타 모델에 기반을 두고 모델을 개발할 경우 플랫폼에 따른 다양한 형태의 모델들을 확장하여 개발할 수 있을 뿐만 아니라 모델의 재사용성이 매우 향상되게 된다.

3. 재사용프레임워크를위한 정적메타모델설계

이 장에서는 본 논문에서 제안하는 임베디드 시스템의 효율적 개발을 위한 재사용 프레임워크의 구조를 정적 메타 모델을 통해 설계하고자 한다.

3.1 재사용 가능한 임베디드 시스템 계층 구조

임베디드 시스템을 구성하는 구성요소들을 살펴

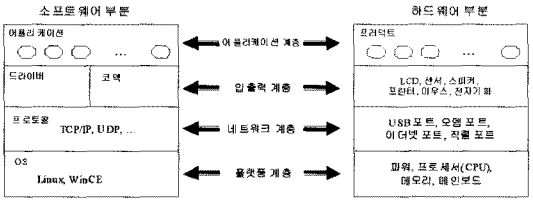

그림 2. 임베디드 시스템 계층 구조

보면 구성 요소들 간에는 독립적인 계층구조 형태로 구별하여 표현할 수 있다. 그림 2에 제시된 것처럼 임베디드 시스템을 재사용 가능한 형태의 계층으로 구별하면 크게 4계층으로 정의할 수 있다. 또한 임베디드 시스템은 하드웨어와 소프트웨어 요소가 함께 결합된 형태이기 때문에 크게 하드웨어 부분과 소프트웨어 부분으로 구별할 수 있고, 각 부분 별로 4계층으로 표현할 수 있으며, 각각의 부분에 속하는 계층의 요소들이 상대방 부분에 속하는 계층 요소와 관계를 갖는다. 예를 들어, 하드웨어 부분의 네트워크 계층에 해당 하는 USB 포트나 모뎀, 네트워크 포트 등에 대응되는 소프트웨어 부분의 네트워크 계층 요소로는 TCP/IP, UDP 등과 같은 프로토콜들이 존재한다. 본 논문에서는 이러한 요소들을 재사용 가능한 컴포넌트들로 추출하여 프레임워크내의 구성 요소들로 정의하고 이들 간의 관계를 표현한다.

3.2 임베디드 시스템 메타 모델

임베디드 시스템의 가변부에 대한 재사용 프레임워크는 어댑터와 핵심 클래스를 통해 동적으로 서비스를 제공할 수 있다. 이러한 재사용 프레임워크의 어댑터와 핵심 클래스들의 내부 구조에 대한 메타 모델은 그림 3과 같다. 임베디드 시스템의 가변성에 대한 어댑터들은 재사용 요구 인터페이스(Required Interface)를 통해 어떤 기능을 사용할지 설정하며, 재사용 프레임워크의 가변성 관리기는 설정 및 서비스 호출에 대한 대행 역할을 수행한다. 가변성 관리기에 의한 설정은 리플렉터, 설정 관리기, 그리고 동적 전개기를 통해 이루어지며, 설정된 서비스는 가변성 어댑터들에 의해 제공된다. 임베디드 시스템의 자원에 대한 관리는 자원 관리기에 의해 설정되며 가변성 어댑터에 따라 다르게 설정된다. 예를 들면, 운영체제 가변 어댑터가 설정하는 운영체제(WindowCE, VxWorks, 등)에 따라 자원 관리기는 서로 다른 자원(메모리, 등)을 할당한다.

3.2.1 가변부(Variation Point)

임베디드 시스템의 큰 부류로는 가변부들을 중계하기 위한 어댑터 클래스들과 가변부 처리를 지원하기 위한 핵심 클래스들로 구성된다. 재사용 프레임워크의 어댑터는 홈 네트워크 시스템의 가변성인 디바이스 프로토콜, 디바이스 드라이버, AV 코덱, OS, 제어 가변성에 대한 대행 역할을 수행한다. 이에 대한 메타 모델이 그림 4에 제시되어 있다.

3.2.2 인터페이스

가변부의 변경을 제공하기 위해 재사용 프레임워크는 그림 5에 제시된 메타모델에서 인터페이스를 제공한다. 제공 인터페이스(Provided Interface)는 재사용 프레임워크의 요구 인터페이스를 통해 설정된 서비스를 제공한다. 홈 네트워크 시스템의 가변성에 대한 요구 인터페이스는 다양한 가변 기능으로 설정될 수 있으며, 이러한 설정은 재사용 요구 인터페이스를 통해 설정된다.

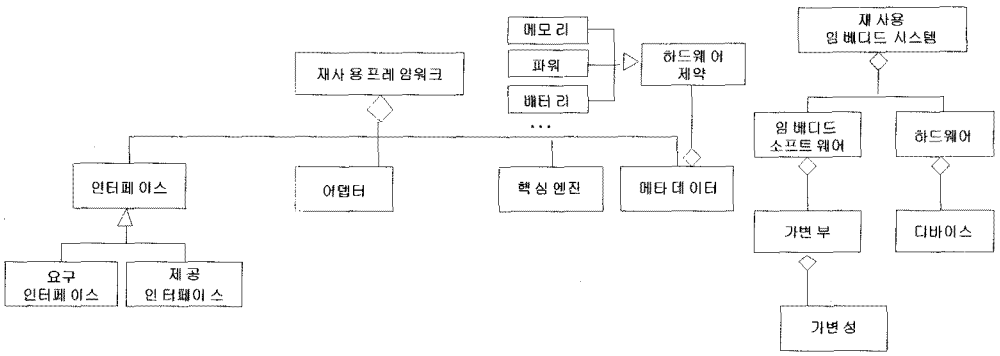

그림 3. 정적 메타 모델

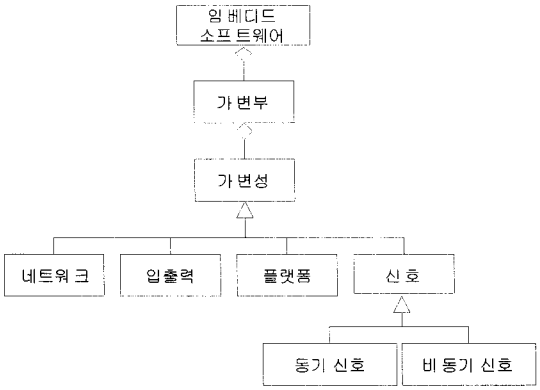

그림 4. 가변부 메타 모델

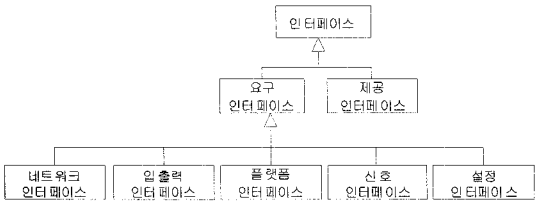

그림 5. 인터페이스 메타 모델

3.2.3 어댑터(Adapter)

가변성 어댑터는 가변부를 대행하는 중계자 역할을 하며, 선택된 가변성 어댑터는 메타정보에 정의된 클래스를 호출한다. 재사용 프레임워크의 가변성 어댑터는 다양한 가변 처리를 지원할 수 있도록 요구 인터페이스로 정의되며, 이러한 요구 인터페이스에

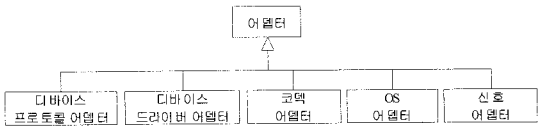

그림 6. 어댑터 메타 모델

맞는 클래스를 통해 가변부를 제공한다. 그림 6은 임베디드 시스템의 가변성인 디바이스 프로토콜, AV 코덱, 운영체제 가변성에 대한 가변부 설계 구조를 나타낸다.

임베디드 시스템은 다양한 디바이스 프로토콜을 통해 다양한 디바이스를 지원하기 때문에 디바이스에 맞는 프로토콜인 RS485 나 PSTN(Public Switched Telephone Network), PLC(Power Line Communication) 등을 가변적으로 선택할 수 있도록 제공해야 한다. AV 코덱과 운영체제 가변성도 이와 동일하게 요구 인터페이스를 구현한 가변부를 설계한다.

어댑터는 메타정보가 변경되면 물리적인 클래스도 변경되어 다른 기능을 제공하도록 한다. 기존 연구와 다르게 본 논문에서는 어댑터를 통해 기능을 제공하는 클래스를 동적으로 변경할 수 있으며, 어댑터를 변경하여 가변부를 사용하는 클래스에서 전혀 다른 기능을 호출할 수 있다.

그림 7과 같이 도메인 요구사항(Domain A, B)에 따라 단일 어댑터('DeviceProtocolAdapter')을 통해 가변부 클래스를 선택하거나 다중의 가변부 클래스

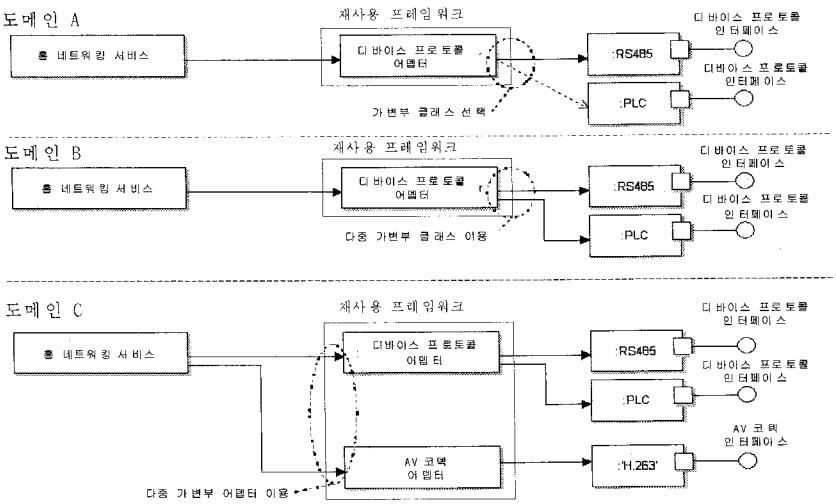

그림 7. 도메인 요구사항에 따른 가변성 어댑터 구조

를 동시에 선택하여 서비스를 이용할 수 있다. 임베디드 시스템은 다양한 디바이스에 신호를 전달해야 하는 경우에 가변적인 서비스 구조가 요구된다. 도메인 C와 같이 서로 다른 형태의 가변성 서비스를 요구하는 경우 다중의 가변성 어댑터를 동시에 이용할 수 있어야 한다. 임베디드 시스템에서 다중의 디바이스를 디바이스 프로토콜뿐만 아니라 디바이스에 따른 멀티미디어 서비스를 제공하기 위해 AV 코덱을 가변적으로 접근하여 이용할 수 있다. 이러한 다중 어댑터를 통한 다중 기능의 제공이 기존 연구와 차별화된 특징이라고 할 수 있다.

3.2.4 핵심 엔진(Core Engine)

핵심 엔진은 재사용 프레임워크를 구성하는 주요 핵심 요소들로서, 이들은 가변성을 담당하는 가변성 관리기(Variability Manager), 동적 전개기(Hot Deployer), 설정 관리기(Configuration Manager), 리플렉터(Reflector), 자원 할당기(Resource Allocator) 등을 말한다. 이에 대한 메타 모델이 그림 8에 제시되어 있다.

- 가변성 관리기

가변성 관리기는 가변부를 사용할 수 있도록 중계하는 역할을 하며 재사용 프레임워크의 인터페이스 역할을 한다. 가변부를 사용하는 영역에 의해 특정 가변부를 호출하게 되면 가변성 관리기는 재사용 프레임워크 내의 리플렉터, 설정 관리기, 동적 전개기, 자원 할당기, 그리고 가변성 어댑터들과 상호 작용을 통해 가변부의 특정 기능을 동적으로 호출한다.

가변부 사용 영역에서는 가변부를 사용하기 위해

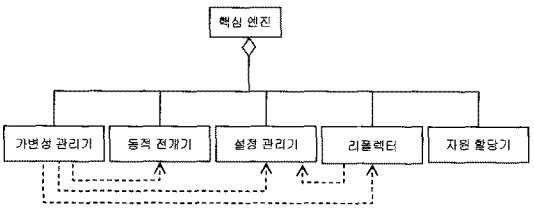

그림 8. 핵심 엔진 메타 모델

```
Object result = VariabilityManager.execute( _VARIABILITY_NAME_ , _PARAMETER_ );
```

_VARIABILITY_NAME_ : 가변부 식별자

그림 9. 가변부 사용 영역에서의 가변부 관리기 실행 코드

그림 9과 같이 가변성 관리기를 통해 가변부를 호출한다. 가변부 호출 시 가변부에 대한 식별자와 입력 데이터를 전달하여 가변부 가변성 관리기가 가변부를 식별하여 요구하는 기능을 호출한다. 가변부 사용 영역에서는 식별자에 의해서 호출하기 때문에 재사용 프레임워크 내부에서 어떤 가변성 어댑터나 클래스로 변경되더라도 전혀 영향을 받지 않는다.

- 동적 전개기

동적 전개기는 홈 네트워크 시스템 내부에서 가변성을 제공할 수 없는 경우 시스템 외부에서 요구하는 기능 클래스를 시스템 내부로 플러그-인 시키기 위한 도구이다. 플러그의 개념은 시스템 패키지 내에 요구하는 클래스를 포함하는 것이 아니라 시스템 운영 환경에 맞게 객체가 생성되는 것을 의미 한다.

그림 10과 같이 재사용 인터페이스 통해 디바이스 프로토콜 가변성을 시스템 외부에서 제공해야 하는 경우에는 재사용 프레임워크의 동적 전개기에서 요구하는 가변부 클래스('RF')의 객체를 초기화하여 현 운영되는 시스템에서 가변적으로 접근할 수 있도록 한다.

- 설정 관리기

설정 관리기는 가변부 사용 영역에 의해 사용될 가변부의 메타정보를 관리한다. 설정 관리기는 가변부의 가변부 식별자를 통해 가변부 메타정보를 호출하며, 이러한 가변부 식별자는 가변부 사용 영역에서 정의하여 가변성 관리기를 통해 설정 관리기에 전달한다. 가변부 사용 영역에서 전달된 가변부 식별자를 기반으로 설정 관리기는 가변부의 상세한 가변부 메타정보를 얻는다.

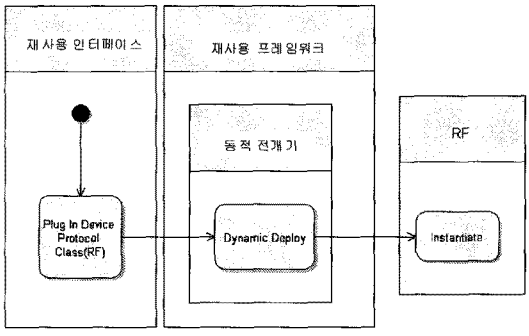

그림 10. 동적 전개기에 의한 가변부 클래스 플러그 인

설정 관리기는 XML 기반의 메타정보를 관리하기 때문에 동적인 메타정보 관리가 가능하며, 이러한 설정 관리기는 홈 네트워크 시스템의 특성상 도메인에 따라 다양 디바이스로 변경해야 하는 요구사항을 만족시킬 수 있는 도구이다.

• 리플렉터

리플렉터는 가변부의 메타정보를 통해 물리적인 가변성 어댑터나 가변성 클래스를 호출하기 위한 도구로서 동적으로 클래스를 호출할 수 있도록 하기 위해 리플렉션(Reflection) 기능을 기반으로 한다. 리플렉션은 메타 형태의 클래스 명(String 타입)과 행위 명(String 타입), 그리고 입력 파라메터(Object Array 타입)를 제공하여 물리적인 클래스의 기능을 호출할 수 있는 메커니즘이다. 이러한 리플렉션 메커니즘은 표준 개발 플랫폼(J2EE, .NET) 에서 제공되고 있다. 본 재사용 프레임워크의 리플렉터는 이러한 메커니즘을 변경하여 가변부의 메타정보를 처리할 수 있는 기능을 제공한다. 이와 같이 본 재사용 프레임워크의 리플렉터는 가변부 클래스의 다양한 행위뿐만 아니라 다양한 인터페이스의 클래스를 동적으로 변경할 수 있도록 한다.

그림 11에서와 같이 리플렉터는 가변성 어댑터 식별자를 이용하여 물리적인 가변성 어댑터를 실행한다. 또한 가변성 어댑터는 리플렉터를 통해 가변성 클래스를 실행한다. 가변부를 변경하고자 할 때는 가

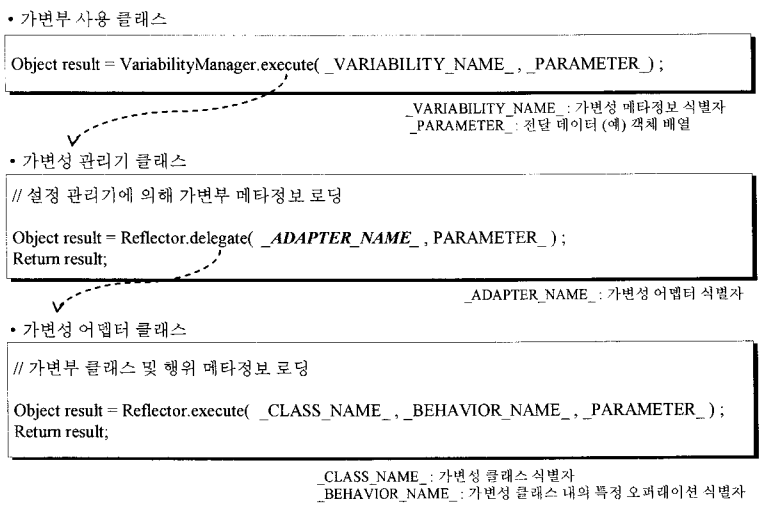

그림 11. 메타정보 기반의 가변부 변경 및 실행 코드

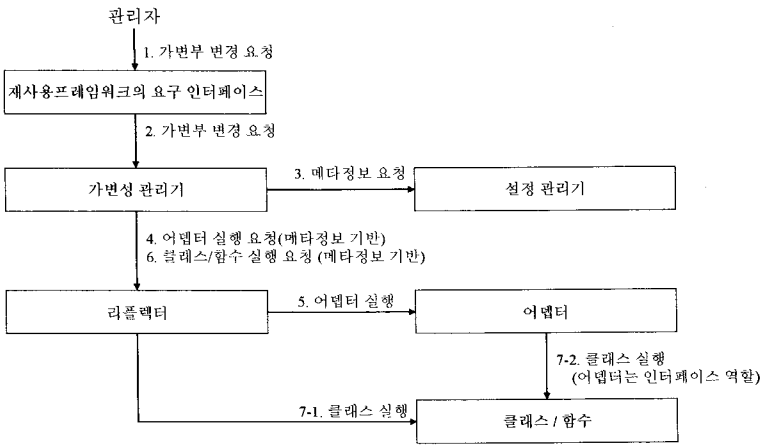

그림 12. 메타정보 기반의 가변부 변경에 대한 흐름도

변부 메타정보 만 변경하면 되며 그림 11의 가변부 사용 구조는 전혀 영향을 받지 않는다. 단지 전달되는 파라메터의 타입은 고려할 여지가 있다.

그림 12는 그림 11의 가변부 변경 코드에 대해 재사용 프레임워크 내에서 구성요소들 간에 어떤 흐름 관계를 갖는지를 나타내는 흐름도 이다.

대부분의 홈 네트워크 시스템의 가변성은 이러한 구조로 가변부를 제공할 수 있으며 디바이스 프로토콜이나 디바이스 드라이버 클래스에 대한 변경도 이러한 구조를 통해 가능하다.

• 자원 할당기

자원 할당기는 임베디드 시스템의 운영 환경에 변경에 따른 자원 할당을 지원하기 위한 도구로서 홈 네트워크 시스템의 경우 주로 운영체제 변경에 따른 메모리나 지원 전압의 변경을 설정한다.

3.2.5 메타 데이터(Meta Data)

메타 데이터는 가변부의 메타정보를 포함하고 있는 저장소로서 XML 기반의 가변부 정보를 관리한다.

그림 13에서와 같이 가변부의 메타정보는 가변부 식별자('<Variability Name="_VARIABILITY_NAME_" >'), 가변성 어댑터 명('<Adapter Name="_ADAPTER_NAME_">'), 가변부의 기능을 제공하는 클래스 명('<Class>'), 클래스의 행위 명('<Behavior>'), 실행 환경의 컨텍스트 정보('<Context>')로 구성된다. 가변부 식별자는 가변부 사용 영역에서 가변부를 호출하기 위해 사용되는 가변부 식별자로서 가변부 사용 영역의 코드에 정의한다. 가변성 어댑터 명은 재사용 프레임워크의 구성요소로서 가변부의 클래스들 중에 동적으로 특정 클래스로 변경하기 위한 중계 역할을 한다. 클래스 명은 어댑터를 통해 선택할 수 있는 클래스를 나타내며 행위 명은 선택된 클래스의 행위(오퍼레이션, 함수)

```
...
<Variability Name = "_VARIABILITY_NAME_">
    <Used-By> _ADAPTER_NAME_ </Used-By>
</Variability>

<Adapter Name ="_ADAPTER_NAME_">
    <Class> _CLASS_NAME_ </Class>
    <Behavior> _BEHAVIOR_NAME_ </Behavior>
    <Context> _CONTEXT_INFORMATION_ </Context>
</Adapter>
...
```

그림 13. 가변부 메타정보

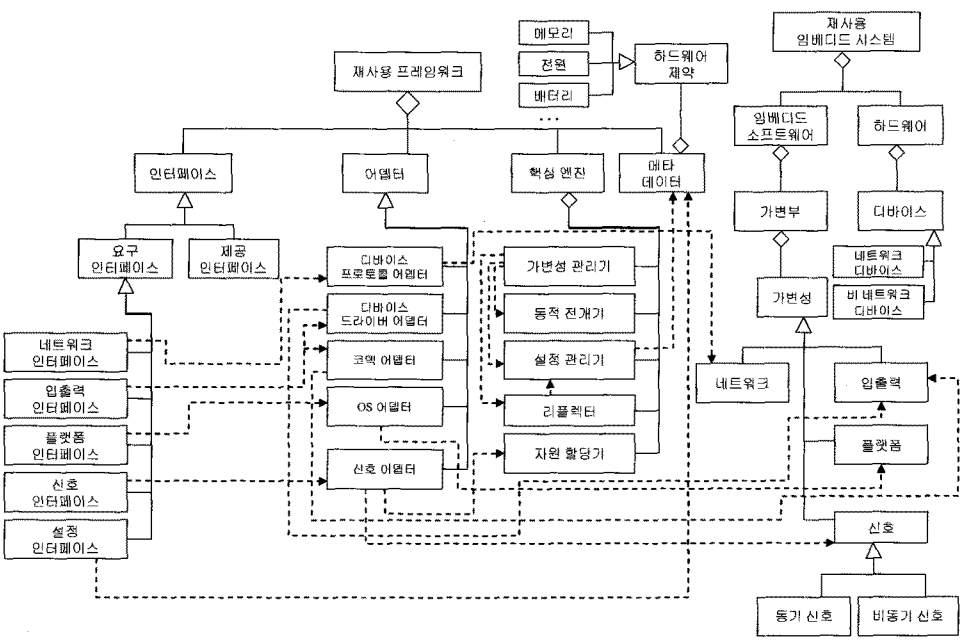

그림 14. 정적 메타 모델

를 나타낸다. 본 논문에서는 제안하는 메타정보는 다중의 어댑터를 통해 다중의 클래스와 다중의 행위를 호출할 수 있는 메타정보를 정의할 수 있다. 이러한 메타정보의 특징은 본 논문이 다양한 기능을 동적으로 변경할 수 기반을 제공한다.

3.2.6 정적 메타 모델

지금까지 설명한 각 요소 별 메타 모델들을 통합하여 그림 14와 같이 임베디드 시스템의 재사용 프레임워크에 대한 정적 메타 모델을 제시한다.

4. 사례 연구

본 사례연구에서는 재사용 프레임워크 메타모델을 기반으로 홈 네트워크 시스템의 출입통제 기능을 설계한다. 이렇게 설계된 정적 설계를 통해 메타모델의 적합성을 증명한다.

4.1 정적 모델 사례

본 논문에서 제안하는 재사용 프레임워크를 이용한 설계 사례는 그림 15과 같으며 'Visiting' 클래스와 가전 디바이스 제어 프로토콜인 'RS485' 클래스가 'Reusability Framework'를 통해 연결된다. 'Visiting' 클래스는 디바이스 제어 프로토콜 'RS485' 클래스와 직접 연결되지 않는다.

그림 16와 같이 'Reusability Framework'을 통해 다양한 디바이스 프로토콜을 지원할 수 있다. 'RS485', 'PLC', 'PSTN' 프로토콜 클래스는 'DeviceProtocolIF' 인터페이스를 따르며, 'Reusability Framework'에서 다양한 프로토콜(PLC, PSTN, RF, 등)을 가변적으로 처리할 수 있다.

그림 17는 'Reusability Framework'의 내부 구조를 설계한 복합 구조 다이어그램(Composite Structure Diagram)이다. 다양한 어댑터를 제공하여 다양한 기능을 가변적으로 처리할 수 있도록 한다. 'CE_Device_Protocol_Adapter' 클래스는 다양한 가전 디바이스 프로토콜을 제공하기 위한 어댑터 이며, 'AV_Codec_Aapter' 클래스는 다양한 코덱을 제공하기 위한 어댑터 이다. 어댑터는 메타정보를 통해 단일 또는 다중으로 선택할 수 있다.

그림 18은 디바이스 프로토콜의 가변부를 처리하기 위한 메타정보를 XML 문서 형태로 표현하고 있다.

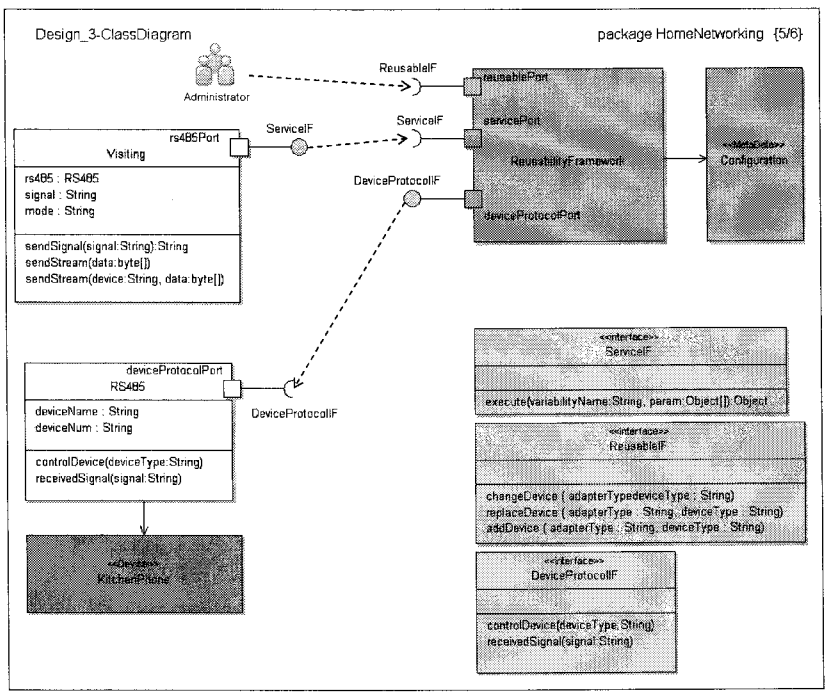

그림 15. Reusability Framework을 이용한 출입통보 설계 사례

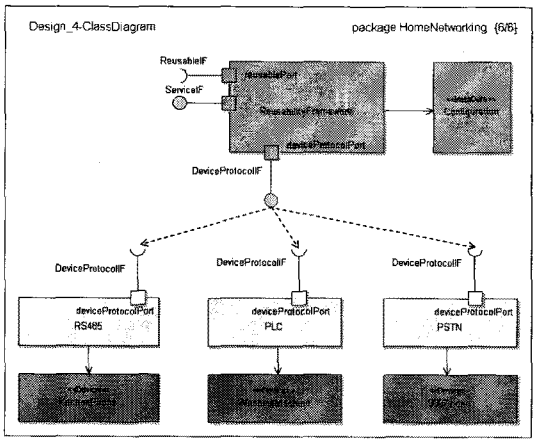

그림 16. 다양한 디바이스 프로토콜 지원하기 위한 설계

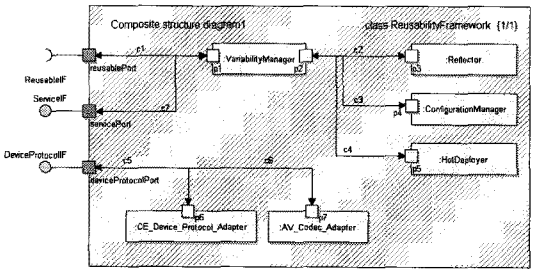

그림 17. Reusability Framework의 내부 구조(Composite Structure Diagram)

5. 평 가

본 논문에서 제시하는 재사용 프레임워크에 대한 평가를 위해 Hironori Washizaki가 제시하는 평가 메트릭[18]을 기반으로 기존의 객체지향 설계 방식과 본 논문에서 제시하는 재사용 프레임워크를 이용한 설계 사례를 평가한다. 재사용 평가 메트릭은 그림 19과 같이 이해성(Understandability), 적응성(Adaptability), 변경성(Changeability), 교체성(Replaceability), 확장성(Extensibility), 이식성(Portability)을 평가한다.

재사용 메트릭을 이용하여 기존 객체지향 설계와 재사용 프레임워크 이용한 설계의 측정 결과는 그림 20과 같다.

기존 객체지향 설계 방식과는 다르게 재사용 프레임워크를 이용한 설계 사례는 COR이 0 보다 크므로 재사용성이 있다고 할 수 있다[18]. 구조적 측면에서는 재사용 프레임워크가 추가 되기 때문에 복잡성이 높아 질 수 있으나 기존 객체지향 설계에 비해 변경성, 교체성, 확장성이 좋아지므로 재사용성이 향상된다고 할 수 있다.

6. 결 론

임베디드 시스템은 하드웨어와 소프트웨어 결합

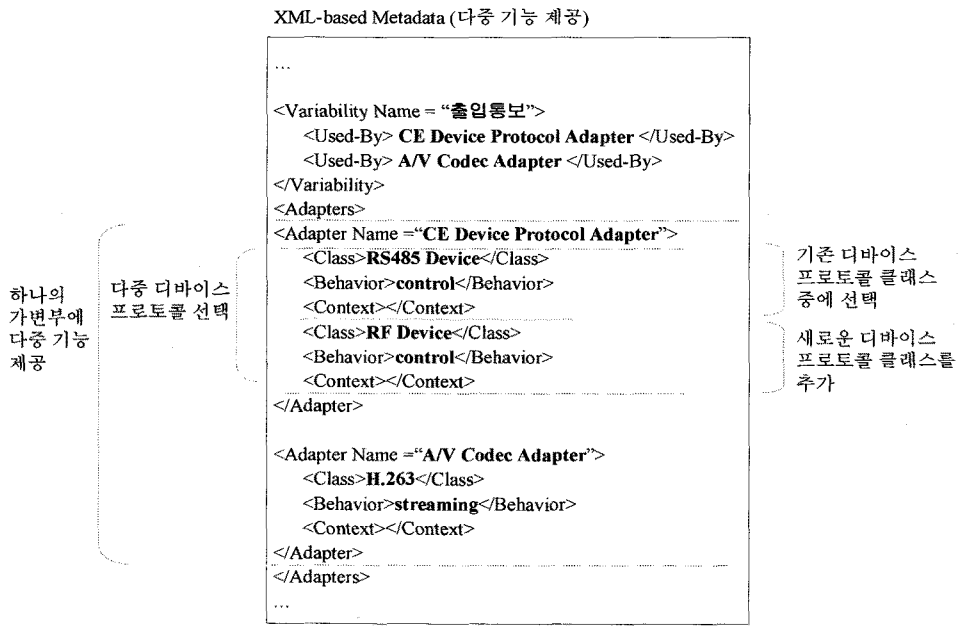

그림 18. 디바이스 프로토콜 가변부 처리 메타정보

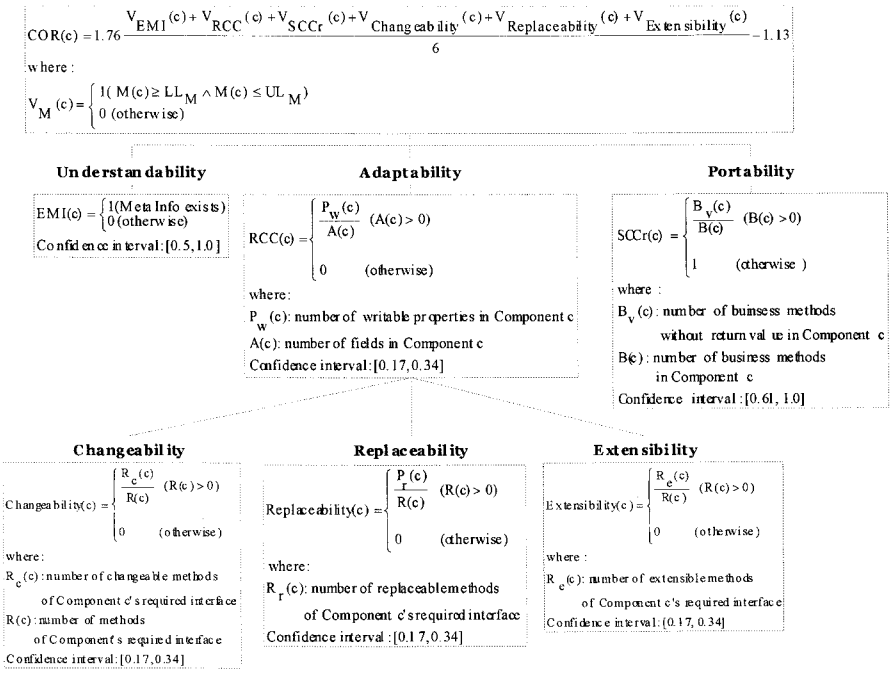

그림 19. 재사용 평가 메트릭

Factor	Measured Value	
	객체지향 설계	재사용 프레임워크 이용 설계
Meta Info	No	Exists
Number of Field	5	5
Number of Writable Properties	3	3
Number of Business Method	5	5
Number of Business Method without return value	4	4
Number of Method of Required Interface	0	3
Number of Changeable Method of Required Interface	0	1
Number of Replaceable Method of Required Interface	0	1
Number of Extensible Method of Required Interface	0	1

Metric	Measured Value		V_{Metric}	Measured Value	
	객체지향 설계	재사용 프레임워크 이용 설계		객체지향 설계	재사용 프레임워크 이용 설계
EMI	0	1	V_{EMI}	0	1
RCC	3/5=0.6	3/5=0.6	V_{RCC}	0	0
SCC_r	4/5=0.8	4/5=0.8	V_{SCCr}	1	1
Changeability	0/0=0	1/3=0.3	$V_{changeability}$	0	1
Replaceability	0/0=0	1/3=0.3	$V_{replaceability}$	0	1
Extensibility	0/0=0	1/3=0.3	$V_{Extensibility}$	0	1

기존 객체 지향 설계 시 재사 용성 $COR = 1.76 \times \frac{0+0+1+0+0+0}{6} - 1.13 = -0.84$

재사용 프레임워크 이용 설계 시 재사 용성 $COR = 1.76 \times \frac{1+0+1+1+1+1}{6} - 1.13 = 0.34$

그림 20. 재사용 측정값

된 형태의 시스템이다. 따라서 기존의 소프트웨어 설계 및 개발 방식과 많은 차이점을 가지고 있다. 특히 임베디드 시스템 설계시 실시간성, 반응성, 소규모, 경량화, 안전성, 신뢰성, 견고성, 저비용 등의 요소들을 고려하여 설계해야 한다. 게다가 저비용을 고려하려면 재사용성이 높은 임베디드 시스템을 개발하여야 하는데 이는 프레임워크 기반의 개발 방식을 도입해야 그 효과를 기대할 수 있다. 그러나 지금까지 개발되고 있는 대부분의 임베디드 시스템들은 이러한 특성들을 제대로 반영하지 않고 있다. 따라서 개발된 임베디드 시스템 내의 코드들 가운데는 데드 코드들이 차지하는 비율이 전체 시스템이 20%를 상회하고 있을 정도의 문제점을 갖고 있다. 따라서 본 논문에서는 이러한 문제점 가운데 재사용성을 향상시키기 위한 전략으로 임베디드 시스템에 대한 재사용 프레임워크를 설계하였고, 이러한 재사용 프레임워크를 다양한 유형의 임베디드 시스템 설계 및 개발에 쉽게 확장 가능하도록 하기 위해 재사용 프레임워크 메타

모델을 제시하였다. 제시된 메타 모델은 특정 임베디드 시스템 만을 위한 것이 아니라 다양한 임베디드 시스템들의 공통된 사항들을 추상화 시킨 모델이기 때문에 특정 임베디드 시스템 개발시 유연하게 확장 설계할 수 있도록 해준다. 또한 특정 임베디드 시스템에 따라 가변적인 부분들에 대해 동적으로 플러그인 할 수 있도록 개방시켜 놓고 있기 때문에 다양한 임베디드 시스템 개발에 적용시킬 수 있다.

향후 연구과제로는 제시한 프레임워크 메타 모델을 기반으로 다양한 분야의 임베디드 시스템에 적용하여 메타 모델의 유연성과 재사용성을 향상시킬 수 있는 요소들을 보완 및 확장하여 보다 안정성 있는 재사용 프레임워크를 개발하는 것이다.

참 고 문 헌

[1] H. Comma, "A Software Design Method for Real-Time Systems," *Communications of ACM*, Vol.27, No.7, pp. 938-949, Sept. 1984.

[2] 김정환, "전세계 임베디드 소프트웨어 시장 동향," IITA 주간기술동향, 통권1107호, 2003.

[3] B. Selic, G. Gullekson, and P.T Ward, *Real-Time Object-Oriented Modeling*, John Wiley&Sons, 1999.

[4] 홍성수, "RSCA: 분산 로봇 플랫폼에서 임베디드 소프트웨어의 동적 재구성을 지원하는 통합 미들웨어," 한국통신학회지, 21권 10호, 2004.

[5] Eclipse, http://www.eclipse.org

[6] 박재성, *Spring 프레임워크 워크북*, 한빛미디어, 2006.

[7] Struts Framework, http://struts.apache.org

[8] E.S. David, *An Embedded Software Primer*, Addison Wesely, 1999.

[9] K. Raj, *Embedded Systems: Architecture, Programming and Design*, McGraw Hill, 2004.

[10] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," *IEEE Software*, pp. 37-45, Nov. 1998.

[11] Anastasopoulos M. and Gacek C., *Implementing Product Line Variabilties*, Technical Report IESE Report No. 089.00/E, Version 1.0, Fraunhofer Institute for Experimental Software Engineering (IESE), Nov. 2000.

[12] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 2002.

[13] K. Short, *Component Based Development and Object Modeling*, Sterling Software, Technical Handbook Version 1.0, Feb. 1997.

[14] M. E. Fayad and D. C. Schmidt, "Object-Oriented Application Frameworks," *Communication of the ACM*, Vol.40, No.10, pp. 32-42, Oct. 1997.

[15] F. Desmond, D'souza and A. C. Wills, *Objects, Components, and Frameworks with UML*, pp. 91-234, Addison-Wesley, 1998.

[16] Rational Software Corp., *Unified Modeling Language(UML) Summary*, 1997.

[17] OMG, *MDA Guide-Version1.0.1*, OMG, 2003.

[18] Hironori Washizaki, Hirokazu Yamamoto and Yoshiaki Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components," Proceedings of the Ninth International Software Metrics Symposium (METRICS'03), pp.1530-1435, IEEE, 2003.

조 은 숙

1993년 동의대학교 전산통계학과 졸업(이학사)

1996년 숭실대학교 대학원 컴퓨터학과(공학석사)

2000년 숭실대학교 대학원 컴퓨터학과(공학박사)

2000년~2005년 동덕여자대학교 정보학부 강의전임교수

2005년~현재 서일대학 소프트웨어과 조교수

관심분야 : CBSE, Embedded Software, Service-Oriented Computing, SOA

김 철 진

1996년 경기대학교 전자계산학과(학사)
1998년 숭실대학교 컴퓨터공학부(공학석사)
2004년 숭실대학교 컴퓨터공학부(공학박사)
2004년 가톨릭대학교 컴퓨터정보공학부 강의전담교수
2004년~현재 삼성전자 책임연구원
관심분야 : CBD, Component Customization, Embedded Software

이 숙 희

1979년 숙명여자대학교 독문학과 졸업(학사)
1982년 동국대학교 경영대학원 정보처리학과 졸업(석사)
1991년 성균관대학교 대학원 통계학과 졸업(박사)
1987년~1993년 동신대학교 전자계산학과 교수
1993년~현재 서경대학교 인터넷정보학과 교수
관심분야 : 객체지향 소프트웨어 개발 방법론, 소프트웨어 테스트, 컴포넌트기반 소프트웨어공학